



www.rittmanmead.com

CONSULTANCY TRAINING SUPPORT

OBIEE Deployment & Change Mgmt Best Practices

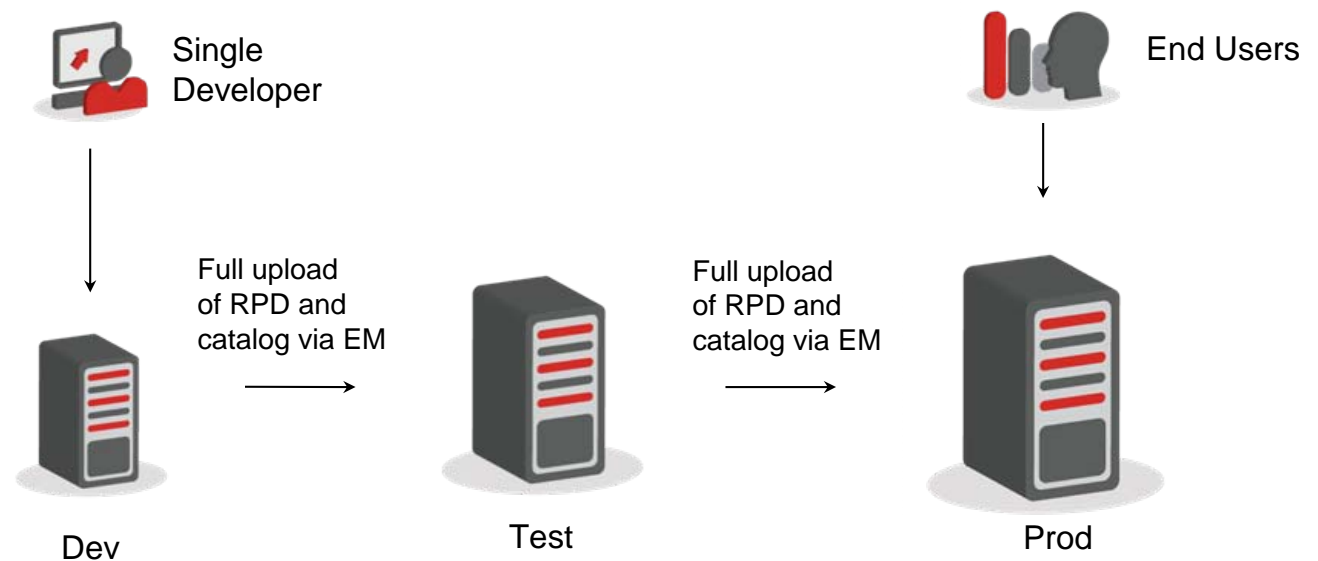
Mark Rittman, Technical Director, Rittman Mead
ODTUG BI/EPM Seriously Practical Conference, Sydney 2011

Elements of an Oracle BI EE 11g Project

- Oracle BI Repository (RPD file)
- Oracle BI Presentation Catalog
- System Configuration Settings
- UI Customizations
- Security artifacts (application roles, users, directory settings)
- Plus associated database schemas, ETL packages etc (out of scope for this though)

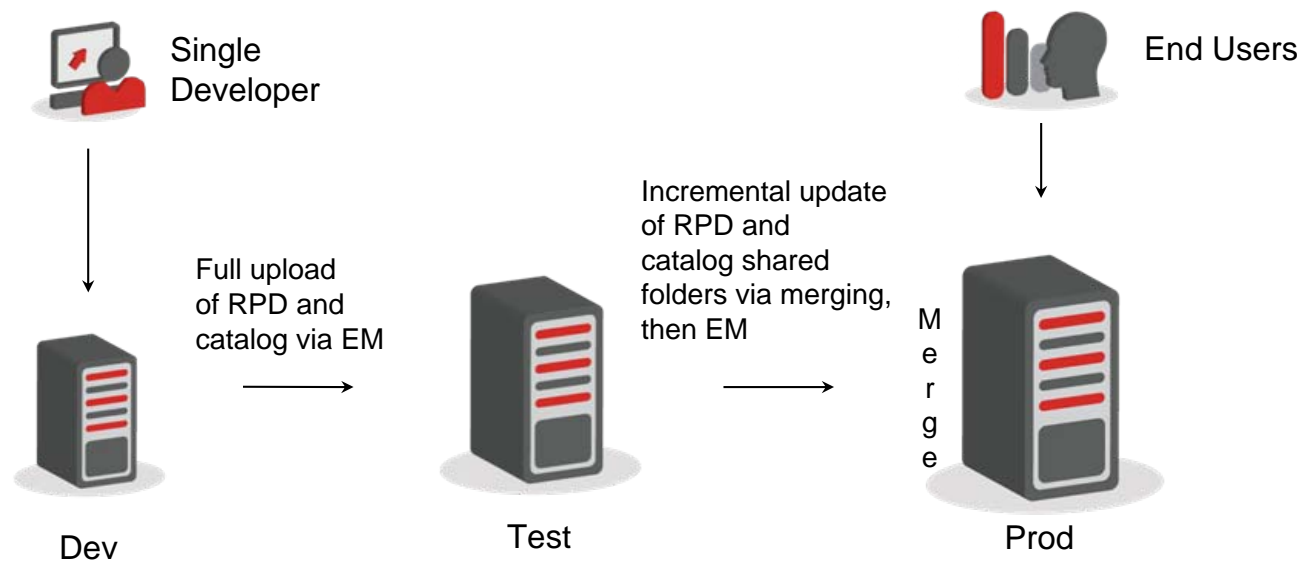
OBIEE 11g Project Lifecycle Stage #1 - Early Days

- Prototype to first production
- Typically a single developer, no version-control
- Initial project is moved from DEV server into PROD once first phase complete



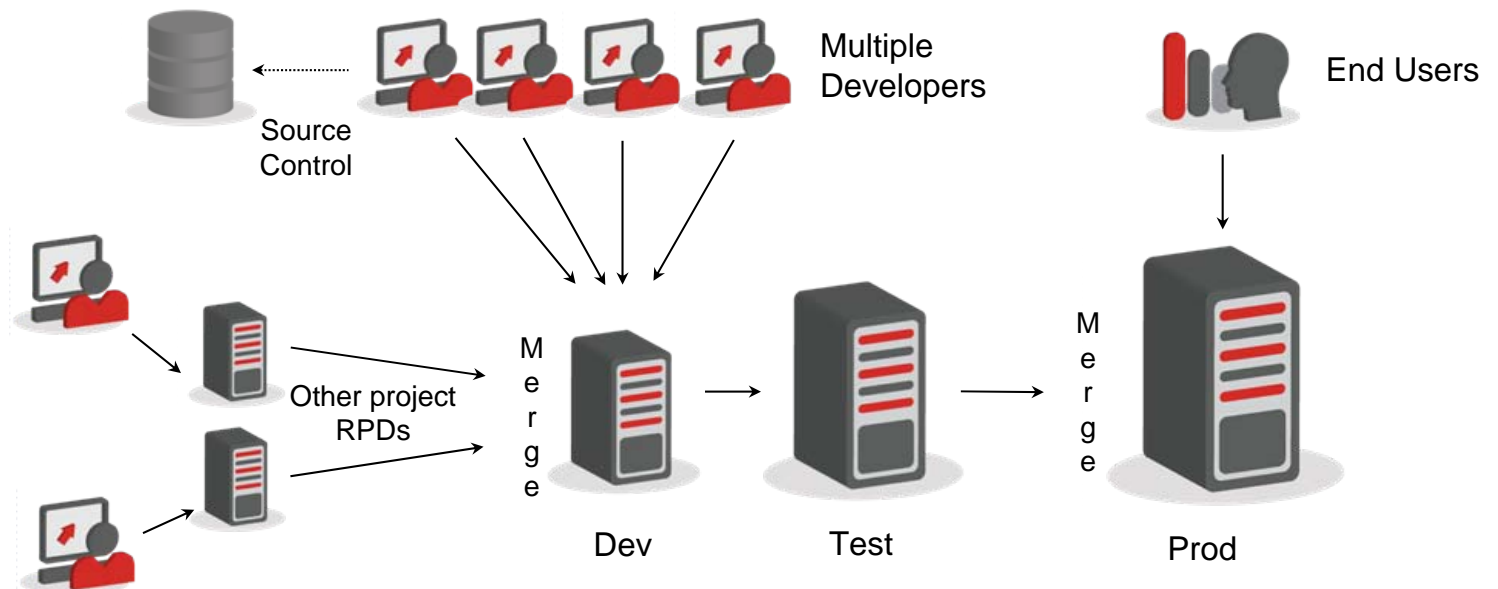
OBIEE 11g Project Lifecycle Stage #2 - Further Releases

- Updates to this first release, to add new RPD objects, shared folder catalog objects
- Incremental metadata needs to be merged into PROD, keeping existing objects
- Uses the three-way merge features for the RPD and the catalog



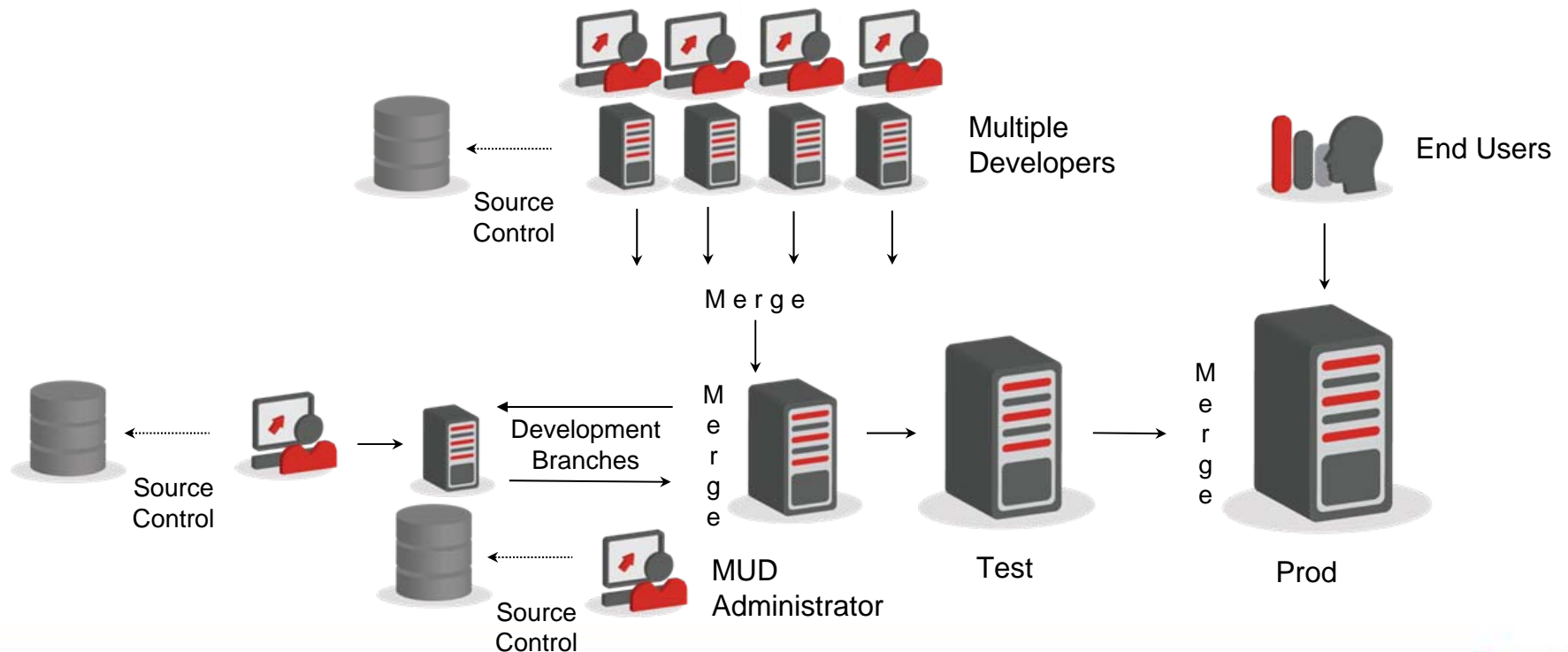
OBIEE 11g Project Lifecycle Stage #3 - Project Expands Out

- Additional developers wish to add content to the RPD
- Typically all developers on the project start accessing the RPD online, concurrently
- Other separately developed projects may need to be merged into the main RPD
- Version control becomes important as multiple developers start contributing changes



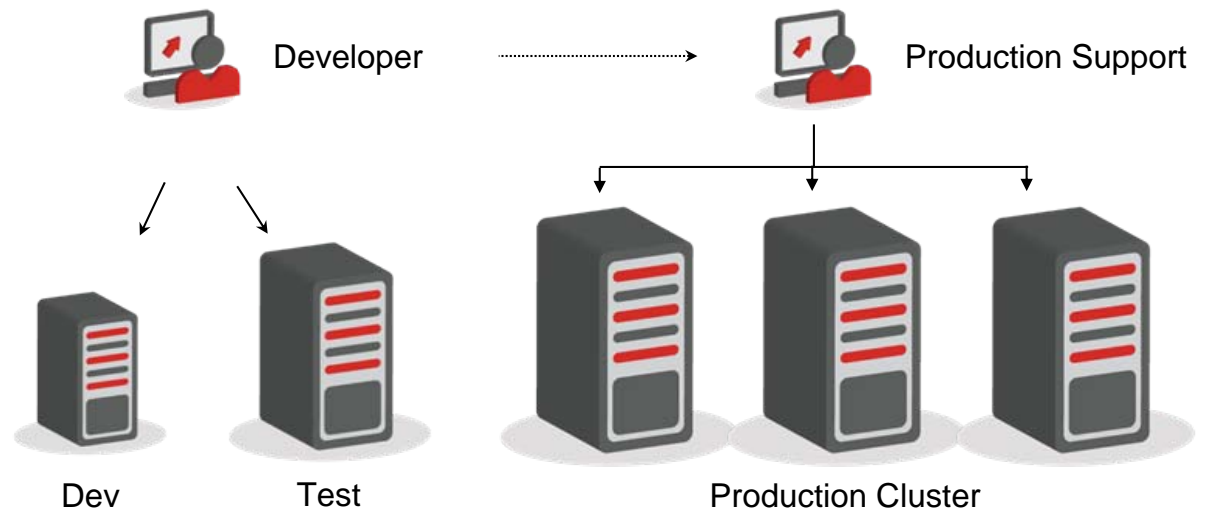
OBIEE 11g Project Lifecycle Stage #4 - Enterprise Deployment

- Soon, ad-hoc merging of RPDs and shared online development becomes unworkable
- A system needs to be put in place to handle distributed development
- Multi-User Development (MUD) Environment then becomes an option



Propagating System Configuration Changes

- At various points, system configuration changes have to be applied to BIEE environments
 - Deploying new repositories, presentation catalogs
 - Enabling SSL, new connections to directories (AD) etc
 - Changing performance parameters
- All changes have to be applied to all nodes in a cluster, possibly with rolling-restarts

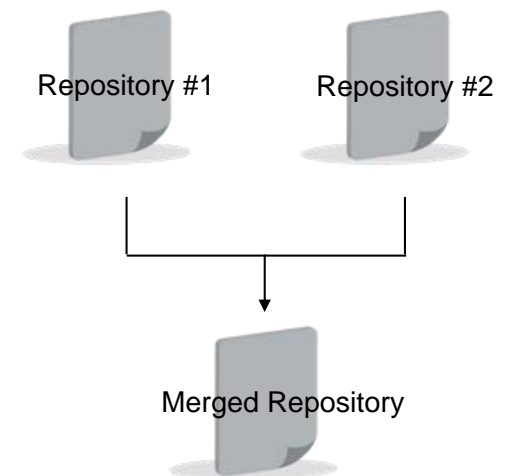


BI EE Features to Support Change Management & Deployment

- Three-way merges of repository files
- Catalog archiving/unarchiving
- *New in 11g* - repository and catalog patching
- Multi-User Development Environment
- *New in 11g* - Enterprise Manager Fusion Middleware Control (“EM”)
- *New in 11g* - WebLogic Server Scripting Tool & Oracle BI Systems Management API

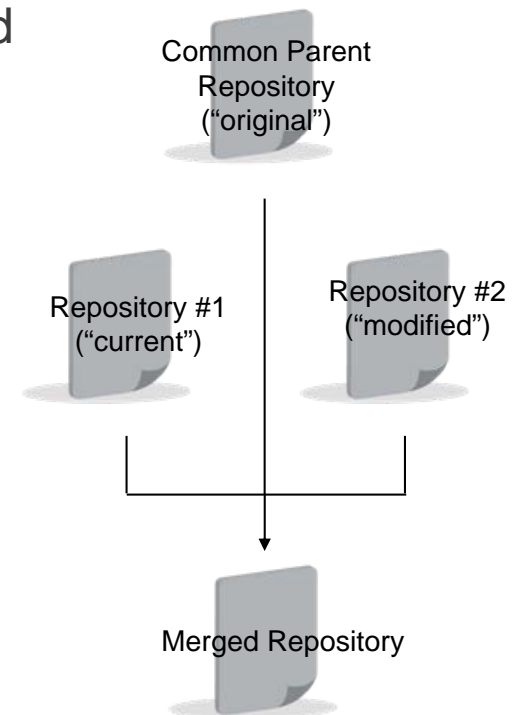
Merging Repository Files (RPDs)

- Merging repositories is a common task on projects past the initial stage
 - To merge new and changed objects in DEV into the PROD repository
 - To merge two RPDs into one, to run online in PROD
- Common task in general software development projects, with common complications
 - Repositories may contain similarly-named objects, but logically different
 - Repository objects may have changed in both DEV and PROD - which do you choose?
 - These, and others, are called “merge conflicts”



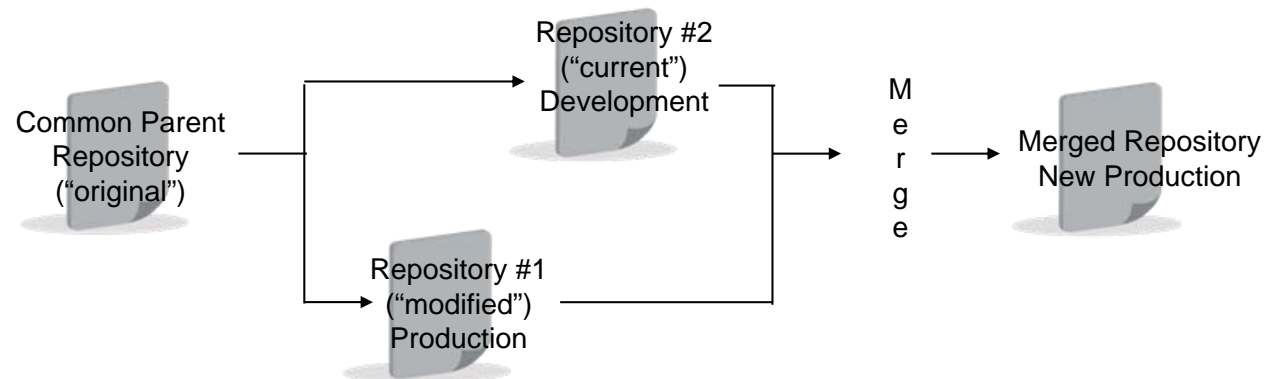
Oracle BI Repository Three-Way Merges

- Oracle BI, like many software development tools, uses the concept of three-way merges
 - A **modified** repository, which is usually the PROD repository
 - A **current** repository, which is usually the DEV repository
 - An **original** repository, from which they were both derived
- Provides a number of benefits compared to 2-way merges
 - Avoids “guessing” whether objects with the same name are actually logically the same
 - For branching development, allows both branches to be updated with subsequent changes to the original
 - More accurate and efficient way of merging two sets of objects with common parentage
- If no common repository available, then substitute blank repository (and lose the 3-way merge benefits)



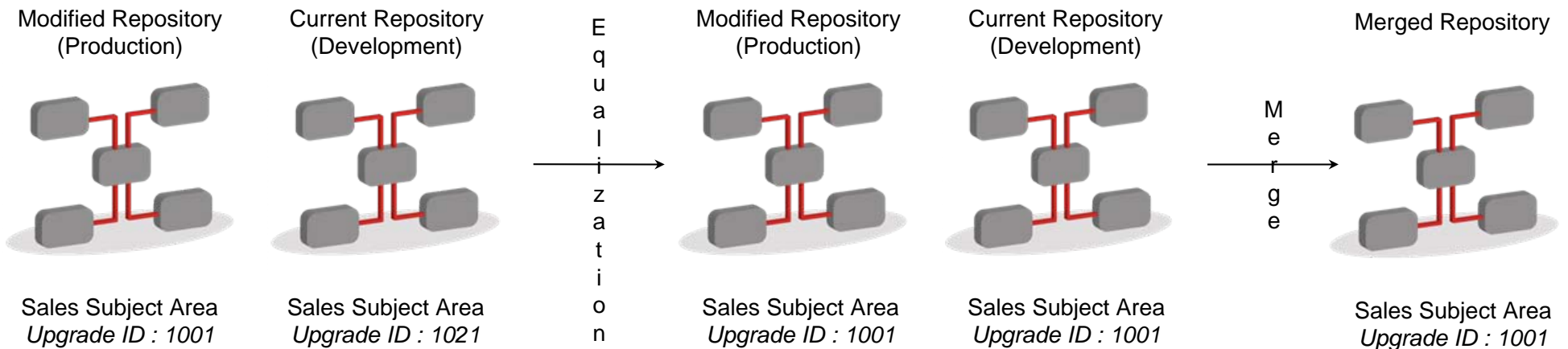
Understanding Merge Rules

- The merge process makes most sense when you understand the merge rules
- The RPDs you select for modified and current are important, do not choose at random
 - Current = development, Modified = production
- Rules assume that changes added to modified want to be preserved
- Deletions in current that are still in modified have to be confirmed
- Additions added, or deletions from, both repositories are automatically propagated
- Objects added to both, but with differences, cause a merge conflict
- Objects modified in both cause a merge conflict



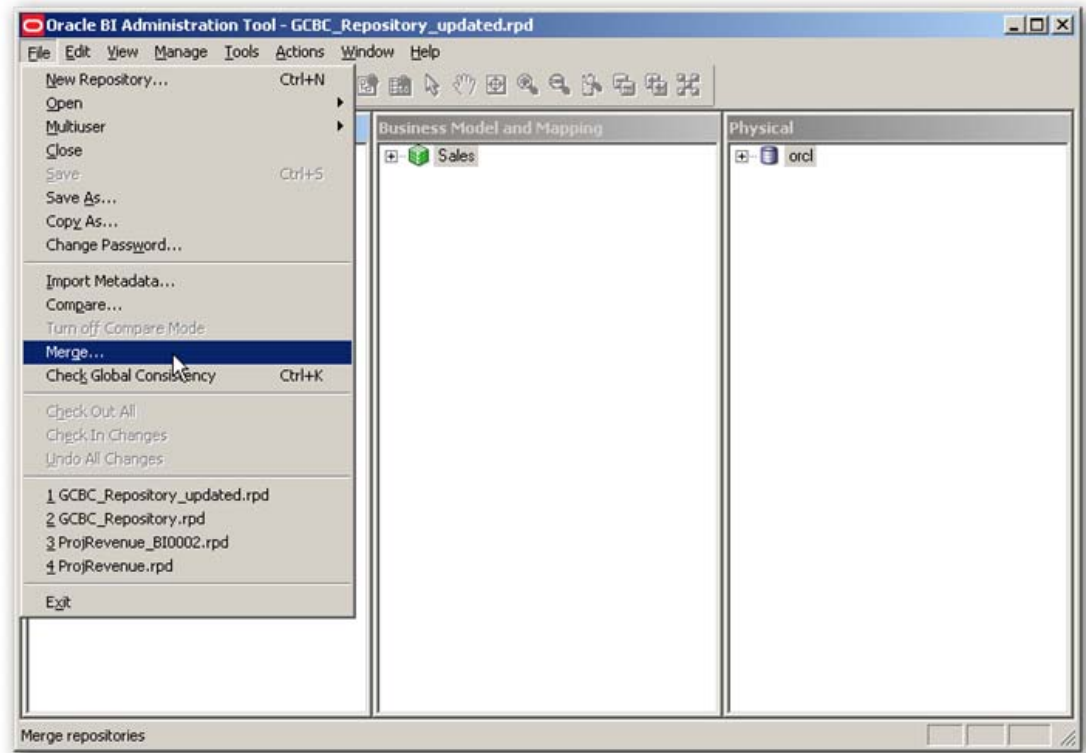
Repository Equalization

- Repositories may contain logically identical objects that have different *upgrade IDs*
 - *Upgrade IDs* are internal ID codes for objects in the repository
- Can be caused by deleting, then recreating, the same object
- Mismatching upgrade IDs can cause the upgrade process to create duplicates in the merge repository, thinking that the two objects are completely different
- Answer is to equalize the repositories



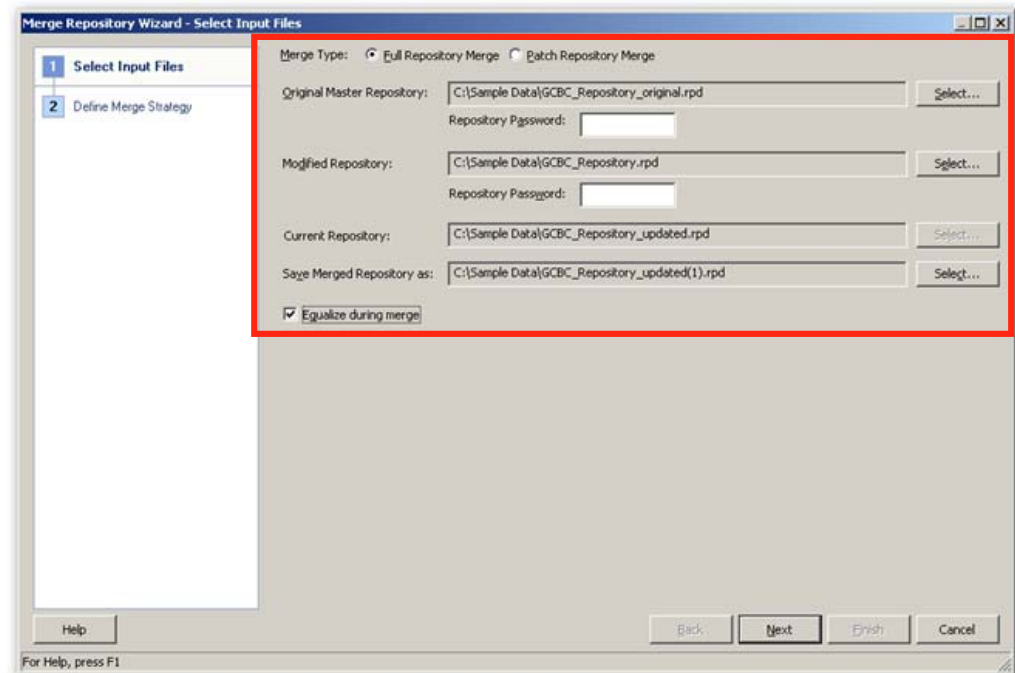
Three-Way RPD Merge Step 1 : Open Current RPD, Select Merge

- Start the BI Administration tool
- Open the **Current** (typically, Development) repository offline
- Select **File > Merge...**



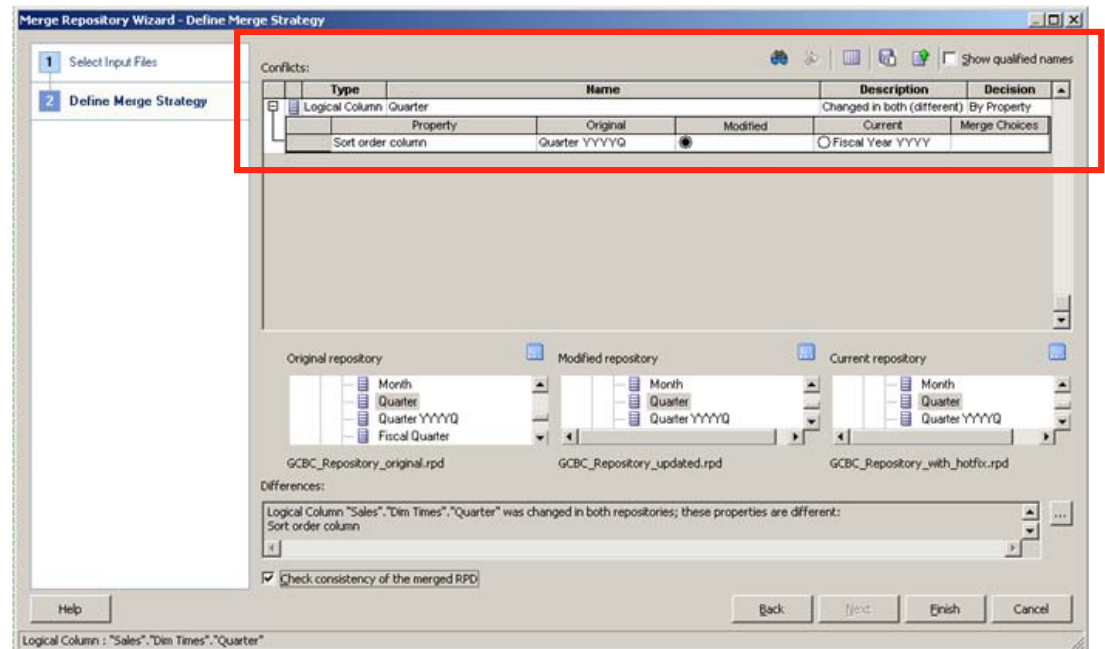
Three-Way RPD Merge Step 2 : Select Modified & Current RPDs

- With the **Merge Repository Wizard - Select Import Files** dialog open, select the modified (production) and original (common parent) RPDs
- Enter passwords
- Tick the **Equalize during merge** checkbox
- Select **Full Repository Merge**



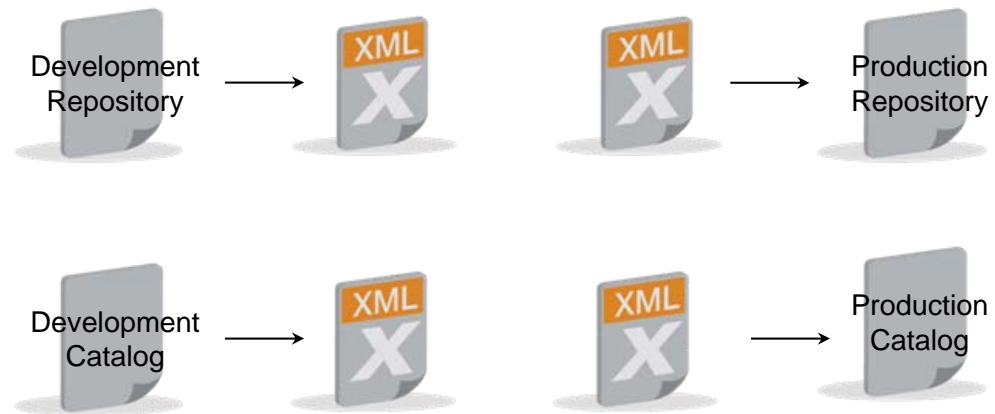
Three-Way RPD Merge Step 3 : Resolve Conflicts

- Conflicts typically occur if a choice needs to be made between two options
- Select the choice either from the modified (prod) or current (dev) repository
- Choices can go down to the object property level
- Once all conflicts resolved, merged repository is then opened for editing



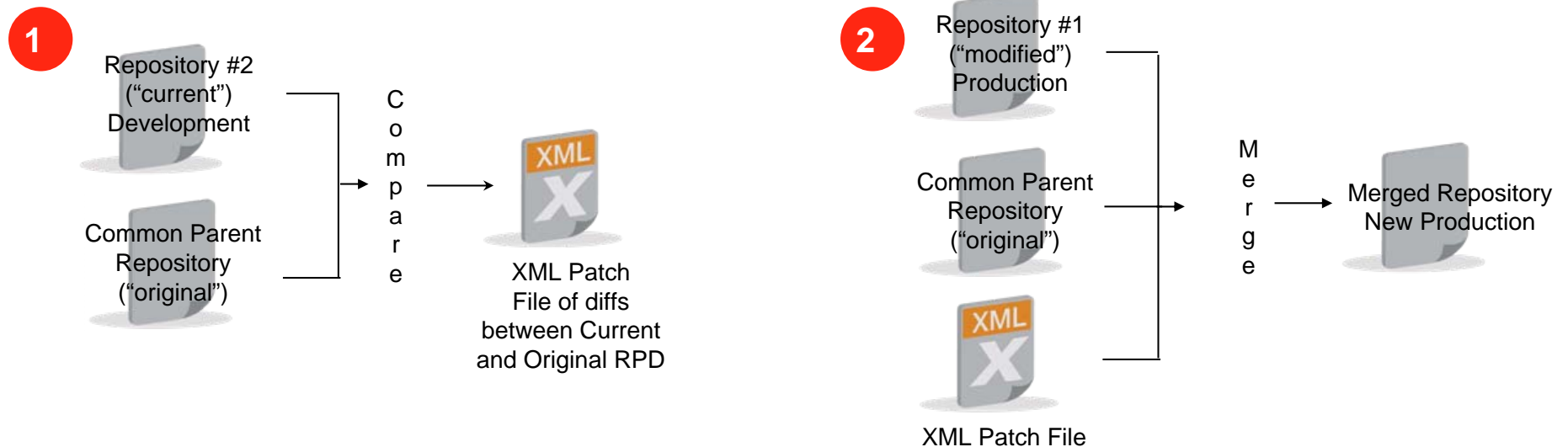
New in 11g : Repository and Catalog Patching

- In some situations, you want to perform the merge hands-off
- To remove opportunity for human error, to allow it to be scripted
- 11g introduces the concept of repository (RPD) and catalog patching
- Whole process, from extracting changes to patching target, can be scripted



Repository Patching

- Repository patching is a two-step process
 1. Compare the current repository to the original one, create an XML patch file of the differences between the two
 2. Apply the XML patch file to the modified repository, as a three-way merge also using the original repository
- Can be performed using BI Administrator tool, or from the command-line

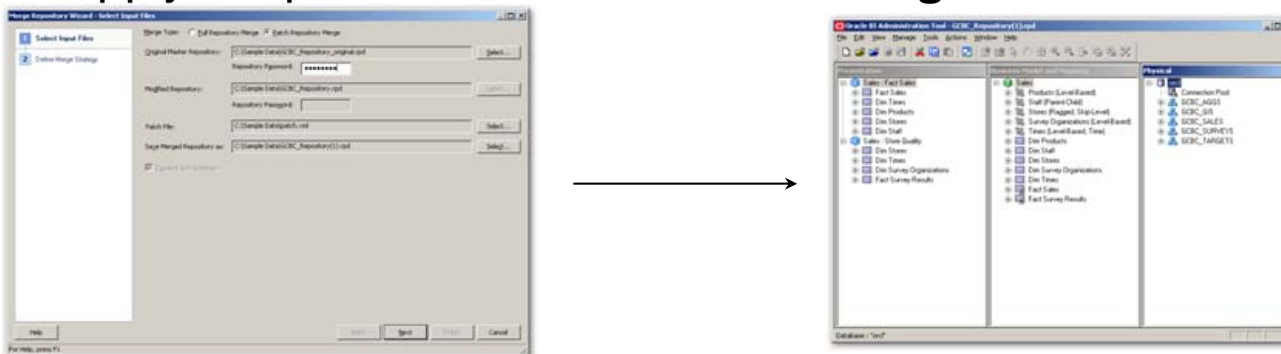


Interactive Repository Patching using the BI Administration Tool

- To create the XML patch file, use the **File > Compare...** feature



- To apply the patch file, use the **File > Merge...** feature



Command-Line Creation and Applying of RPD Patches

- Command-line utilities are available for creating, and applying, RPD patch files
- **comparerpd** creates a patch file based on current and original repositories

```
comparerpd -P [current repository password] -C [current repository path and name]  
- W [original repository password] -G [original repository path and name]  
-D [patch file path and name]
```

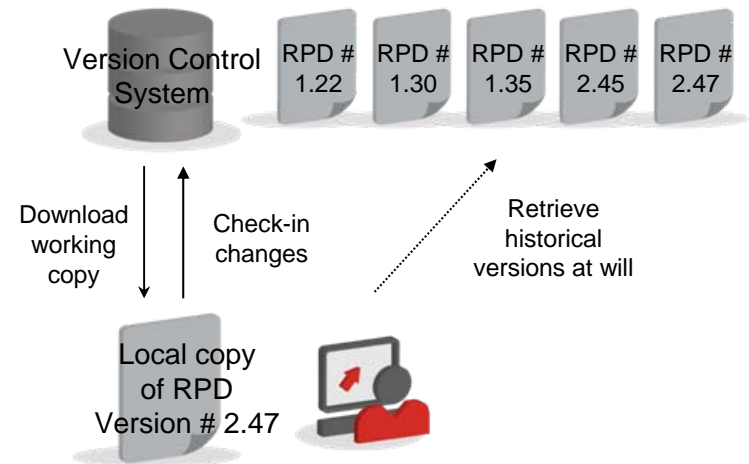
- **patchrpd** does a three-way merge with the patch file, original and modified repositories

```
patchrpd -P [modified repository password] -C [modified repository path and name]  
-Q [original repository password] -G [original repository path and name] -I [patch  
file path and name] -O [new repository path and name]
```

- Both located at `[middleware_home]\Oracle_BI1\bifoundation\server\bin\`

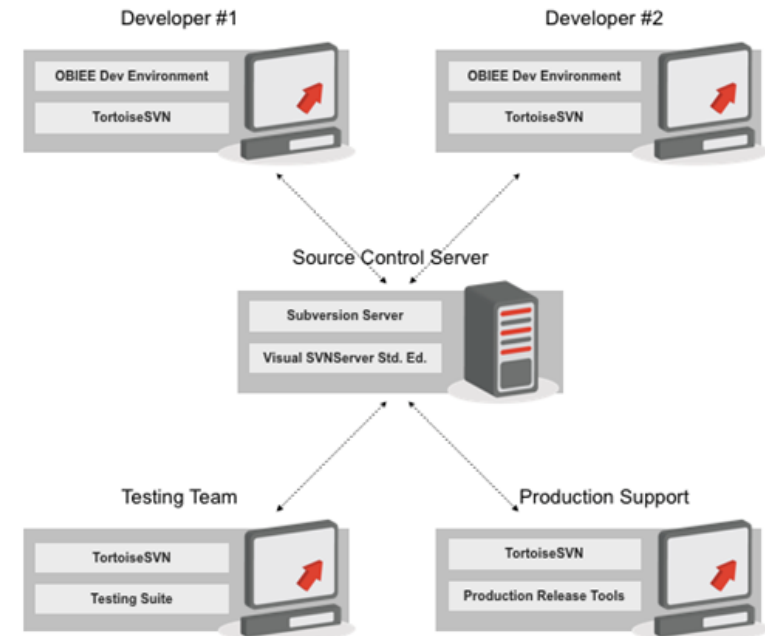
Version Control in Software Development Projects

- Version control is a common concept in software development
- Allows you to store copies (versions) of project elements over time
- Refer back to old versions, restore old versions, create named/numbered releases
- Branch projects, re-combine branches
- Typically performed using tools such as PVCS, Subversion, Git, Visual Sourcesafe



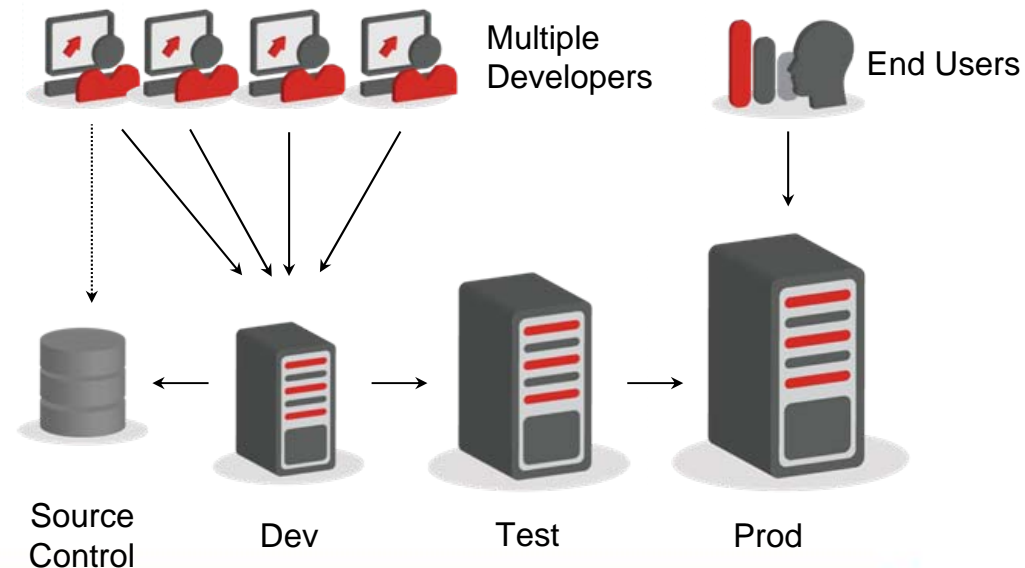
Subversion and OBIEE 11g

- OBIEE 11.1.1.5 does not have in-built integration with version or source control
- But you can store the various project artifacts in any version control tool
- Subversion, together with VisualSVN Server and TortoiseSVN, are suitable tools
- There are however some limitations
 - ▶ The RPD has to be uploaded in its entirety
 - ▶ Although you can also upload XML patch files
 - ▶ The Catalog has to be archived before uploading
 - ▶ You cannot use the merge/patch facility in SVN, you must use BI Administrator / Catalog Manager patch/merge instead



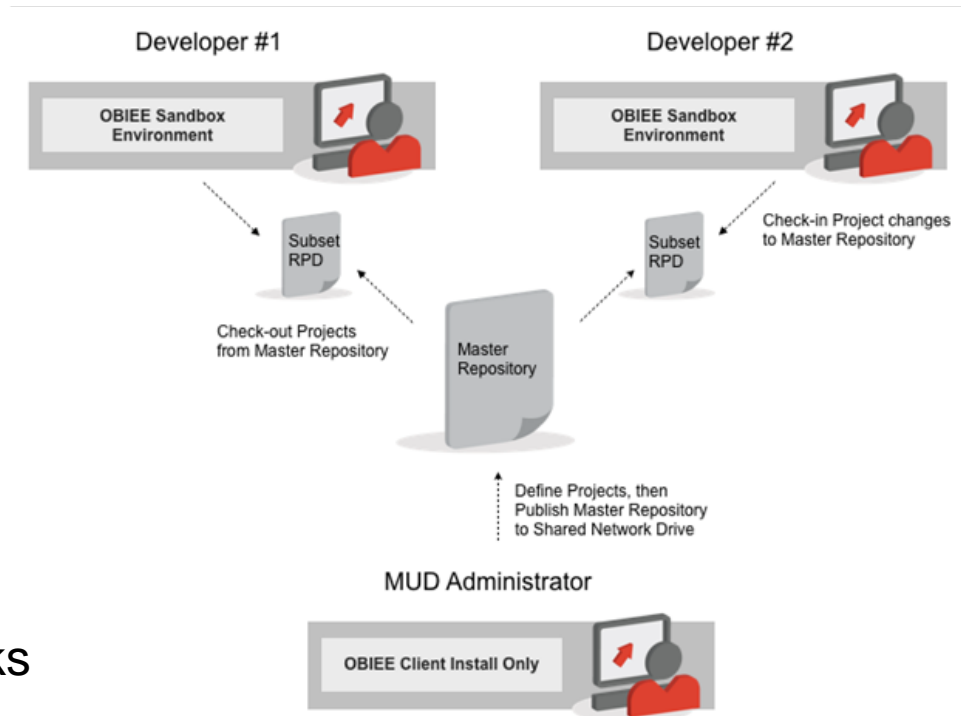
Managing Change with Large Teams of Developers

- So far, we have looked at projects where there is a single developer
- On many projects though, you wish to scale-up developers to deliver larger scope
- The catalog supports multiple developers editing, adding objects etc
- For smaller teams, you might consider concurrent online editing of the RPD
 - Has the virtue of simplicity
 - 11g certifies up to 5 concurrent developers
 - Works through a system of check-out and check-in of objects
 - Check-out is coarse-grained though
 - Edits to a logical table lock the whole business model

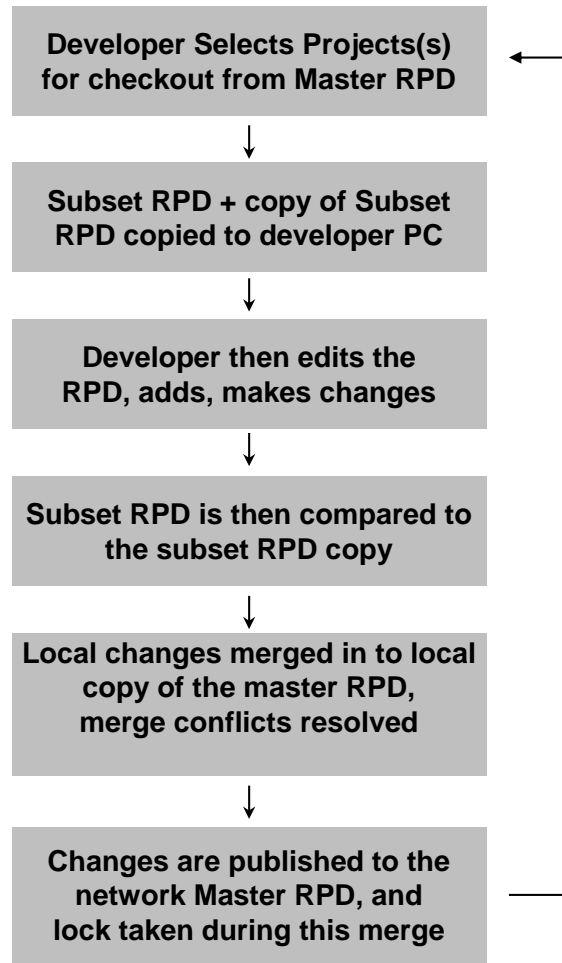


Multi-User Development Environment (MUD)

- MUD Administrator divides main repository into projects; self-contained RPD subsets
- Master repository is then published to a network share
- Projects are then worked on independently, and then merged back into the master RPD
- Uses the repository compare and merge features under the covers
- Works best when each developer has a full OBIEE “Sandbox” environment
 - License considerations through - may require named user plus licensing to be financially viable
- More complex than online development, but makes sense when you know how it works



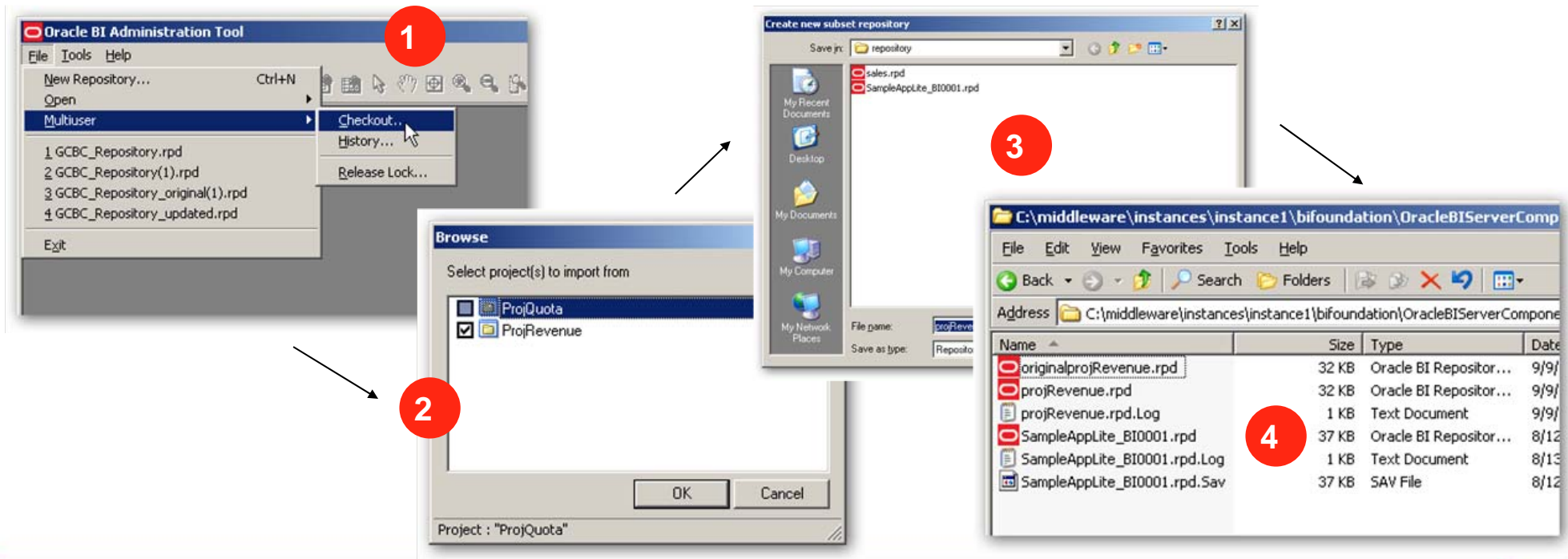
What Happens During MUD Check-Out / Merge / Check-In?



Change in 11g compared to 10g :
Locking only now takes place
at the publish step, not merge of local
changes

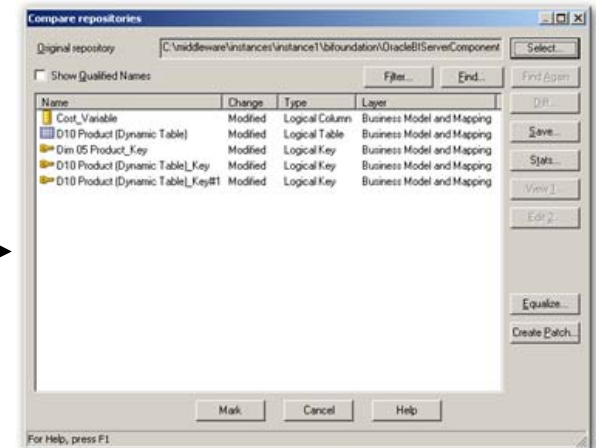
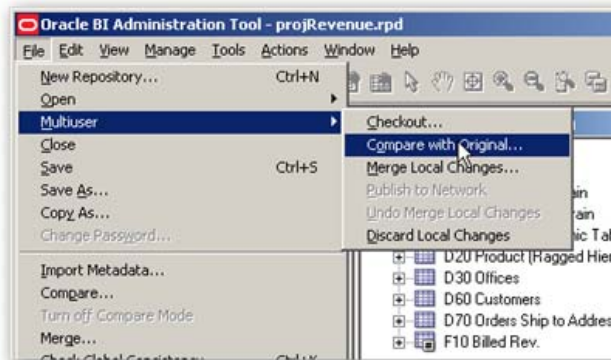
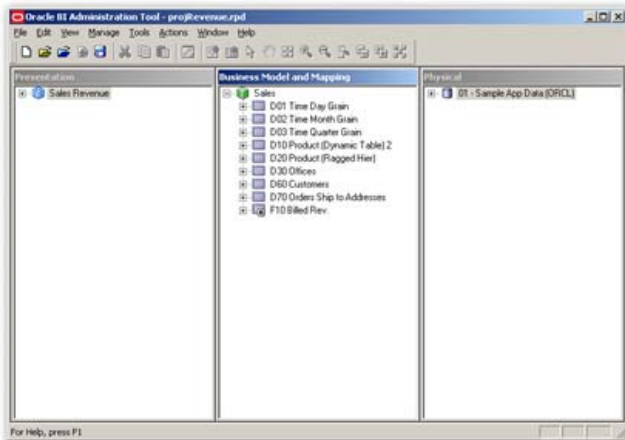
MUD Lifecycle Step 1 : Select and Checkout Project

- From the developer workstation, select **File > Multiuser > Checkout...**
- Select the project(s) to check-out
- Name the subset RPD file (a temporary copy of the whole RPD is made here)
- On save the subset RPD, plus a duplicate, is saved and the temporary copy removed



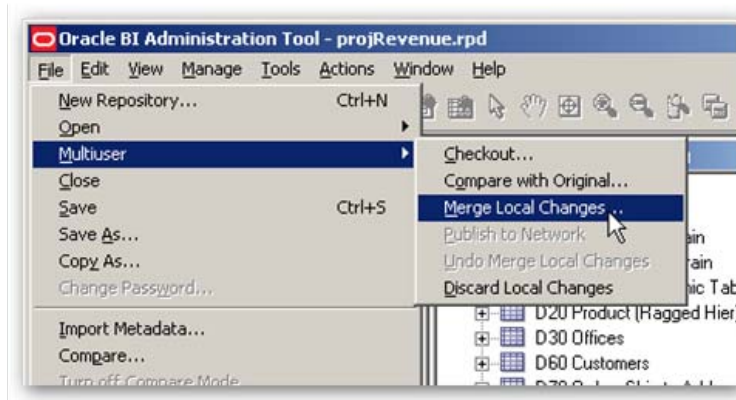
MUD Lifecycle Step 2 : Make Changes to Subset, Do Compare

- Make changes to the subset RPD, such as adding, deleting or modifying objects
- After a time, use **File > Multiuser > Compare with Original...**
 - Performs an automatic **File > Compare...** with the duplicate subset RPD
- Save your changes as normal
- Upload to a sandbox OBIEE environment and run online if required

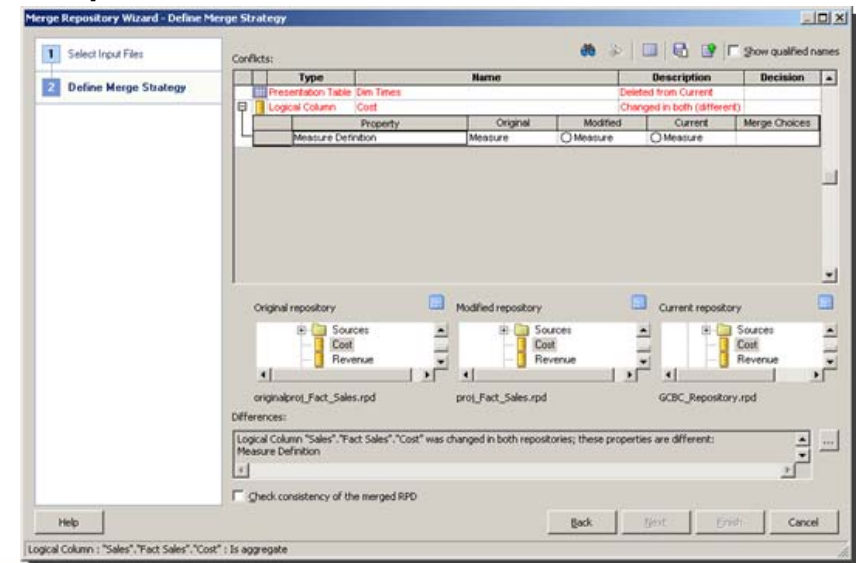


MUD Lifecycle Step 3 : Merge in Local Changes

- Once work is complete, select File > Compare > Merge in Local Changes
- Merges your subset RPD in with a fresh copy of the master RPD
 - Performed using an automatic three-way merge
 - If there are merge conflicts, this is where you will deal with them
- Merged results are then stored locally until published by the next step
- *Change compared to 10g: lock is not taken at this step*

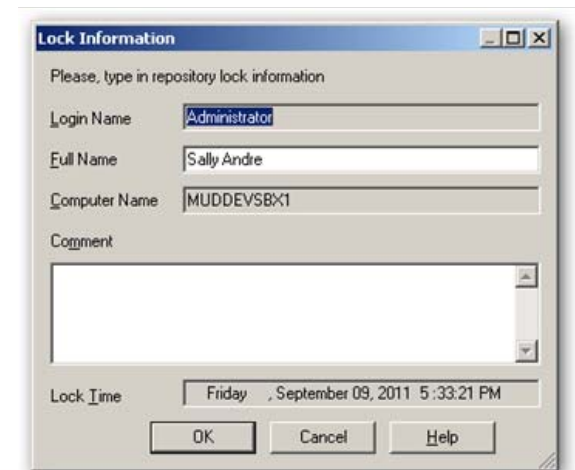
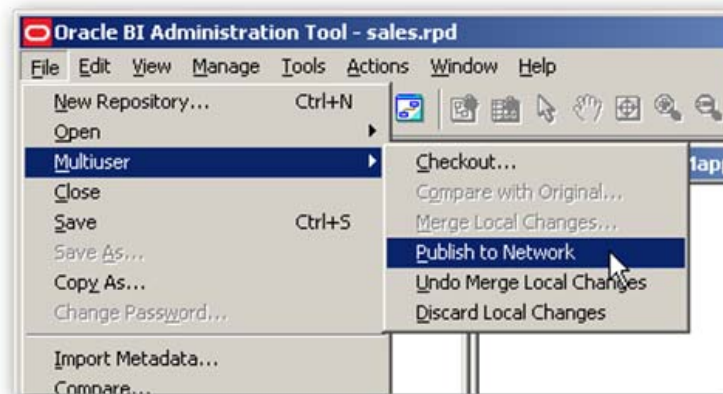


Define Merge Strategy only displayed if merge conflicts encountered



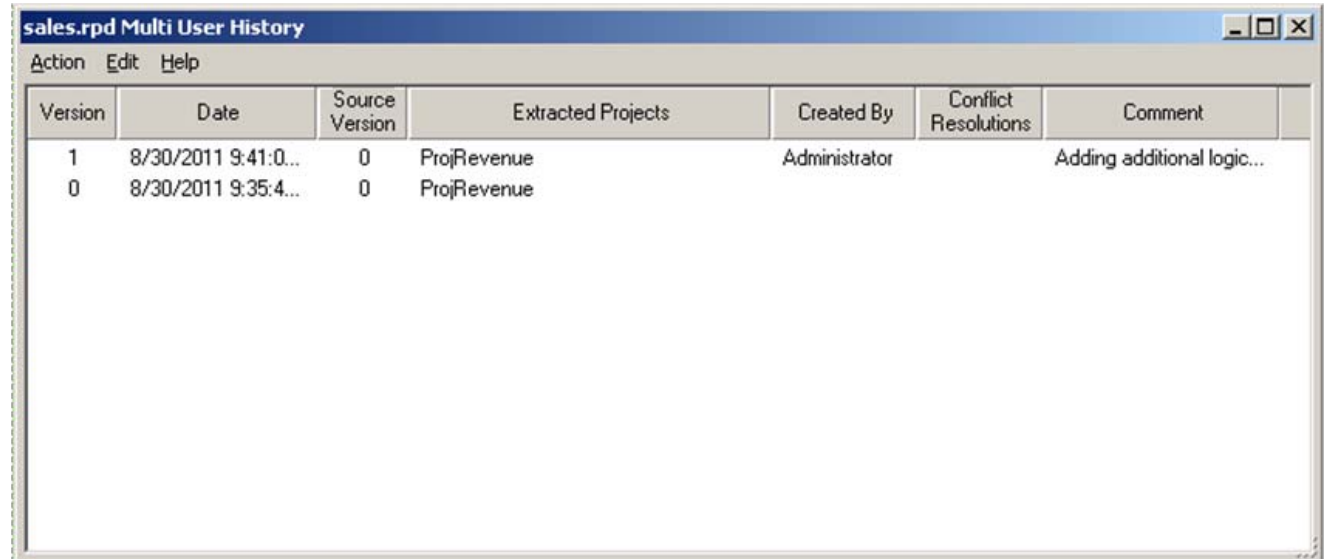
MUD Lifecycle Step 4 : Publish or Discard Changes

- After local changes have been merged into local copy of the master repository, these changes can then be merged in with the actual master repository
 - Again performed using an automatic three-way merge
- Changes can also be discarded, or rolled-back (giving you the original subset RPD again)
- At this point, the lock is taken (to stop multiple sessions trying to write to the master)
 - Only taken briefly in 11g as in most cases, conflicts dealt with in previous step
- If master RPD has been updated since local merge, local merge is rolled-back and performed again



MUD Lifecycle Step 5 : Viewing MUD History

- Developers with MUD configured on their workstations can view the MUD history
- See history of checkouts, check-ins, comments added during publish (lock) phase
- Useful history of MUD activity

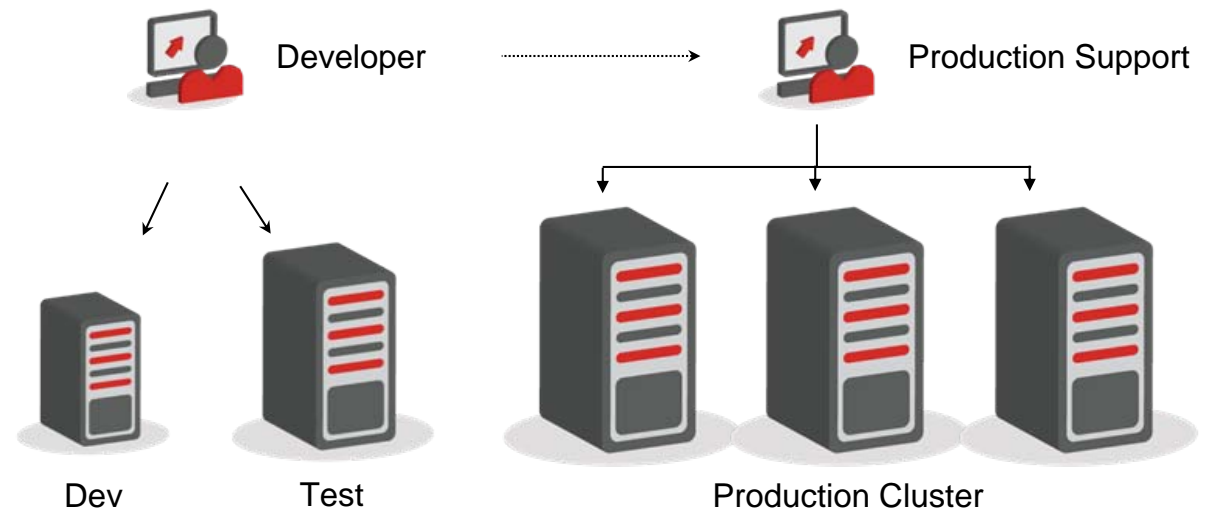


The screenshot shows a window titled "sales.rpd Multi User History" with a menu bar containing "Action", "Edit", and "Help". Below the menu bar is a table with the following columns: "Version", "Date", "Source Version", "Extracted Projects", "Created By", "Conflict Resolutions", and "Comment". The table contains two rows of data.

Version	Date	Source Version	Extracted Projects	Created By	Conflict Resolutions	Comment
1	8/30/2011 9:41:0...	0	ProjRevenue	Administrator		Adding additional logic...
0	8/30/2011 9:35:4...	0	ProjRevenue			

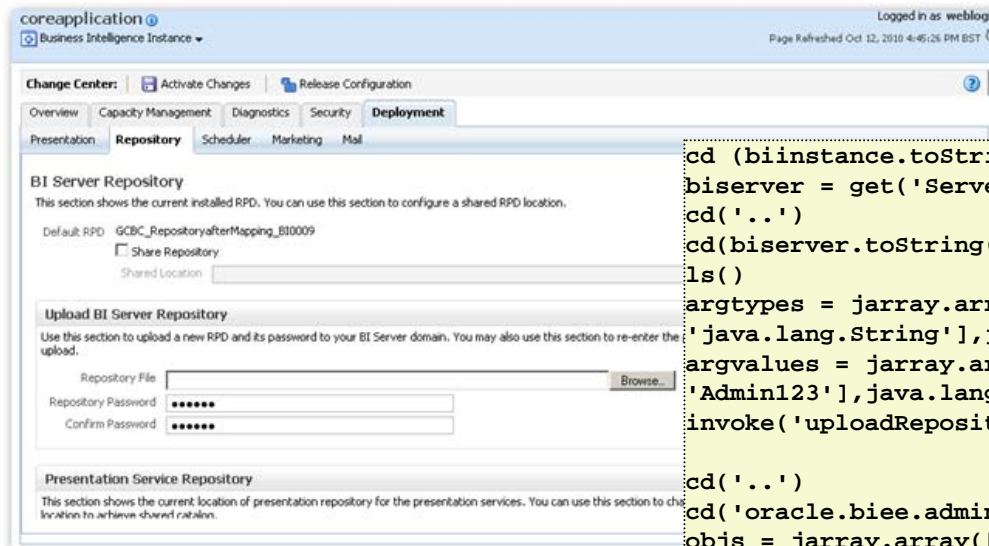
Deploying Configuration Changes across Clustered OBIEE Systems

- Another aspect of managing change and deployments is at a system level
- How do you apply configuration changes across multiple clustered nodes?
- How do you deploy repositories when your BI servers are clustered?
- How do you script the process so that it is automated?



Project Deployment and Migration Best Practices

1. Use Enterprise Manager to deploy repositories and catalogs between environments
2. Use Enterprise Manager to apply system configuration changes to environments
3. Use WLST and the Oracle BI Systems Management API to script these tasks



```
cd (biinstance.toString())
biserver = get('ServerConfiguration')
cd('..')
cd(biserver.toString())
ls()
argtypes = jarray.array(['java.lang.String',
'java.lang.String'], java.lang.String)
argvalues = jarray.array(['C:/SampleAppLite.rpd',
'Admin123'], java.lang.Object)
invoke('uploadRepository', argvalues, argtypes)

cd('..')
cd('oracle.biee.admin:type=BIDomain,group=Service')
objs = jarray.array([], java.lang.Object)
strs = jarray.array([], java.lang.String)
invoke('commit', objs, strs)
```

Managing the Oracle BI Repository and Web Catalog using EM

- Enterprise Manager is now used to deploy new RPD files (repository) and presentation catalog directories
 - RPD files are uploaded using EM; catalogs have to be manually copied to servers
- Deploys metadata across all BI Server and Presentation Server nodes in the cluster (unless shared directories have been defined)

coreapplication Business Intelligence Instance Logged in as weblogic
 Page Refreshed Aug 16, 2010 10:44:23 PM BST

Change Center

Overview | **Ca** | **Presentation**

Upload BI

Use this sectic
password if a

Presentation Service Repository

This section shows the current location of presentation repository for the presentation services. You can use this section to change the repository location and point to shared location to achieve shared catalog.

Catalog Location

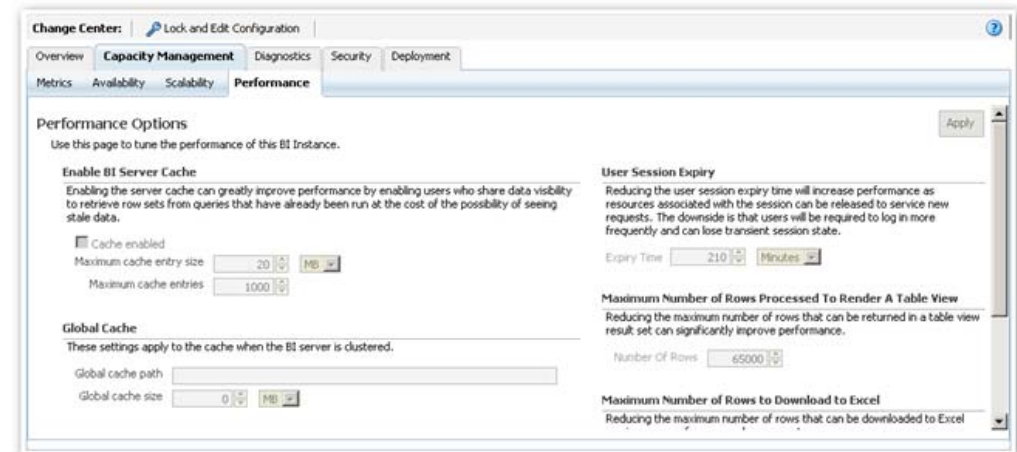
Repository File

Repository Password

Confirm Password

System Configuration Changes using Enterprise Manager

- Most important system configuration settings are now managed through EM
- Ensures that all changes you make are applied across all nodes in the cluster
- Graphical interface for managing common settings including
 - Caching and other performance settings
 - Number and scale-out of system components across cluster
 - Miscellaneous settings including # rows returned, read-only RPD etc
- Each BI environment has its own EM website, which manages all nodes in the domain



Summary

- Projects that scale beyond a single developer need deployment & change management
- Many tools are available within OBIEE 11g to handle multi-developer teams
- Keep things as simple as possible; but if required, there is MUD
- Key to MUD is understanding what goes on when you check-out/check-in projects
- 11g introduces far less intrusive locking, makes MUD more viable
- The lack of in-built version control can be overcome with tools such as Subversion
- Always use EM to propagate system changes, and if required, script with WLST.



www.rittmanmead.com

CONSULTANCY TRAINING SUPPORT

OBIEE Deployment & Change Mgmt Best Practices

Mark Rittman, Technical Director, Rittman Mead
ODTUG BI/EPM Seriously Practical Conference, Sydney 2011