An Oracle White Paper
December 2009

Metadata Services (MDS) in Fusion Middleware 11g

ORACLE

Disclaimer

The following paper 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.

Introduction

The term *metadata* is generally interpreted as "data about data" or "data about applications." In the enterprise application space, the definition has been expanded to encompass various kinds of standardized languages, such as XML Schema Definition (XSD), Web Service Definition Language (WSDL), Java Server Faces (JSF) pages, and custom and proprietary languages. More importantly, metadata is increasingly used to represent and affect application logic, resulting in applications that are metadata driven as opposed to code driven.

The logic underlying the Oracle Fusion Middleware product family is metadata driven. The structural and behavioral aspects of all Fusion Middleware components—as well as Java Enterprise applications and Service-Oriented Architecture (SOA) composites built with and deployed into the infrastructure—are declaratively described using metadata. Consider the following examples of metadata tied to various Fusion Middleware (FMW) components and applications:

- **User interaction:** Page layouts, task flows, business logic
- **Process management:** Business Process Execution Language (BPEL) process definitions, business rules, message routing rules, event definitions
- **Runtime configuration:** Configuration files, data source definitions
- **Web services:** WSDL files, XML schemas, transformations, policies, service-level agreements

By providing a unified store for metadata, Oracle Metadata Services (MDS) ensures consistent, reliable access to the metadata for FMW components and for the applications built and deployed on middleware. The benefits of a common MDS infrastructure are seen throughout the middleware. The same metadata that is used during the design phase of an application is used at application runtime through the metadata services layer. This ensures consistency through the lifecycle of the application, illustrated in the following figure. MDS also provides common administrative tooling for the metadata that can be used across various types of metadata stored in the common repository.

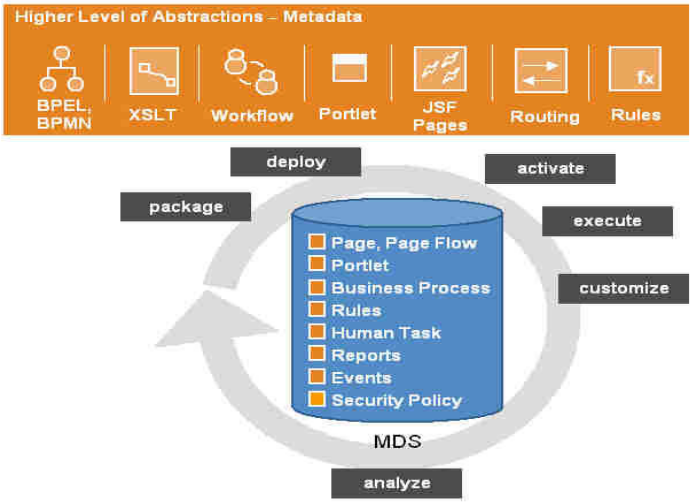

Figure 1 – MDS in Fusion Middleware

Among the biggest benefits of metadata-driven applications that use MDS are the easy and upgrade-safe mechanisms for tailoring or shaping the application logic represented by the metadata—such changes are called *customizations*. A common problem you may face with your business applications is that after you customize the applications to fit your business, the cost of maintaining the code-based customizations makes it difficult to upgrade.

In an ideal world, software systems would simply change their behavior according to their configuration. But organizations have found that such systems, while functionally sound and configurable, do not possess the inherent flexibility needed to meet all the expectations regarding customization. The underlying business challenge can be expressed this way: how can we tailor a single code base in a standardized way without compromising the link between the original and the customized versions?

MDS delivers a common customization infrastructure that is used by various Fusion Middleware components to provide ways for customers to customize application metadata. This solves the underlying business challenge by keeping customizations separate from the base application source metadata and merging them when needed. Clearly, one of the advantages of such a comprehensive customization platform is upgrade safety.

There are many artifacts that you can customize in an upgrade-safe manner. These include JSF pages, JSF fragments, task flows, data controls, and data bindings—all declarative in nature—and Application Development Framework (ADF) Business Components entity objects, view objects and applications modules. Since all ADF declarative features store data in XML format, the

changes managed by MDS are defined the same way. Tooling support in the Fusion Middleware through Oracle Composer and JDeveloper enables you to create such upgrade-safe customizations.

Architecture

Feature Overview

MDS addresses a number of design-time and runtime use cases within the Fusion Middleware infrastructure. Key features and architectural principles include:

- Simplified resource management through a single, unified repository for all artifacts used by various Fusion Middleware components. For example, UI metadata and application business and process logic are all managed in a single repository.
- Management of the metadata lifecycle for each artifact as it moves through the various stages of development, testing, staging, and production. For example, one can migrate all the metadata easily from development to the test instance.
- Sharing and reuse of metadata across projects and applications. Reuse of services across applications is a classic example of this. Impact of change across components can be analyzed through the impact-analysis features.
- Categorization and discovery of artifacts, encouraging reuse. Metadata annotations and metadata extensions are supported; this enables categorization. An example of categorization and discovery is a business dictionary that provides mechanisms for browsing and reusing metadata, such as services or reusable UI components.
- Subscriber-based change notifications when the metadata is modified or deleted.
- Versioning capabilities, which form the basis for various features, such as labeling consistent metadata, auditing, and sandboxing.

- An upgrade-safe, layered customization mechanism through which metadata and application logic can be tailored per usage of the metadata. This hugely beneficial feature is discussed more thoroughly later in this paper.

You can use MDS to manage a variety of metadata types. Validation services are available for validating metadata objects with the corresponding types. Metadata itself is in the form of XML. The grammar that defines the metadata is created using standards such as XML schemas (XSDs). You can provide further extensions using XSD annotations. You can build common types once, as XSDs, and then share them across multiple applications. For example, the types for various metadata of middleware components are defined once and then shared. You can define application-specific types through extending the platform's metamodel.

Metadata Repository

Metadata accessed and managed via MDS can be in either a file-based repository or the database-based repository. You can also configure the application to read some metadata directly from the enterprise application's archive (EAR). Several repository-specific features perform better when you use them with a database-based repository—the expected mode of usage in production environments. Also, additional features, such as versioning and sandbox support, are available when you use a database repository (more details later). MDS supports Oracle databases above version 10.2, and supports SQLServer database in Patch-Set1 release, and there are plans to support additional databases in upcoming patch-set releases.

As a true repository, MDS supports creation and management of multiple versions of metadata objects. This comprehensive versioning capability forms the basis for auditing functionality. Runtime updates of metadata objects create new versions of the affected objects, enabling you to configure the system to use both old and new versions. This is especially useful to ensure metadata consistency, even in cases of long running processes or applications. MDS versioning capability is also the basis for the granular backup and recovery management operations.

MDS provides a "sandbox" feature that enables you to make and test changes on multiple metadata objects before you commit them. You can even update changes over a period of time before you commit them. This enables you to roll them all out to your users in one action. Once committed, these sets of changes are all available in an atomic fashion to all the users, and the sandbox is cleaned up.

Performance and Caching

MDS features are designed to deliver the best performance and scalability possible for your applications. In keeping with this requirement, it is extremely critical that MDS provide a high degree of sharing of metadata objects across users and sessions. Except for metadata that will be customized or personalized by users, the rest of the metadata is typically not updated after deployment; consequently, the in-memory object structures for the metadata are also shared across users and sessions to avoid excessive object creation. One cache per enterprise application is created for all the metadata accessed by that application.

Even customized metadata is shared, so long as multiple users request the same customization layers. Advanced assembling techniques are in place to ensure that only the parts of the metadata that are customized by a layer—and not the entire metadata document—are maintained separately in the cache. This helps in both minimizing memory usage and CPU performance. MDS caching provides several configurable aspects, such as maximum cache size, that can be tuned on a running application (this is discussed later in this paper). Cached objects are also invalidated based on notifications if the corresponding metadata is modified in the database repository.

The database repository is tuned for performance both in terms of its structure and its indexes. Configurable aspects, such as periodically purging unwanted versions, are available as management operations for performance tuning. Further, the architecture is designed to support flexible database schema design, enabling the infrastructure releases to tune performance at the schema level without impacting the consuming upper layers.

Metadata Lifecycle

The key aspect of MDS and metadata-driven applications and components in Fusion Middleware is that the same metadata that is built in design time (that is, in the IDE environment) is used at runtime through the metadata services access layer. Thus the same metadata goes through the various phases of the application, including design, development, testing, deployment, customization, and migration phases. MDS plugs into the key lifecycle aspects of the middleware to ensure that metadata patching, deployment, and upgrade are all part of the same processes in the application lifecycle.

Customizations

A typical challenge endemic to the building and deploying of enterprise applications is that business users often tend to have requirements that entail application tailoring. Very often these are simple functionality changes, such as a slightly different page layout, a revision to display text, the reordering of page content, or a change in the business logic. To accommodate these types of business requirements, one can implement the changes in the development environment, and, after going through thorough testing and staging, can redeploy the application. Not only can this be a time consuming and complex process, but it may also need downtime for patching or redeployment, often compromising a company's or a customer's core business. Alternately, if you want to perform such changes to an application, you must both make the changes and maintain and port the code base across the upgrades to the application.

MDS customization support meets this challenge by enabling the customization of metadata in an upgrade-safe manner without requiring an upgrade to the base metadata. The customization "deltas" themselves are saved separate from the base. Depending on the context at runtime, the appropriate customizations are merged with the base as required. Hence, you can upgrade the base metadata without compromising the customizations. The customizations can be created and updated for the runtime to load them without any downtime of the applications.

While addressing such requirements, customization support also empowers application administrators or developers to build different customizations for different "classes" of users. For example, one can build customizations for a specific industry or an enterprise or a business role, thus adapting a software system to a specific business situation. All of these customizations still apply and can be merged with the updated base metadata that is deployed via an upgrade.

You can use multiple layers of customizations together. Such layers are merged in an order defined by the application configuration. For example, you can have both industry customizations and organization customizations. Depending on the user or the context, both customizations can take effect.

In a way, MDS customizations benefit from inheritance since each layer specifies only the changes that are required by its core business context. If a specific element is manipulated through several layers, its ultimate value is the one resulting from the layer at the top of the hierarchy. This means that a company can build a single foundational application that individuals, departments, and organizations throughout the company can customize without changing the

core application. And each division in turn can tailor the application differently for different groups of users. For example, Fusion Applications have 11 such layers of customization.

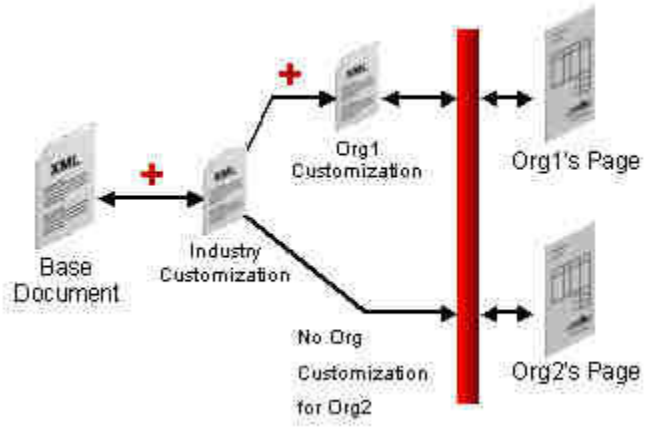

Figure 2 – Layered Customizations

You can use customizations in several different ways and create or update them in several tools:

1. You can build and ship an application with several customizations that can be applied based on the context in the deployed environment. That is, you can use customizations to build modular and reusable software. Such customizations are typically built in JDeveloper and are referred to as *seeded customizations*.
2. Consultants or system integrators can further customize an application. They can package these customizations in the updated application or ship them separately to be imported into the MDS repository directly.
3. Business administrators can customize the metadata of the application further via easy-to-use browser-based tools—also called Design-Time at Runtime (DT@RT). Oracle WebCenter Composer empowers business users to perform ADF page metadata customizations. These customizations are stored directly in the runtime MDS repository and hence the runtime loads them immediately without any application downtime or a separate import step.

4. End users can personalize the metadata—typically to adjust its look, such as reordering columns in a table, changing the order of components, or hiding components. These are also customizations in the architecture, but are at a special user-specific customization layer. ADF and Oracle Composer have built-in features to enable users to create such customizations via easy runtime gestures, such as drag-and-drop reordering of components or table columns.

The following figure depicts how these different types of customizations can be supported. In this example, verticalization and country-level customizations are packaged as seeded customizations, while organization-, site-, and user-specific customizations are done through DT@RT after deployment. Note that any metadata to be deployed to the MDS repository is shipped in a metadata archive (MAR) inside the EAR (more details on deployment later in this paper), and the MAR must include all the seeded customizations created in the IDE design time. Alternately, these customzations can be directly imported into the runtime repository via the management tooling available.

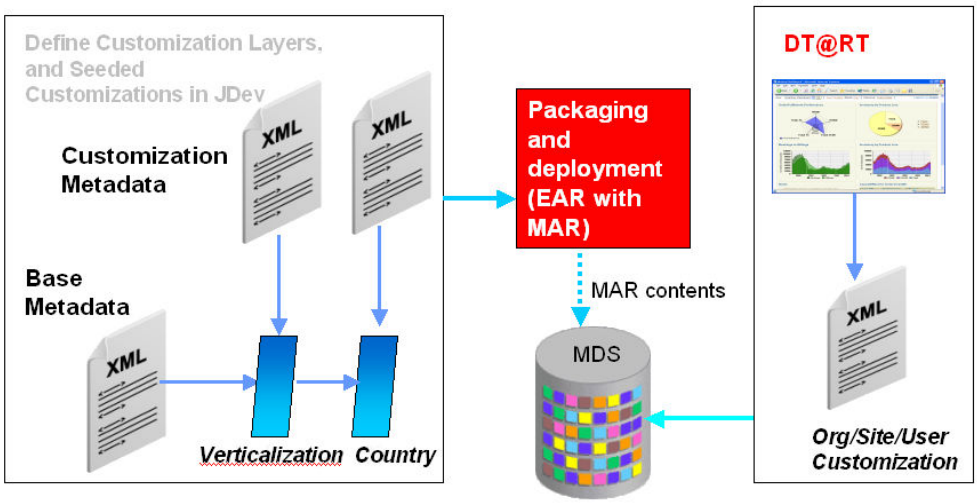

Figure 3 – Customizations and Lifecycle

How is MDS Used in Fusion Middleware 11g R1?

Metadata management services are used across various components in Fusion Middleware. Advanced features in MDS provide several benefits to Fusion Middleware components and to the applications built on these components. While in-depth discussion of how all the components of Fusion Middleware use MDS is out of the scope of this paper, we will briefly examine some key usages.

ADF and MDS

Oracle Application Development Framework (ADF) uses metadata in all model, view, and controller areas. Management and access to all of this metadata is through MDS. This metadata can be in the application's archive or deployed it into the MDS repository. You can customize this metadata using the MDS customization infrastructure. For example, you can customize the page and page flow metadata in JDeveloper at any application-configured customization layer. You can also customize ADF Business Components metadata, thus enabling the tailoring of business logic per customer requirements. ADF has built-in features to assist users in easily personalizing a running page. Personalization is a type of customization through easy user gestures, such as drag-and-drop reordering of table columns. Thus, ADF brings the benefits of customization to all users.

WebCenter and MDS

Oracle WebCenter Suite 11*g* is a unified, standards-based enterprise portal platform that provides a full range of functionality for developing the Web-based applications so necessary for a productive, efficient, and agile enterprise. The WebCenter Spaces application and the custom WebCenter applications built using WebCenter Framework use MDS to access metadata and store metadata in the MDS repository. These applications make heavy use of the customization infrastructure through Oracle Composer, a browser-based platform that enables business users

to extend and customize applications at runtime. Through Composer one can customize page metadata and business logic at any application-configured customization layer. Composer also provides a business dictionary (also referred to as the *resource catalog*), which exposes components to users, enabling them to add new components to a page. The WebCenter Framework provides capabilities to create and manage pages or ADF task flows at runtime through the Page Service. Such dynamically created pages are stored in the MDS repository and loaded by the runtime.

SOA and MDS

Oracle SOA Suite 11g supports rapid assembly of services into modular and flexible business applications. It simplifies the lifecycle of next-generation business applications by integration of BPM with SOA. This involves several metadata artifacts, including business process definitions, Web Service definitions, business rule definitions, and security policy definitions. All of this metadata is accessed through the MDS layer and is stored in the MDS repository. SOA composites use advanced features in MDS, such as versioning to store and execute metadata for various revisions of SOA composites. Reusable metadata is stored in MDS and is reused through the business dictionary (resource catalog) that is integrated into JDeveloper. You can also customize SOA metadata using the layered customizations features in MDS.

JDeveloper and MDS

The same metadata is used from design time to packaging, deployment, and runtime. So, it is not surprising that MDS services are used and exposed in JDeveloper. You can use the JDeveloper customization role to build customizations of metadata in JDeveloper. You can package these customizations with the application and deploy them to the MDS repository. You can then expose the reusable metadata in the resource catalog. Additionally, you can browse and access it from MDS and reuse it across applications, especially SOA applications.

Fusion Applications and MDS

Oracle Fusion Applications are built using the Fusion Middleware technology; thus, they use MDS extensively. In addition to the loading and storing of ADF and SOA metadata via MDS, Fusion Applications effectively use the customization features for application extensibility and tailoring. Fusion Applications have several seeded customizations of the metadata, such as product-specific customizations of components reused across products, that can be applied at

runtime based on context. These seeded customizations are maintained separate from the base metadata.

Oracle Composer is integrated into Fusion Applications to enable customers and their business administrators to tailor their application UIs in accordance with their business use cases. Eleven different customization layers are configured for all Fusion Applications. Much of the metadata is reused across ADF and SOA applications in Fusion Applications, and this is achieved through MDS.

Administration

Applications built with Fusion Middleware have several metadata artifacts that are managed through MDS. This section discusses the operational aspects of the metadata services layer and the metadata repository that administrators can configure and use. This includes deploying applications to use an MDS repository, migrating metadata from test instances to production instances, and tuning the MDS configuration used by a specific application. These management tools are available through Oracle Enterprise Manager (EM) and as WebLogic Scripting Tool (WLST) commands.

Repository creation and registration

The MDS database repository schema must be created and registered with the WebLogic Server to enable its use by deployed applications. You can use the Repository Creation Utility (RCU) to create the database-based metadata repository in an existing database. RCU creates the necessary schemas. (The *Oracle Fusion Middleware Administration Guide*, available on the Oracle Technology Network, provides details about these schemas.) You can view all registered MDS repositories in an EM screen (as shown in the figure below) and obtain them using a WLST command. You can use these tools to remove a registered MDS repository as well.

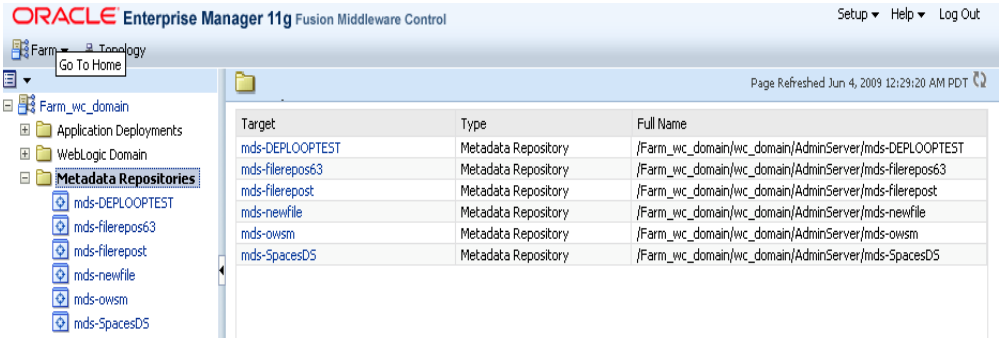

Figure 4 – Registered MDS repositories in EM

Each MDS repository can have multiple partitions. Partitions are logical separations in the tables to enable metadata from different applications to coexist in the same physical repository schema without conflicting with each other. You can create and delete partitions in a given metadata repository through EM or WLST commands.

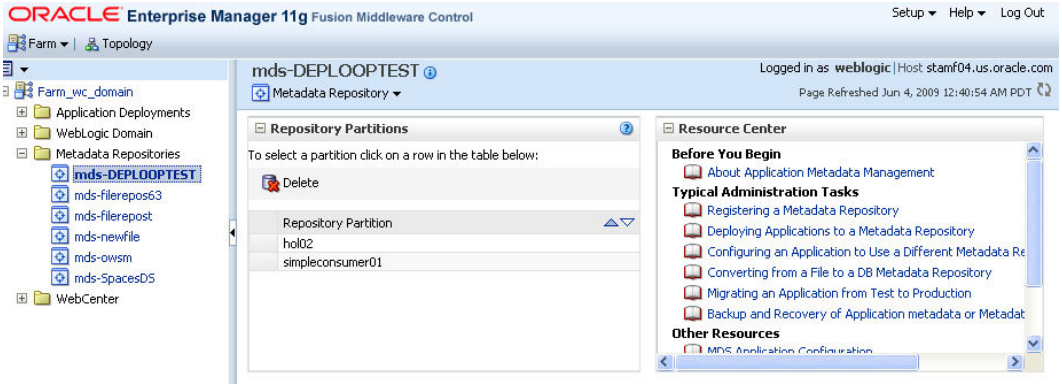

Figure 5 – MDS repository details page in EM

Deploying Applications with Metadata

When you deploy an application, you must specify the metadata repository and partition the application will use. In EM and JDeveloper, the deployment wizards prompt you to choose the

required MDS repository and partition for that application. There are WLST commands to do this as well.

While some metadata can be retained in and loaded from the application's classpath—that is, in the EAR file—there are typical cases where metadata must be in a database-based MDS repository and, thus, must be deployed to the MDS repository. Examples include applications with seeded customizations, applications enabled to create or customize metadata at runtime after deployment (such as through Oracle Composer), or SOA composite metadata. Any metadata to be deployed to MDS is shipped in a metadata archive (MAR) inside the EAR. MDS is tightly integrated with FMW deployment mechanisms and deploys the contents of the MAR to the target MDS repository that was selected at deployment.

An application can be configured to access metadata from multiple partitions/repositories, even though there is only one target repository where the MAR metadata is deployed. For example, an application can load its own private metadata from the target repository and some shared metadata from another repository. The following figure illustrates the EM deployment wizard UI for selection of the MDS repository.

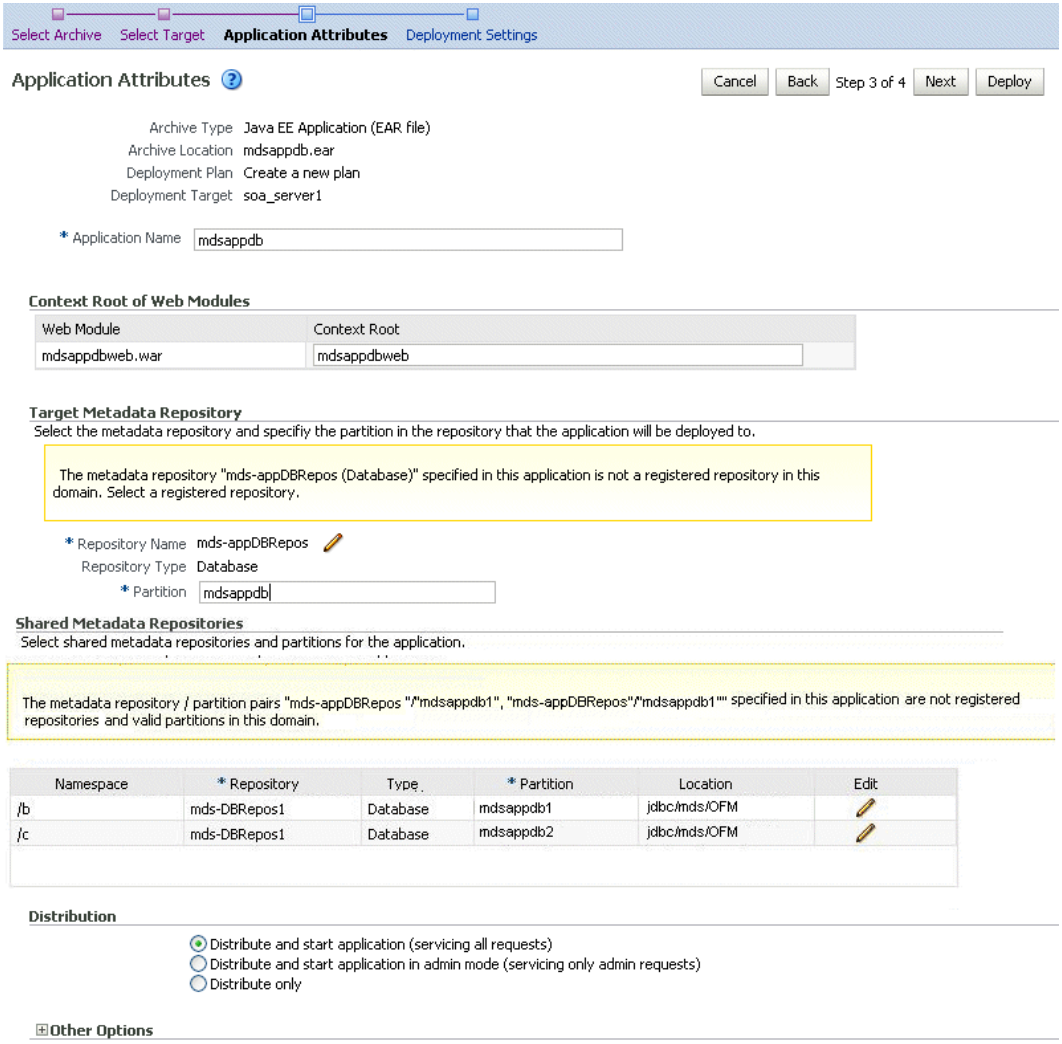

Figure 6 – Choosing MDS repository during deployment

When an application is undeployed, the metadata or metadata partition is not removed. This enables reuse of the metadata or the customizations in the partition. When you redeploy an application, you can again choose the MDS repository and partition. For example, this can be a

way to change the repository used by an application. You can delete the partition if you do not intend to reuse it.

Migrating or Deleting Metadata

Once you deploy an application, you can update or customize the metadata in its MDS repository partition. For example, you can use Oracle Composer to tailor UI pages to a particular deployment site. Often such changes must be migrated to another deployment or to a production instance. You can accomplish this through exporting and importing metadata infrastructure. Both actions are available with EM and WLST. These tools enable you to pick a specific metadata file or specific customization layers. In other words, you can use EM or WLST commands to export a granular slice of metadata.

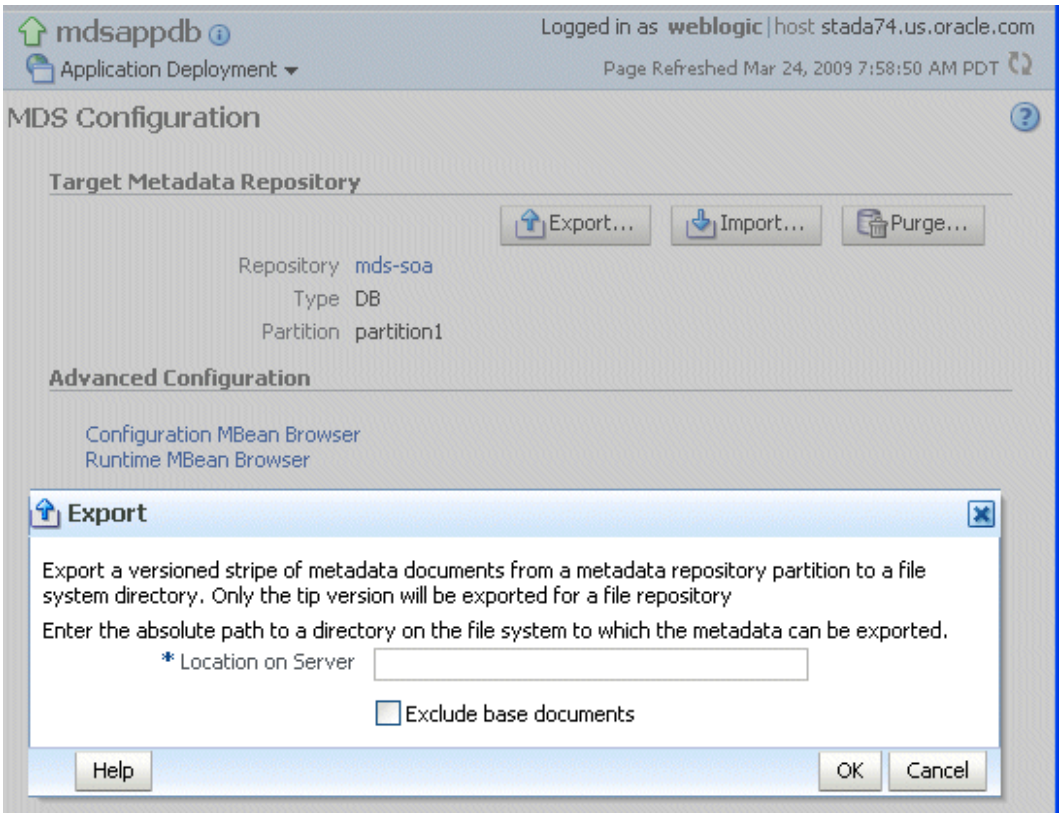

Figure 7 – Exporting metadata from EM

Tools are available for deleting specific metadata or specific customizations from an MDS repository partition. Several options are supported. For example, you can delete a specific customization layer or customizations of specific document patterns. An illustration of when this would be useful is when a customer creates customizations for a specific organization, which become unnecessary due to reorganizations. The customer can check the repository for the unnecessary customizations and delete them using metadata management tools.

If the entire database partition and its metadata must be copied to another partition, this can be accomplished through the WLST command for cloning metadata. Unlike the export command, which migrates only the chosen version of the metadata, cloning also migrates the version history.

Labels, Versions and Purging

The MDS database-based repository supports versioning. You can store multiple versions of a metadata object in the repository. Any update automatically results in creating a new version with updated contents. You can create a label to stripe all the metadata in a partition so that a consistent set of metadata can be accessed through that label. You can also access historical versions of metadata by changing the configuration of the application. Additionally, you can modify application configuration to access the metadata associated with a particular label.

Label manipulation operations, such as creating, listing, and deleting, are available as WLST commands. You can purge unlabeled, old versions of metadata from the repository using the EM or WLST purge tool.

Backup and Recovery

You can use database backup and recovery tools, such as Recovery Manager (RMAN), to back up the entire MDS database repository schema. You can also use MDS tools to back up the schema at lower levels of granularity. You can promote a label that was created earlier to remove the metadata changes that happened between that label's creation and the label promotion. You

can also use the export and import operations to back up metadata at a much more granular level.

Runtime Configuration Changes

You can configure some runtime aspects of the metadata services layer at the application level. You can change several of these configurations after the application is deployed. One example of this is the cache size. With MDS, you can set the cache size and even turn caching on or off or clear the cache at an application level. You can do this via the managed bean (MBean) browser in EM or via WLST commands. Similarly, you can configure the app to automatically invoke the purge tool periodically at specific time intervals.

Through a configuration change, you can set an application to be read only. You may want to try this when there are multiple versions of the application running, and you want to disallow updates to the metadata in an older version that is slowly being retired.

Monitoring and Loggging

You can configure the level at which MDS messages are logged so that finer-level messages are also logged for improved diagnostics. Finer-level messages can be key to diagnosing and trouble-shooting.

You can view the log using the standard log viewer functionality, available in the Enterprise Manager. Log viewing can be especially helpful during troubleshooting.

Application level metrics are available through the EM UI to enable you to monitor the health of an application with regard to its MDS access. This provides built-in support for instrumentation and diagnostics. The following figure shows some of the metrics displayed at application level in EM.

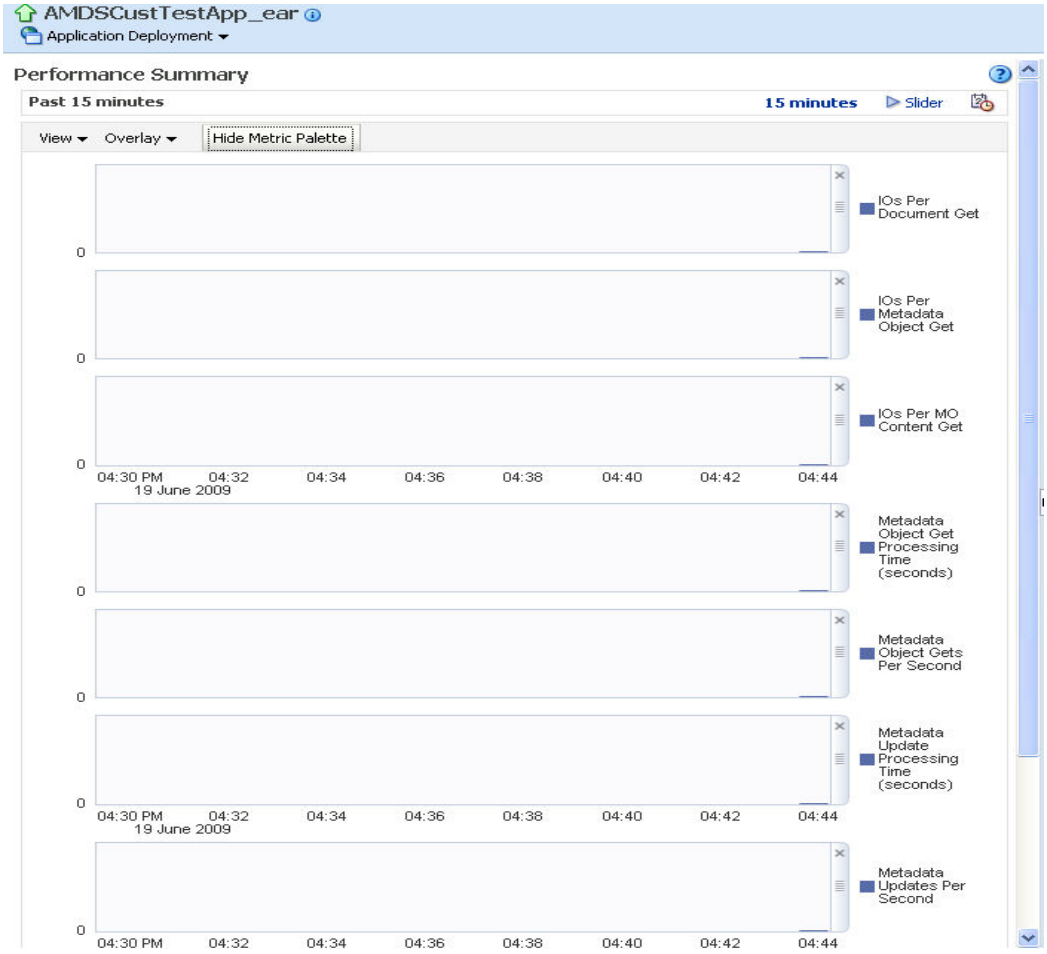

Figure 8 – MDS Metrics in EM

Conclusion

MDS is a key infrastructure component in Oracle Fusion Middleware. It is the layer through which metadata is loaded, saved, cached, stored, managed, and customized both by various middleware components and by the applications built on Fusion Middleware. MDS provides a common metadata management framework as an integral part of the middleware platform. Metadata customizations are a key feature for tailoring metadata. Several administrator tools and modifiable configuration make MDS both effective and nimble. Fusion Applications provide key features, such as reusability and upgrade-safe customizations, through the implementation of various MDS features in their architecture.

White Paper Title
December 2009
Author: Gangadhar Konduri

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

Copyright © 2009, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

0109