John Fortado is a BI thought leader with over thirty years of IT experience with a focus on, architecting and integrating complex systems, networking, security, software design and quality assurance, customer support documentation and process improvement.


John is part of the growing BI analytics practice within Emtec, a leading IT consulting firm that provides transformational digital solutions. John holds degrees in Physics and Electrical Engineering.
OBIEE OPERATION WITH EBS IN THE SAME SECURITY CONTEXT

Integration of Oracle Business Intelligence Enterprise with Oracle Enterprise Business Suite is a straightforward process that requires an understanding of some rather obscure detail from several layers of enterprise including system servers, network domains and protocols, fusion middleware, weblogic, rpd variables, intiblocks, the physical layer, the object layer, the presentation layer, role and responsibilities permissions, data filters, dashboard administration, EBS Forms and Menu Security.

Integration of the two stand-alone and independent systems provides for seamless yet controlled flow between the systems allowing for drilldown and drillback from key metrics to underlying detail. Theoretically, this can be done between any OBIEE form and any EBS form with pass-thru to any connected EBS subsystem.

Corporate security in EBS and OBIEE including object authorization and row level data filtering within both systems determines accessibility and pass-thru capability. With drillback, getting to detail is a matter of clicking on a cell in a dashboard table activating an action link which takes the cell value back to a designated form within EBS and calls that form into a browser window for display. Drillback is ability to click on an object and be taken into the source system to the associated line item. Drilldown is the ability to click on an object on a dashboard and be taken to the underlying data.

If you are integrating OBIEE with EBS, you likely are using OBIA, Oracle Business Intelligence Analytics, although that is certainly not a requirement. OBIA is a prebuilt, prepackaged BI solution that delivers role based intelligence to the organization. It is a set of OBIEE dashboards and reports that run from a prebuilt warehouse previously serviced by Informatica/DAC. The next generation of the OBIA warehouse utilizes Oracle’s Data Integrator, ODI, which runs high volume batch load plans, event driven load plans, and even SOA data services as it runs on the Fusion layer as part of the application stack with OBIEE and can take advantage of an OBIEE distributed/clustered environment. The very latest in the chain is Oracle Golden Gate which is billed as real-time data extraction and replication. Putting Golden Gate in front of the ODI Source Dependent Data Store makes it fast. If you’re looking for speed, here it is. The BI Applications warehouse can now deliver metrics to you in near real-time and since it’s OBIEE, it can be delivered to your phone or mobile device.
PREPARATION

How is all this accomplished? A good place to start is with Oracle. There is now a technical document available from Oracle, Doc ID 555254.1, which outlines the process along with screenshots. The document illustrates important setup that is required within EBS System Administration.

There are couple of prerequisites if you are running EBS R12. Since EBS is usually accessed through SSL, OBIEE must also be setup with SSL on the analytics port. There is no need to put SSL on the OBIEE internals. Out of the box, secure communication is in place between the internal processes. Secondly, the EBS and OBIEE servers must be in the same domain. There are some “good to know” items. Such as, the designated user in the OBIEE R12 connection pool must have access to the EBS APP_SESSION functions and procedures, and the connection pool user accessing EBS security must have read access to the EBS security tables.

Any EBS form you choose to link from OBIEE must actually accept parameters if you intend to pass parameters. Many EBS forms do not accept parameters. Changing that is not a simple matter, and it will require close coordination with the EBS group developers. Calling such a form (in OBIEE, the EBS forms are called Function Codes as they are known by their Function Code in EBS) will result in the EBS form popping up along with an error that no such parameter exists. The EBS file that controls the form is generally named like this - <functioncode_name>.fmb - and can be found under the $AU_TOP/forms folder. Opening this file with Oracle Forms will allow you to find what parameters if any the form accepts.

Beware of calling for data from EBS that is not filtered, especially if you use Oracle’s example of an Opaque View as EBS systems contain very large datasets which OBIEE is not optimized to handle. Although OBIEE can operate on gigabytes of data and convert that into a spreadsheet with some very specialized tuning, it’s not the intended design.

Keep in mind when going through Oracle’s example, setting up the function, the menu, the responsibility, the user and the profile in EBS that the OBIEE/OBIA menu (the actual clickable menu link) needs to be placed under the responsibility and not at the same level as the Assigning Responsibility to the User. This link will be the “FND: Oracle Business Intelligence Suite EE base URL”.

SETUP

When creating or configuring init-blocks to retrieve data from EBS, always ensure that Row-wise initialization is checked as this allows multiple results to be stored in the variable regardless of whether the variable is static or dynamic. You will otherwise only retrieve the last item in the result set.

Finally, although it may sound obvious, be sure to set the Execution Precedence of
the EBS Integration initblock that attaches the session through the session cookie so that it fires before any attempt is made to retrieve security information.

Once EBS setup is complete, you will need to modify files and the RPD in order for OBIEE to attach to the EBS session with a custom connection pool, query EBS to retrieve responsibilities for the session user with a custom authorization initblock, and map the responsibilities into WebLogic Roles. Two important files must be modified in order for WebLogic to find, accept and attach to the EBS session.

Now to configure external authentication, you will need to modify instanceconfig.xml:

$ORACLE_INSTANCE/config/OracleBIPresentationServicesComponent/coreapplication_obips1/instanceconfig.xml

It will look like this when finished:

<Authentication>

<!--This Configuration setting is managed by Oracle Enterprise Manager Fusion Middleware Control-->

<EnabledSchemas>UidPwd,Impersonate,UidPwd-soap,Impersonate-soap,EBS-ICX</EnabledSchemas>

</Authentication>

Notice the EBS-ICX. This is the internal schema name that Middleware recognizes for the interface to EBS. The hard work has already been done by Oracle development. It is only necessary to make known to the system that this is to be utilized. Then, let the system know the name of the session cookie that EBS writes.

**ACCESSING THE COOKIE**

Now that OBIEE knows we are connecting with EBS, it will look for the EBS session cookie. The default/out-of-the-box session cookie name must be changed in authenticationschemas.xml:

$ORACLE_BI_HOME/bifoundation/web/display/authenticationschemas.xml

Change the value of the nameInSource parameter to match the name of the EBS cookie for NQ_SESSION.ICX_SESSION_COOKIE

<RequestVariable source="cookie" type="auth" nameInSource="CookieName" biVariableName="NQ_SESSION.ICX_SESSION_COOKIE"/>

Then, in this same file, also find and change the SchemaKeyVariable for the source="cookie" to set the value of its nameInSource parameter to the correct cookie name:

<SchemaKeyVariable source="cookie" nameInSource="CookieName" forceValue="EBS-ICX"/>
Don’t know the cookie name? Taking the suggestion, login to EBS then type or paste into the browser url dialog `javascript:alert(document.cookie)` and press return. The alert will popup showing the current cookies, one of which will be the EBS session cookie. Although the name may not be obvious, it is sometimes the same as the EBS db reference. It is the value to the left of the equal sign, and yes Case does matter. The value, the part on the right of the equal sign, isn’t important unless testing the call to `APP_SESSION` which is made in the connection pool to EBS.

Finally comes the part that is familiar to every OBIEE administrator, the RPD modifications. If you are following the document, the sample EBS connection pool can be used or create a new one just for the initialization process and retrieving security.

Either way, as noted, the important part of the connection pool definition is to add the correct call to `APP_SESSION` to the connection pool Connection Script which gets fired on connection to the EBS database:

```
call apps.APP_SESSION.validate_icx_session('valueof(NQ_SESSION.ICX_SESSION_COOKIE)')
```

Be sure that the script is enabled. The user name specified for this connection pool must have access to the APP_SESSION package. If there is a problem, get with the dba and test the call with the APPS user (or EBS superuser). If the problem persists, it could be due to a missing patch on EBS. So, verify that EBS patches are up-to-date.

fires before any attempt is made to retrieve security information.

Once EBS setup is complete, you will need to modify files and the RPD in order for OBIEE to attach to the EBS session with a custom connection pool, query EBS to retrieve responsibilities for the session user with a custom authorization initblock, and map the responsibilities into WebLogic Roles. Two important files must be modified in order for WebLogic to find, accept and attach to the EBS session.

Now to configure external authentication, you will need to modify `instanceconfig.xml`:

```
$ORACLE_INSTANCE/config/OracleBIPresentationServicesComponent/coreapplication_obips1/instanceconfig.xml
```

It will look like this when finished:

```xml
<Authentication>

<!--This Configuration setting is managed by Oracle Enterprise Manager Fusion Middleware Control-->"
Notice the EBS-ICX. This is the internal schema name that Middleware recognizes for the interface to EBS. The hard work has already been done by Oracle development. It is only necessary to make known to the system that this is to be utilized. Then, let the system know the name of the session cookie that EBS writes.

Don’t know the cookie name? Taking the suggestion, login to EBS then type or paste into the browser url dialog javascript:alert(document.cookie) and press return. The alert will popup showing the current cookies, one of which will be the EBS session cookie. Although the name may not be obvious, it is sometimes the same as the EBS db reference. It is the value to the left of the equal sign, and yes Case does matter. The value, the part on the right of the equal sign, isn’t important unless testing the call to APP_SESSION which is made in the connection pool to EBS.

Finally comes the part that is familiar to every OBIEE administrator, the RPD modifications. If you are following the document, the sample EBS connection pool can be used or create a new one just for the initialization process and retrieving security.

Either way, as noted, the im<!--This Configuration setting is managed by Oracle Enterprise Manager Fusion Middleware Control--> <EnabledSchemas>UidPw,Impersonate,UidPw-soap,Impersonate-soap,EBS-ICX</EnabledSchemas> </Authentication>

Notice the EBS-ICX. This is the internal schema name that Middleware recognizes for the interface to EBS. The hard work has already been done by Oracle development. It is only necessary to make known to the system that this is to be utilized. Then, let the system know the name of the session cookie that EBS writes.

INITBLOCKS

Now, create an initblock which will use this connection pool to retrieve the EBS context and set that into OBIEE session variables. The initblock will use the connection pool just defined and will send a datasource query to the EBS database:

```sql
SELECT
    FND_GLOBAL.RESP_ID,
    FND_GLOBAL.RESP_APPL_ID,
    FND_GLOBAL.SECURITY_GROUP_ID,
```
Referring to another Oracle document, 1539742.1, create these static session variables to hold the context:

- `EBS_RESP_ID`
- `EBS_RESP_APPL_ID`
- `EBS_SEC_GROUP_ID`
- `EBS_RESP_NAME`
- `EBS_USER_ID`
- `EBS_EMPLOYEE_ID`
- `USER`

If you created a new initblock and called it Oracle EBS SSO, it should look like this:

Notice the EBS-ICX. This is the internal schema name that Middleware recognizes for the interface to EBS. The hard work has already been done by Oracle development. It is only necessary to make known to the system that this is to be utilized. Then, let the system know the name of the session cookie that EBS writes.

Don’t know the cookie name? Taking the suggestion, login to EBS then type or paste into the browser url dialog `javascript:alert(document.cookie)` and press return. The alert will popup showing the current cookies, one of which will be the EBS session cookie. Although the name may not be obvious, it is sometimes the same as the EBS db reference. It is the value to the left of the equal sign, and yes Case does matter. The value, the part on the right of the equal sign, isn’t important unless testing the call to `APP_SESSION` which is made in the connection pool to EBS.

Finally comes the part that is familiar to every OBIEE administrator, the RPD modifications. If you are following the document, the sample EBS connection pool can be used or create a new one just for the initialization process and retrieving security.
Either wFND_GLOBAL.EMPLOYEE_ID,

FND_GLOBAL.USER_NAME,

There are two things to note. Row-wise initialization is not checked. This means only one value will be stored in each variable. Checking Row-wise would allow multiple values to be retrieved in a semicolon delimited string, but we do not want that here. Also note that one variable, EBS responsibility, is missing. This needs to go into a dynamic variable. We will use the Authorization block to retrieve the EBS responsibilities, as there can be multiple responsibilities assigned to a user.

Finally, make the modifications to the Authorization block which will retrieve the EBS responsibilities. The out-of-the-box Authorization block will need to be changed to connect through the EBS connection pool and send the responsibility_name query to EBS, retrieving as many values as required.

SELECT DISTINCT 'ROLES',
RESPONSIBILITY_NAME
FROM
FND_USER,FND_USER_RESP_GROUPS, FND_RESPONSIBILITY_VL
WHERE
FND_USER.user_id=FND_USER_RESP_GROUPS.user_id AND
FND_USER_RESP_GROUPS.RESPONSIBILITY_ID = FND_RESPONSIBILITY_VL.RESPONSIBILITY_ID
AND FND_USER_RESP_GROUPS.RESPONSIBILITY_APPLICATION_ID = FND_RESPONSIBILITY_VL.APPLICATION_ID AND
FND_USER_RESP_GROUPS.START_DATE < SYSDATE AND
(CASE WHEN FND_USER_RESP_GROUPS.END_DATE IS NULL THEN SYSDATE ELSE TO_DATE(FND_USER_RESP_GROUPS.end_Date) END) >= SYSDATE
AND FND_USER.user_id = (SELECT USER_ID FROM FND_USER WHERE
UPPER(USER_NAME) = UPPER('VALUEOF(NQ_SESSION.USER)')) ;

Note here that there is no static variable target and that we are doing row-wise initialization. The result set will be placed in a dynamic variable called ROLES which will be visible from the Admin Tool Identity Manager when looking at current session variables.
FND_GLOBAL.EMPLOYEE_ID,
FND_GLOBAL.USER_NAME,
FND_GLOBAL.RESP_NAME
FROM DUAL

Ritiation process and retrieving security.

Either wFND_GLOBAL.EMPLOYEE_ID,
FND_GLOBAL.USER_NAME,
be retrieved in a semicolon delimited string, but we do not want that here. Also note that one variable, EBS responsibility, is missing. This needs to go into a dynamic variable. We will use the Authorization block to retrieve the EBS responsibilities, as there can be multiple responsibilities assigned to a user.

Finally, make the modifications to the Authorization block which will retrieve the EBS responsibilities. The out-of-the-box Authorization block will need to be changed to connect through the EBS connection pool and send the responsibility_name query to EBS, retFND_GLOBAL.USER_NAME,
FND_GLOBAL.RESP_NAME
FROM DUAL

Ritiation process and retrieving security.

Either wFND_GLOBAL.EMPLOYEE_ID,
FND_GLOBAL.USER_NAME,

CLOSING

That gets OBIEE and EBS operational within the same session, meaning the user can move transparently between OBIEE and EBS without multiple logins and while maintaining the same profile.

Linking back to an object in EBS is done by configuring a clickable OBIEE Action Link either on a report or underlying a data column or data value.

Of course, the WebLogic roles must match the EBS responsibilities by name. If you haven’t added the role definitions through Enterprise Manager, that needs to be done. As we are not connecting OBIEE to Active Directory, there is no role mapping to groups involved. However, you must assign the WebLogic roles permissions on OBIEE objects and to the Presentation Layer as usual. If you are implementing row-level security on the data, that is an entirely new discussion involving Security Filters on logical tables in the RPD.
REFERENCES:

Doc ID 555254.1 “Integrating Oracle Business Intelligence Applications with Oracle E-Business Suite”

Doc ID 1343143.1 OBIEE 11G: Integrating Oracle Business Intelligence Applications (OBIA) With Oracle E-Business Suite

Doc ID 1457278.1 “OBI EE Action Links To EBS Are Not Working…”

Doc ID 1527849.1 “OBIEE 11g: Link to Navigate to E-Business Suite is not Visible”

Oracle® Fusion Middleware Integrator’s Guide for Oracle Business Intelligence Enterprise Edition

11g Release 1 (11.1.1) E16364-01

Integrating Oracle Business Intelligence Applications with Oracle E-Business Suite, Saurabh Gautam

Oracle® Fusion Intelligence For E-Business Suite Implementation Guide Release 12 Part No. E05295-02