## **Meta Data Services Overview**

Microsoft® SQL Server™ 2000 Meta Data Services is an object-oriented repository technology that can be integrated with enterprise information systems or with applications that process meta data.

A number of Microsoft technologies use Meta Data Services as a native store for object definitions or as a platform for deploying meta data. One of the ways in which SQL Server 2000 uses Meta Data Services is to store versioned Data Transformation Services (DTS) packages. In Microsoft Visual Studio®, Meta Data Services supports the exchange of model data with other development tools.

You can use Meta Data Services for your own purposes: as a component of an integrated information system, as a native store for custom applications that process meta data, or as a storage and management service for sharing reusable models. You can also extend Meta Data Services to provide support for new tools for resale or customize it to satisfy internal tool requirements.

Meta Data Services documentation contains the following sections.

Section	Description
Meta Data Services	Describes fundamental concepts and
<u>Fundamentals</u>	strategies for using Meta Data Services.
Meta Data Services	Explains the components of Meta Data
<u>Architecture</u>	Services and how they relate to each other.
OIM in Meta Data Services	Describes the
	Open Information Model (OIM) and how it
	is used in Meta Data Services.
<u>Using Meta Data Browser</u>	Introduces Meta Data Browser and
	explains how to use it.
Programming Meta Data	Provides information about programming
Services Applications	against meta data in a repository.
Repository API Reference	Describes the classes, interfaces, and
	objects that you can use to program the
	repository engine and <u>information models</u> .

XML Encoding Reference	Describes the classes, interfaces, and
	objects that you can use to implement
	Extensible Markup Language (XML)
	encoding in your tool or application.

# See Also

What's New in Meta Data Services

#### **Meta Data Services Fundamentals**

Microsoft® SQL Server™ 2000 Meta Data Services provides a way to store and manage meta data about information systems and applications. This technology serves as a hub for data and component definitions, development and deployment models, reusable software components, and data warehousing descriptions.

Product components consist of the repository engine, tools, APIs, standard information models, a browser, and a Software Development Kit (SDK). For more information about product components, see <a href="Meta Data Services">Meta Data Services</a> Architecture.

Integrating a meta data management service into a product, a development environment, or an information system requires understanding what meta data is and how it is used. The topics in this section describe fundamental concepts and usage strategies that identify ways in which this meta data management technology is applied.

Topic	Description
Meta Data Fundamentals	Defines and explains the characteristics of
	meta data.
Information Model	Provides basic information about
<u>Fundamentals</u>	information models.
Meta Data Management	Explains why meta data management is
	important and how repository technology
	supports management tasks.
<b>Using Meta Data Services</b>	Describes ways to use Meta Data Services
	based on user roles and objectives.
Designing Meta Data Types	Explains the role of Meta Data Services in
<u>Using Information Models</u>	the design process.
<b>Developing Applications Using</b>	Explains the role of Meta Data Services in
Meta Data	the development process.
Processing Meta Data at Run	Explains the role of Meta Data Services
<u>Time</u>	during run time.

# See Also

OIM in Meta Data Services

Repository API

Repository Engine

Repository Object Architecture

<u>Using Meta Data Browser</u>

## **Meta Data Fundamentals**

Microsoft® SQL Server™ 2000 Meta Data Services is a set of services that allows you to manage meta data. Using Microsoft Meta Data Services requires understanding meta data characteristics. If you are new to the concept of meta data, this overview will help you learn about the type of data that Meta Data Services manages.

Meta data describes the structure and meaning of data, as well as the structure and meaning of applications and processes. It is important to remember that meta data is abstract, has a context, and can be used for multiple purposes in a development environment.

#### **Meta Data Is Abstract**

The simple act of describing real-world phenomena generates abstract information that qualifies as meta data. For example, describing natural phenomena such as rain, wind, and sunshine requires abstractions like the concept of weather. Weather can be further abstracted by defining concepts such as temperature, precipitation, and humidity.

In data design, real-world phenomena are also described in abstract terms. People, places, things, and numbers are grouped or designated as employee, customer, or product data.

In software design, the application and database structures that represent or store data can be abstracted into meta data classification schemes that make sense to developers and designers. A table or form is derived from an object, which, in turn, can be derived from a class.

There are multiple levels of abstraction in meta data. You can describe a data instance, then describe that description, and continue to describe subsequent descriptions until you reach some practical limit. Typically, meta data descriptions used in software development extend to two or three levels of abstraction. In real terms, a data instance of "loan table" can be described as a database table name. A database table can be described as a database table object. Finally, a database table object can be described by an abstract class that formalizes the fixed set of characteristics to which all derived objects must

conform.

#### Meta Data Has Context

The distinction between data and meta data is often called the type/instance distinction. Model designers articulate types (such as classes or relationships) and software developers articulate instances (such as a **Table** class or a *table has columns* relationship).

The distinction between instance and type is context-sensitive. What is meta data in one scenario becomes data in another scenario. For example, in a typical relational DBMS, the system catalog describes the tables and columns that contain your data. You can think of the data in the system catalog as meta data because it describes data definitions. However, with the right software tool, you can manipulate it as you would manipulate any other data. Examples of manipulating meta data include viewing data lineage or table versioning information, or identifying all tables that express financial data by searching for columns that have a currency-based data type. In this scenario, standard meta data like the system catalog becomes data that you can manipulate.

## **Meta Data Has Multiple Purposes**

You can work with meta data type and instance information just as you would with any kind of application or data design elements. Expressing design information as meta data, especially standard meta data, opens up new possibilities for reuse, sharing, and multiple tool support.

For example, defining data objects as meta data enables you to see how they are constructed and versioned. Versioning support provides a way to view, branch, or retrieve any historical version of a particular DTS package or data warehousing definition. When you develop code based on meta data, you can define a structure once and then reuse it to create multiple instances that can be versioned for specific tools and applications. You can also create new relationships among existing meta data types to support a new application design.

#### See Also

Information Model Fundamentals

Meta Data Management

OIM in Meta Data Services

#### **Information Model Fundamentals**

An information model is a set of meta data types that describe a tool, application, data structure, or information system. You can model a business process, for example, to describe the progression of an order as it moves from order entry to final invoicing. If you model a database application, your information model describes the tables and columns that are supported by the application. If your goal is to define an application for booksellers, your information model will include elements that describe books, authors, and publishers. Books, authors, and publishers are the kinds of data that a bookseller application would need to manipulate.

Notice that these examples depict types of data rather than instances of data. The first example describes an order process, not the specific orders placed by a customer. Similarly, an information model for a database application describes tables, keys, constraints, and stored procedures, but not the actual data that these elements store and manipulate. In the same way, the information model for the bookseller application describes the concept of a book, but not data about individual books. As you can see, information models articulate things that are always two steps removed from end user instance data.

## **Information Model Building Blocks**

Information models are described by classes, relationships, and properties.

A class is a template that defines the characteristics of objects. A class represents entities in an information model. In previous examples, an order, a table, a key, a constraint, and a stored procedure represent different classes.

A relationship type defines a template to which stored relationships must conform. For example, you can define the type of relationship between a table and a column, a column and a data type, or a schema and a table. A relationship type defines a set of criteria that describe how two objects relate.

A property is a template to which stored property values must conform. For example, when you store an invoice total, you must store a currency value.

## **Standard Information Models**

Microsoft® SQL Server™ 2000 Meta Data Services distributes an implementation of the Open Information Model (OIM) that provides standard meta data types that tools can use. The OIM is a generic set of information models that describe object modeling, database modeling, and component reuse.

You can extend the OIM by adding custom elements. For example, if you are creating an application that requires elements that are not included in the OIM, you can add those elements to complete your design. Although the OIM is not required, deploying an OIM-based strategy provides integration possibilities that are not otherwise achievable.

## **Importance of Information Models**

In SQL Server 2000 and in other Microsoft products that integrate with Meta Data Services, OIM-based models are predefined, installed, and operational. No action is required on your part if all you want to do is use these services. However, if you want to build your own tools and applications that work with meta data types, or if you want to create or extend an information model, you will need to know all about information models.

Information models are the key to integrating Meta Data Services with other tools and technologies. For more information about how information models fit into Meta Data Services architecture, see <u>Information Models</u>.

#### See Also

**Using Meta Data Services** 

Meta Data Fundamentals

Meta Data Management

# **Meta Data Management**

Tool developers can make use of Microsoft® SQL Server™ 2000 Meta Data Services by adding meta data management support in products they provide. Meta Data Services provides a platform for building meta data management capability into dedicated tools or into add-on features of existing tools.

Shared meta data is a way to deploy data and application structures across heterogeneous platforms and development environments. It provides common definition so that tools and applications can interpret the same meta data definition and transform it into application-specific structures. Meta data is an integration point because it is abstract, containing essential details that remain constant regardless of the implementation strategy. This flexibility makes it ideally suited for design purposes because it allows you to separate design from implementation. When you work with predefined meta data, you can implement a specific design using the development tool that best serves your needs.

As you incorporate meta data into your information systems, you need tools that can keep track of the meta data you create, how it is used, and how it can be reused in subsequent projects.

Meta Data Services helps you manage meta data by providing a platform that enables you to:

- Store meta data constructs.
- Version meta data objects and relationships so that you can work with current and historical editions with equal ease.
- Allocate workspaces to isolate modifications to a specific set of objects.
- Import and export meta data structures in Extensible Markup Language (XML) format to work with your meta data in a variety of environments.

## See Also

Using Meta Data Services

Meta Data Fundamentals

# **Using Meta Data Services**

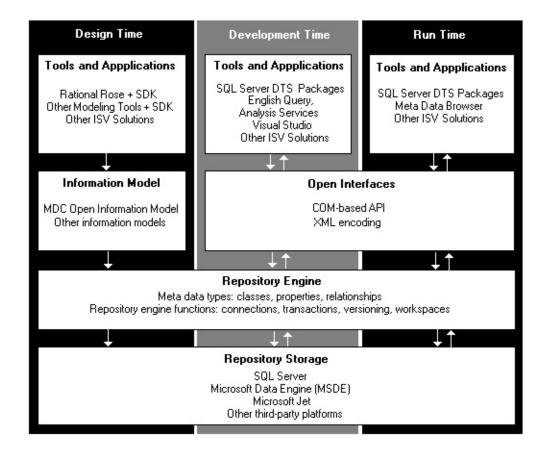
Microsoft® SQL Server™ 2000 Meta Data Services is a technology that you use with other tools. The tools that you use vary depending on whether you are designing meta data, programming with meta data, or accessing meta data at run time.

The following diagram shows the ways you can interact with Meta Data Services. It shows design time, development time, and run time phases. In practice, the distinction between these phases is not so precise. However, making these distinctions can help you understand the various ways you can work with Meta Data Services.

During design, the focus is on model creation. You can use modeling tools and the Meta Data Services Software Development Kit (SDK) to create meta data to store in a repository database.

Development begins after you have model information in the database. You can then use the API to program against it, or you can use XML encoding to exchange meta data with other repositories.

At run time, you can use browser tools that work directly with repository contents. At every level, you can create custom solutions that are based on Meta Data Services.



#### See Also

Designing Meta Data Types Using Information Models

**Developing Applications Using Meta Data** 

Meta Data Fundamentals

Meta Data Management

Meta Data Services Architecture

Processing Meta Data at Run Time

# **Designing Meta Data Types Using Information Models**

Deploying Microsoft® SQL Server™ 2000 Meta Data Services technology begins with an <u>information model</u>. Meta Data Services is intended to be used with information models that provide type information about meta data. The repository engine, repository API, add-on tools, and Software Development Kit (SDK) work with information models. The meta data types that are defined in an information model provide the design data that interacts with development tools, applications, and browsers. All Microsoft products that integrate with Meta Data Services technology base integration on some type of information model.

If you want to build an application with Meta Data Services, the information models that you use should completely describe the data, tool, or application structure that you will code later. For example, if you want to build an inventory control application, the information model that you need should completely describe the inventory control application.

If you are using the Open Information Model (OIM), your design elements are predefined. You can also use a subset of OIM elements and then supplement the model with the additional elements you require. OIM can be extended to support tool-specific meta data types or any other meta data types that your design requires.

Although the OIM provides significant advantages in terms of tool and programming support, you are not required to use it. You can create custom information models in Unified Modeling Language (UML) that are completely unrelated to the OIM.

Custom or OIM-extended information models that you create must conform to the abstract classes provided through the repository API. To build custom information models or extend an OIM model, you should use the Meta Data Services SDK. It includes a model compiler that validates your model against the repository API.

Using an information model does not eliminate the need for coding. Rather, it changes the role that coding plays. In a model-driven development environment, code provides the implementation strategy. For more information about

programming against information models, see <u>Developing Applications Using Meta Data</u>.

# **See Also**

**Using Meta Data Services** 

**Information Model Fundamentals** 

Meta Data Services SDK

OIM in Meta Data Services

Processing Meta Data at Run Time

# **Developing Applications Using Meta Data**

After an information model is installed in a repository database, you can program against it using the repository API.

Before you begin programming, it is helpful to understand how the <u>information model</u> is constructed. The information model completely describes at least a portion (if not all) of the code that you must provide. If it contains customer and order objects, your code should instantiate customer and order object instances.

Model-driven development does not place boundaries on what your application can do. As always, application code can support whatever structures and behaviors are required of it, regardless of whether they are described by a model. To be especially useful, however, your information model must contain the most complete set of meta data types that is possible. The key point to understand is that the information model provides the minimum design that you must implement in your code.

If you are using the Microsoft® SQL Server™ 2000 Meta Data Services Software Development Kit (SDK), you can speed up your development effort by incorporating samples and using the tools it provides to generate program files from your information model.

## **Development Scenarios**

For tool vendors, Meta Data Services provides a basis for integrating tools and managing tool meta data. Meta Data Services can act as an intermediate store, converting the output from one tool into the input for another tool. It also provides a way to create variants of a specific application, so that different workgroups can simultaneously pursue new application development or maintain an existing application. Meta Data Services includes functions that allow you to track this activity and then synchronize or merge the versions later.

Using meta data improves the way you develop applications. When you use meta data, you separate design from implementation. You can create a design once and then implement that design using a variety of tools. When you use meta data types, you can redirect a specific application design to different

operating systems, database, networks, and transaction processors by using repository data to drive the implementation tools.

## **See Also**

**Designing Meta Data Types Using Information Models** 

**Using Meta Data Services** 

**Information Model Fundamentals** 

Meta Data Management

Meta Data Services Architecture

Processing Meta Data at Run Time

# **Processing Meta Data at Run Time**

To support interoperability, application designers and vendors rely on processed meta data in their applications. Although it is not a common or obvious implementation of Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services, repository technology can be deployed in run-time scenarios.

SQL Server 2000 can retrieve and process meta data constructs at run time to produce cleansed data for data warehousing, queries, or Data Transformation Services (DTS) packages.

For data warehousing professionals, Meta Data Services enables tool integration and single-sourcing of data warehousing definitions used to produce cleansed data.

Meta Data Browser is another example of a tool that processes meta data at run time. The browser enables you to view existing meta data and how it is defined. For application developers, the browser provides a way to scan repository contents for objects to decide which one to use for a particular purpose.

You can create custom browsers or analysis tools for studying meta data content stored in a repository database, or create full-featured applications that manage the meta data types used by your organization.

These examples represent just some of the uses of processed meta data. You can create similar tools or new ones that use meta data in innovative ways.

#### See Also

Designing Meta Data Types Using Information Models

**Developing Applications Using Meta Data** 

**Using Meta Data Services** 

Meta Data Services Architecture

**Using Meta Data Browser** 

## What's New in Meta Data Services

Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services extends and renames the former repository component known as Microsoft Repository. Meta Data Services extends repository technology by introducing a new browser for viewing data in a <u>repository</u> database, new Extensible Markup Language (XML) interchange support, and new repository engine features.

The What's New topics contain brief overviews of the new Meta Data Services features with links to the conceptual topics that discuss each feature in more detail and provide further links into the documentation.

Topic	Description
Meta Data Browser	New in this release, Meta Data Browser is
<u>Enhancement</u>	a tool that you can use to browse a
	repository database.
XML Encoding Enhancements	New in this release, XML Encoding
	supports a new implementation of meta
	data interchange in Meta Data Coalition
	(MDC) Open Information Model (OIM)
	XML.
Repository Engine	Programming enhancements detail new and
Programming Enhancements	better ways for programming against an
	installed <u>information model</u> .
Repository Engine Modeling	Modeling enhancements support new
<u>Enhancements</u>	definitions that you can include in an
	information model.

## See Also

Meta Data Services Architecture

Meta Data Services Overview

Meta Data Services SDK

**Upgrading from Earlier Versions** 

## **Meta Data Browser Enhancement**

Meta Data Browser is a new tool that you can use to browse the contents of a repository database. This tool is introduced in Microsoft® SQL Server™ 2000 Meta Data Services. You can run Meta Data Browser when you select Meta Data Services. Meta Data Services is available for each copy of SQL Server you install.

For more information, see <u>Using Meta Data Browser</u>.

#### See Also

Meta Data Services Overview

What's New in Meta Data Services

# **XML Encoding Enhancements**

Microsoft® SQL Server™ 2000 Meta Data Services now uses Extensible Markup Language (XML) encoding in native mode. You can import, export, and publish repository meta data in a format that more closely matches your information model.

XML encoding supercedes the XML Interchange Format (XIF) that was part of previous versions of the software. For more information, see <u>XML in Meta Data Services</u> and <u>Using XML Encoding</u>.

#### **New COM Interfaces**

XML IExport Interface Overview

XML IImport Interface Overview

#### See Also

Meta Data Services Overview

**Upgrading an Information Model** 

What's New in Meta Data Services

# **Repository Engine Programming Enhancements**

Repository engine version 3.0 is backward compatible with version 2.0 functionality and interfaces. You can use version 3.0 with no change to the databases. In this case, only the version 2.0 features will work. You can upgrade the database version 2.0 to the database version 3.0 format by passing the REPOS\_CONN\_UPGRADE flag when you open the repository database. Another way to upgrade a repository database is through Meta Data Browser, during repository database registration. Upgrading allows you to use all the features of version 3.0. After you upgrade, however, you cannot open the upgraded database using a version 2.0 engine.

Microsoft® SQL Server™ 2000 Meta Data Services introduces the following new repository engine features in version 3.0. These features extend your ability to program against an information model that resides in a repository database.

View Generation

#### **New COM Interfaces**

IViewClassDef Interface

IViewInterfaceDef Interface

**IViewPropertyDef Interface** 

IViewRelationshipDef Interface

**Performance Hints** 

#### **New COM Interfaces**

IReposOptions Interface

Property Extensions for BLOBs and Large Text Fields

#### **New COM Interfaces**

IReposProperty2 Interface

IReposPropertyLarge Interface

## IRepositoryObject2 Interface

**Collection Filters** 

#### **New COM Interfaces**

**IReposQuery Interface** 

Integration with MS DTC

#### **New COM Interfaces**

IRepositoryTransaction2 Interface

Version Propagation

## **See Also**

Meta Data Services Overview

**Repository Engine Modeling Enhancements** 

**Upgrading the Repository Engine** 

What's New in Meta Data Services

# **Repository Engine Modeling Enhancements**

Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services introduces the following new repository engine features in version 3.0 of the engine. These features extend your ability to create an information model that can take advantage of new features added to this version of the repository engine.

**Scripting Support** 

#### **New COM Interfaces**

IClassDef2 Interface

**IInterfaceMember2 Interface** 

IInterfaceDef2 Interface

**IScriptDef Interface** 

**Interface Implication** 

#### **New COM Interfaces**

**IInterfaceDef2 Interface** 

**Member Delegation** 

#### **New COM Interfaces**

**IInterfaceMember2 Interface** 

**IInterfaceDef2 Interface** 

**Sharing Model Information** 

#### **New COM Interfaces**

IReposTypeLib2 Interface

Naming Semantics for Objects

Parameter Support

#### **New COM Interfaces**

**IMethodDef Interface** 

IParameterDef Interface

**Enumeration Definition** 

#### **New COM Interfaces**

IEnumerationDef Interface

IEnumerationValueDef Interface

IPropertyDef2 Interface

Type Information Aliasing

#### **New COM Interfaces**

IReposTypeInfo Interface

IReposTypeInfo2 Interface

IInterfaceMember2 Interface

Version Labeling

#### **New COM Interfaces**

IVersionAdminInfo2 Interface

Virtual Members

#### **See Also**

Meta Data Services Overview

**Repository Engine Programming Enhancements** 

<u>Upgrading the Repository Engine</u>

What's New in Meta Data Services

### **Meta Data Services Architecture**

Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services architecture is based on a set of integrated components. Key components are summarized in the following list and presented in a subsequent diagram.

### **Component Summary**

Architecture components are described in the following list. You can find out more about each component by reading specific topics.

Торіс	Description
Tools and Applications	Describes how tools and applications relate to the Meta Data Services architecture.
	Meta Data Browser is a tool provided with Meta Data Services. For more information, see Using Meta Data Browser.
Open Standards: OIM, COM, XML	Describes open standards that Meta Data Services supports, including Component Object Model (COM)-based interfaces, Extensible Markup Language (XML) encoding, and Open Information Model (OIM).
	OIM supports standard meta data. XML encoding supports import and exporting of OIM meta data. For more information, see OIM in Meta Data Services and XML in Meta Data Services.
Information Models	Describes how Meta Data Services supports standard and user-defined meta data through the OIM and other <u>information models</u> .
	Model designers and programmers can use the Software Development Kit (SDK) to create meta data-based applications and to build or

	extend information models that are the basis of shared meta data. For more information, see <a href="Meta Data Services SDK">Meta Data Services SDK</a> .
Repository Engine	Discusses the <u>repository engine</u> , which stores, consolidates, and retrieves meta data in repository databases.
	The repository engine is exposed as an object model that you can access using the repository API.
Repository API	Discusses the API, which exposes repository engine functions and information model definitions through COM interfaces.
Repository Databases	Discusses the database storage of meta data.

### **Architecture Diagram**

In Meta Data Services architecture, tools and applications connect to the core engine and storage components through open standards. Information models define type information that determines the structure and behavior of meta data that is exposed by tools and applications at the top layer.

#### See Also

Meta Data Services Fundamentals

OIM in Meta Data Services

**Specifications and Limits** 

**Using Meta Data Services** 

XML in Meta Data Services

# **Tools and Applications**

Microsoft® SQL Server™ 2000 Meta Data Services is a technology designed to be used by tools and applications. A tool is a software program intended to help application developers design, implement, deploy, and maintain applications. An application is a program designed to assist in the performance of a specific task, such as word processing, accounting, or inventory management.

In Meta Data Services architecture, tools and applications are programs that you build or provide. In the architecture, tools and applications exist outside of the core engine and storage components, connecting to these core components through open standards.

Tools and applications range from modeling software that you use to build <u>information models</u> at design time to data warehousing or application development tools that use or transform <u>meta data</u> at run time. The type of tools and applications that can be used with Meta Data Services is open-ended.

The following list summarizes some of the ways in which Microsoft, independent software vendors (ISVs), and software developers have integrated Meta Data Services with tools and product offerings. This list shows a few examples of how the Microsoft repository technology is being applied.

- SQL Server 2000, SQL Server 2000 Analysis Services, English Query, and Microsoft Visual Studio® use Meta Data Services to store meta data, to interchange meta data with other tools, and to add versioning capability to tools that support meta data creation.
- ISV providers use Meta Data Services in commercial product offerings to store value-added, predefined information models that can be used by ISV customers.
- Software developers have incorporated Meta Data Services into application development environments to deploy application designs across a variety of development tools.

For more information about how the technology is deployed by third-party

vendors, see the SQL Server page at the Microsoft Web site.

### **See Also**

**Developing Applications Using Meta Data** 

Meta Data Services Architecture

Open Standards: OIM, COM, XML

<u>Using Meta Data Browser</u>

### **Open Standards: OIM, COM, XML**

Open standards are publicly available specifications that describe the characteristics of a technology. The objective of open standards is to promote interoperability. For integration platforms like Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services, interoperability is essential. For this reason, you find open interfaces and specifications used wherever possible in Meta Data Services architecture.

Meta Data Services supports three open standards: the Meta Data Coalition (MDC) Open Information Model (OIM), Component Object Model (COM) interfaces, and Extensible Markup Language (XML) encoding. At each phase of an application life cycle, there is an open standard that you can use to lock in integration.

#### **OIM for Design-Time Integration**

OIM is a standard information model that can be shared, reused, and extended. It is published by the MDC and is widely supported in the tool vendor market. The wide support of OIM makes it possible to use the same information model design in a variety of implementation tools.

In Meta Data Services, you can use OIM as a framework on which to build type information. OIM is not required by the repository engine or the API. However, when you use OIM, you can take advantage of resources and features in the Meta Data Services Software Development Kit (SDK) that greatly simplify your development effort. Furthermore, you can use information models in any OIM-compliant tool.

#### **COM Interfaces for Program-Level Integration**

COM interfaces are binary specifications for building, using, and evolving component software. COM is supported in programming languages such as Microsoft Visual Basic®, Microsoft Visual C++®, and Microsoft Visual J++®. COM architecture and supporting infrastructure are developed and maintained by Microsoft.

When programming Meta Data Services, you can use COM interfaces defined in

the repository API to access both the repository engine and your information model from your application code. Because the repository engine and information models are exposed as COM objects, the only thing that varies from one programming language to another is the COM-implementation strategy of your development platform.

#### XML for Run-Time and Storage-Level Integration

XML is a World Wide Web Consortium (W3C) standard for the representation of information as structured documents. XML is used increasingly for data transport between heterogeneous systems.

The <u>repository engine</u> supports XML encoding to provide import and export of stored meta data in XML format. XML encoding enables the exchange of meta data between different OIM-compliant repositories.

#### See Also

Meta Data Services Architecture

OIM in Meta Data Services

**Repository API** 

**Using XML Encoding** 

XML in Meta Data Services

#### **Information Models**

Information models define meta data types that are stored in a repository database and used by tools and applications. Information models used with Microsoft® SQL Server™ 2000 Meta Data Services must be described using Unified Modeling Language (UML).

Meta Data Services distributes a set of standard information models called the Open Information Model (OIM). SQL Server 2000 preinstalls into the **msdb** database the standard OIM subject areas that describe Data Transformation Services (DTS) packages, data warehousing definitions, and online analytical processing (OLAP) cubes. These information models require no modification to perform the functions for which they are intended.

You can use these same standard models as a framework for building new applications. You can also create new models to work with using the Meta Data Services Software Development Kit (SDK). As long as your information models are defined in UML, you can use them in Meta Data Services.

Although you can create information models programmatically, most information models are created in modeling tools like Rational Rose. Custom information models must conform to the repository API. The repository API includes abstract classes that formally describe the elements you can include in a model. If you are creating a custom model, you may want to review the repository API for more information about the type information objects that the repository engine supports.

After you define and test an information model, you can install it in a repository database. Model installation creates the storage structure for your meta data. Tools and applications that use the model can populate the storage with instance data about the model definitions.

At run time, the repository engine reads the meta data and instantiates Component Object Model (COM) objects in a tool or application that correspond to the objects, relationships, and members of your information model. The COM interfaces that you use are derived from an information model installed in a repository database. In this way, the information model is a blueprint for the COM objects that the repository engine exposes.

You can also use Extensible Markup Language (XML) to import and export meta data between platforms, tools, and applications. For more information, see <u>Using XML Encoding</u>.

#### **See Also**

**Creating and Extending Type Information** 

Designing Meta Data Types Using Information Models

**Information Model Fundamentals** 

Meta Data Services SDK

OIM in Meta Data Services

# **Repository Engine**

The repository engine is a service that provides basic functions for storing and retrieving objects and maintaining the relationships among them.

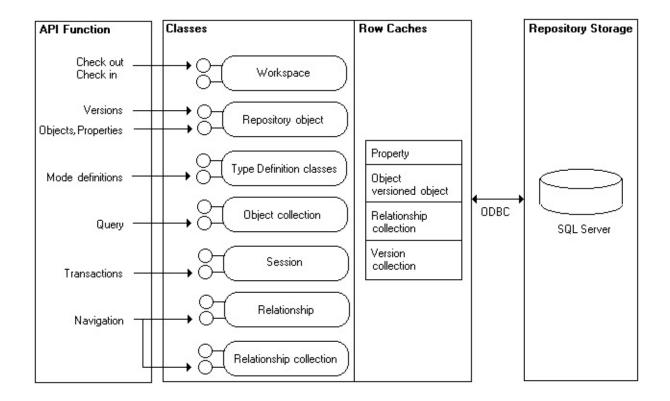
The engine performs these functions within the context of an <u>information model</u>. In this way, it is a model-driven interpreter. The engine processes user-defined model information to determine how to store and support objects, relationships, and actions. When you use the repository engine to manipulate instances of information models, the engine does so only to the extent that model structure allows. For example, the engine will establish an object relationship only if the underlying model supports it.

To use the repository engine, you program against the repository API. The repository engine responds to directives issued by application code through Component Object Model (COM) and COM Automation interfaces.

The repository engine handles all interaction with the storage layer for you. Although you control when and how transactions occur, the engine generates the SQL commands that execute specific actions.

#### **Engine architecture**

The following diagram illustrates engine architecture. The engine executes as a class library. It buffers instance data from repository databases in a cache. Objects that you manipulate in code point to the cached data. Row caching maintains state information about the objects, properties, and relationships that are instantiated.



#### **See Also**

Meta Data Services Architecture

**Repository API** 

# **Repository API**

The repository API is a programming interface that is used to drive the repository engine from within application code.

The API is based on an object model that describes repository engine functionality and type definitions that correspond to information models. Examples of engine functionality include transaction, workspace management, and connection services. Examples of type definitions include class, interface, property, and relationship definitions. In practice, the distinction between the two parts of the object model is artificial. Depending on the requirements of your application code, you will invoke objects of either type whenever and wherever you need to.

The API is exposed through Component Object Model (COM) and COM Automation interfaces, supporting an open standard for application development so that you can program using any COM-compliant programming language.

By using the Meta Data Services Software Development Kit (SDK), you can build models that conform to the type definitions supported by the API. After you build and install a model, you can instantiate objects and invoke interfaces through application code. Instantiated objects support interfaces that have single-valued properties and collections of relationships.

#### See Also

Meta Data Services Architecture

Meta Data Services SDK

Open Standards: OIM, COM, XML

**Programming Meta Data Services Applications** 

Repository API Reference

# **Repository Databases**

A repository database stores physical data. Such data includes repository type libraries that contain type information or object instance data, and tables used to map or otherwise manage object relationships. In the versions of Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services that Microsoft distributes, storage is provided through SQL Server, SQL Server Runtime Engine, or Microsoft Jet.

Meta Data Services is an installed component of SQL Server. If you are using SQL Server, a repository database already exists for your use. By default, repository tables are predefined in the **msdb** system database. These tables store data warehousing meta data used by SQL Server and other add-on components.

At a minimum, a repository database includes standard tables that are present in every repository database. Additional tables are created for custom interface definitions. Within its tables, the repository engine stores properties and relationships. For more information about repository tables, see <a href="Repository SQL Schema">Repository SQL Schema</a>.

If you prefer, you can choose to add your custom meta data to **msdb**. By keeping all your meta data in one database, you can combine existing definitions in new ways by creating relationships.

#### **Managing Repository Databases**

Repository databases are driven by the <u>repository engine</u>, which manages all transactions and determines storage structure. To save space in the database, Meta Data Services can sometimes eliminate redundant data definitions. For example, it may store a single copy of a property value, even if that property value describes many object versions. Similarly, Meta Data Services can sometimes store a single copy of a relationship, even if many different object versions have that relationship.

Repository databases should not be modified directly. Unless you are an expert database programmer or administrator, avoid modifications because you can introduce changes that the repository engine cannot manage.

You can store multiple information models in a single repository database.

Connection to a repository database is made using ODBC drivers. To access a specific information model in the tables, use the repository API.

#### Other Database Types

If you are using Meta Data Services as an add-on component of SQL Server, physical storage of meta data is implemented as a SQL Server database. However, Meta Data Services is also distributed with other Microsoft products. If you are using Meta Data Services as an add-on component of Microsoft Visual Studio®, you can implement data storage as SQL Server tables using Microsoft Jet or SQL Server Runtime Engine is a SQL Server compatible data engine and it can be used to provide local data storage. For more information about Microsoft Jet or the SQL Server Runtime Engine, see the MSDN® Library at the Microsoft Web site.

Other third-party vendors support Meta Data Services on non-Microsoft database platforms. Future development by third-party vendors will expand the number of database platforms that you can use with Meta Data Services. For more information about third-party support, see Meta Data Services on the SQL Server page at the Microsoft Web site.

#### See Also

Connecting to and Configuring a Repository

Meta Data Services Architecture

Repository API

**Repository Engine** 

Storage Strategy in a Repository Database

#### **Meta Data Services SDK**

The Microsoft® SQL Server™ 2000 Meta Data Services Software Development Kit (SDK) contains resources for model designers and programmers. It includes the Modeling Development Kit (MDK), modeling documentation, sample code, and add-on tools. Together, the SDK and MDK provide essential resources for extending models, customizing models, building new models, and programming against models.

The following components are distributed in the SDK.

- The MDK includes documentation, programming extensions, and resource files that you can use to validate models and generate programming resources for Microsoft Visual Basic® and Microsoft Visual C++® programs.
- The Open Information Model (OIM) contains models organized by subject area. Each subject area model is distributed as a separate file. Associated with each model file is a set of ready-to-use modeling files that can help you get started.
- The Model Installer automates the process of adding information models to a repository database. The Model Compiler prepares models for installation by compiling them into Repository Distributable Model (RDM) files.
- Development samples provide sample files and documentation that explains how to use the files. You can practice working with Meta Data Services using development samples. Working with sample files will help you develop the skills you need to build and work with your own files.

You can download the SDK from the Meta Data Services Web site. You must already have Meta Data Services installed in order to use the SDK. For more information, see the SQL Server page at the Microsoft Web site.

# See Also

**Installing Information Models** 

Meta Data Services Architecture

OIM in Meta Data Services

Using OLE DB Scanner

**Using XML Encoding** 

# **Specifications and Limits**

Memory and storage limits depend on the amount of RAM on your server and the disk storage resources available to the DBMS providing database storage services.

You can fine-tune repository engine performance by following the performance hints provided in this documentation. For more information, see <a href="Optimizing Repository Performance">Optimizing Repository Performance</a>.

# **Storage Limits**

Microsoft® SQL Server™ 2000 Meta Data Services uses storage provided by a DBMS. SQL Server 2000 imposes no practical limit on database size. If you are using **msdb**, or if you are creating a new repository using SQL Server 2000, you can configure database size when you require more storage.

#### **Memory Limits at Run Time**

The repository engine works with available RAM to process transactions and instantiate model information. The more RAM you have available, the better the repository engine performs.

#### See Also

Meta Data Services Architecture

**Programming Environment** 

**Repository Databases** 

#### **OIM in Meta Data Services**

The Open Information Model (OIM) is a set of standard object models that tools and applications use to create exchangeable <u>meta data</u>. The OIM is published by the Meta Data Coalition (MDC).

Microsoft distributes a version of the OIM with Microsoft® SQL Server™ 2000, the SQL Server 2000 Meta Data Services Software Development Kit (SDK), and Microsoft Visual Studio®. Meta data models based on the OIM are used to define meta data in these and other Microsoft offerings. To accommodate tool-specific meta data, Microsoft has extended the version of the OIM that it distributes to support its meta data requirements.

The OIM is an evolving standard. You can always obtain the latest version of the OIM from the MDC to support your model-based application development. You can also extend it by adding new definitions to support your tool-specific meta data requirements.

This section includes topics that explain why the OIM is important, where you can get additional information and resources, and who backs OIM development.

Topic	Description
Why the OIM is Important	Explains why Microsoft integrates the
	OIM with Meta Data Services.
OIM Resources and	Tells you where to obtain OIM
<u>Documentation</u>	documentation and model development
	resources.
Meta Data Coalition	Introduces the Meta Data Coalition (MDC).

#### See Also

<u>Information Models</u>

**Information Model Fundamentals** 

Open Standards: OIM, COM, XML

# XML in Meta Data Services

# Why the OIM is Important

The Open Information Model (OIM) is a formal specification of <u>meta data</u> that provides common ground for defining standard meta data. To achieve maximum integration across its product lines, Microsoft uses the OIM standard when defining meta data constructs. To promote the use of standard meta data for model-driven application development, Microsoft distributes the OIM with the Microsoft® SQL Server™ 2000 Meta Data Services Software Development Kit (SDK), with SQL Server 2000, and with Microsoft Visual Studio®.

OIM definitions are widely supported by third-party data warehousing, enterprise application, and knowledge management tool vendors. By building OIM-compliant models, you introduce a level of standardization that provides recurring benefits by later allowing you to choose from a variety of development tools that support OIM.

The OIM is an established specification based on the collective experience of many vendors and developers. An important advantage to using the OIM is that it is a stable model framework. Modeling is not trivial. If you are unfamiliar with modeling, the OIM can help you get started by providing a general architecture within which you can organize your development effort.

The OIM is extensible and evolving. If existing definitions do not meet your needs, you can create new core information models that satisfy your requirements exactly. In the same way, as tool and data warehousing vendors identify new requirements for meta data definition, the OIM can be extended in a way that supports a common implementation.

#### See Also

Meta Data Coalition

Meta Data Fundamentals

OIM Resources and Documentation

Open Standards: OIM, COM, XML

# **OIM Resources and Documentation**

This topic identifies important resources and documentation that can help you get started with the Open Information Model (OIM).

- The OIM is a formal specification that is extensively documented. OIM documentation is published by the Meta Data Coalition (MDC) and can be downloaded from the MDC Web site. It is also distributed with the Microsoft® SQL Server™ 2000 Meta Data Services Software Development Kit (SDK). For more information about the OIM, see <a href="http://www.mdcinfo.com">http://www.mdcinfo.com</a>.
- The Meta Data Services SDK includes a version of the OIM, modeling documentation, and several resources to help you use and deploy the OIM right away. OIM resources include definition files for Microsoft Visual C++® and Microsoft Visual Basic®, Extensible Markup Language (XML) files, Interface Definition Language (IDL) files, and installation scripts for OIM models. For more information about the SDK, see Meta Data Services SDK.

#### See Also

Developing Applications Using Meta Data

Meta Data Coalition

## **Meta Data Coalition**

The Meta Data Coalition (MDC) is an independent organization of vendors and users who volunteer time and resources to pursue agreed-upon goals related to the standardization of enterprise <a href="meta-data">meta-data</a> for the mutual benefit of all interested parties.

The MDC drives the definition, implementation, and evolution of the Open Information Model (OIM) and its support mechanisms. Proposals for new OIM models and extensions are made available to members for in-depth review. From this process, agreed-upon models are formally adopted into the OIM specification and published through the MDC Web site.

The MDC maintains a Web site to disseminate information and an e-mail address to allow members and potential members to communicate electronically. For more information about the MDC, see the MDC Web site at <a href="http://www.mdcinfo.com">http://www.mdcinfo.com</a>.

## XML in Meta Data Services

Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services supports Extensible Markup Language (XML) Encoding of <u>information models</u> for the purpose of importing, exporting, and publishing <u>meta data</u> in XML. You can exchange meta data between two <u>repository</u> databases, between a repository database and an application, or between two applications that can interpret the same XML format.

Meta Data Services encodes, exchanges, and decodes XML documents for you. This functionality is provided through dual interfaces so that you can manage these operations from code.

The XML format supported by Meta Data Services is defined by the Meta Data Coalition (MDC) <u>Open Information Model (OIM)</u> XML Encoding format. This format defines rules for generating XML that is based on an information model. Applying these rules enables Meta Data Services to generate XML that corresponds to your information model. These same rules also enable Meta Data Services to convert an XML document back into repository instance data.

XML Encoding provided with this release of Meta Data Services supersedes the XML Interchange Format (XIF) that was part of previous versions of the software. For more information about backward compatibility, see <u>Using XML Encoding</u>. For more information about how to use XML Encoding, see <u>Ways to Use XML in Meta Data Services</u>.

## **About MDC OIM XML Encoding**

Both the OIM and the MDC OIM XML Encoding format are defined by the MDC. To make best use of the XML Encoding functionality, your meta data should conform to the most recent version of the OIM. The MDC OIM XML Encoding format is optimized for the most recent version of the OIM. You can generate richer, more accurate XML if your information model is based on the version of OIM that best matches the XML Encoding format.

You can generate valid and well-formed XML for any information model, however, even if it is not based on OIM. If the information model is not based on the OIM, the MDC OIM XML Encoding rules still determine which XML tag

elements are used to structure your repository data. To see which XML elements will be created for your information model, you can use the Meta Data Services Model Development Kit (MDK) to generate an XML Document Type Definition (DTD). XML DTDs are definitions of the structure that an XML document can assume.

#### See Also

Meta Data Coalition

OIM in Meta Data Services

Open Standards: OIM, COM, XML

## Ways to Use XML in Meta Data Services

Extensible Markup Language (XML) support for Open Information Model (OIM)-based meta data opens up new possibilities for publishing and sharing meta data. For example, you can build an application that creates XML and then let the repository engine manage it. You can also exchange meta data with other repositories and with other tools that use meta data. If you have two applications that understand the same XML format, you can exchange meta data between the two applications directly, without interacting with a repository database or the repository engine.

In Microsoft® SQL Server™ 2000 Meta Data Services, you can use XML Encoding to achieve the following benefits. You can compare each benefit to the diagram to see how XML is used between repositories and applications.

- You can export and import meta data between two repository databases.
   The diagram shows that you can exchange meta data between
   Repository A and Repository B through XML documents.
- You can export meta data from a repository database to a tool or runtime object. In the diagram, you can provide data from Repository A to Application A through an XML document.
- You can import meta data from a tool or run-time object to a repository database. In the diagram, you can import meta data from Application A to Repository A through an XML document.
- XML Encoding supports a fourth benefit outside the scope of repository technology. OIM-enabled tools that support the same OIM models can exchange meta data directly, without the support of an underlying repository database. Although the diagram does not indicate this, you can exchange data between Application A and Application B (for example) using XML documents.

The following diagram shows the relationship and flow of XML from one

repository to another, and subsequently to other applications.

# See Also

Open Standards: OIM, COM, XML

**Using XML Encoding** 

# **Upgrading from Earlier Versions**

A Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services installation consists of certain core components, each of which can vary by version or format. A single configuration includes the following: a DBMS, a <u>repository engine</u>, a <u>repository database</u>, one or more <u>information models</u>, and an Extensible Markup Language (XML) interchange format.

- The DBMS used to manage the repository database can be Microsoft Jet 3.5 or later, SQL Server 6.5 or 7.0, or SQL Server 2000.
- The repository engine can be version 2.0 or 3.0.
- The repository database can be a 2.0 database or a 3.0 database (created with repository engine 2.0 or 3.0, respectively).
- Information models can be Microsoft Open Information Model (OIM) 1.0, Microsoft OIM 1.1, Meta Data Coalition (MDC) OIM, or a custom information model you define.
- XML interchange support can be XML Interchange Format (XIF) or the MDC XML Encoding format.

## **Recommended Configuration**

We recommend that you upgrade to repository engine 3.0 on each computer, and that you upgrade your repository databases to the 3.0 format. In addition, you will get better performance and have access to more repository engine features if you migrate your repository database to SQL Server 2000.

Upgrading an information model provides access to more meta data types (for example, access to Unified Modeling Language (UML) 1.3 elements).

Whether you should upgrade your information models depends on the requirements of your tools and applications. For example, if your tools and

applications are using Data Transformation Services (DTS) or Microsoft Visual Component Manager, it is not necessary to upgrade your information model. SQL Server 2000 and Visual Component Manager use the existing OIM as it is currently implemented.

Before upgrading an information model, you should determine whether a new model format will provide you with the definitions required for your repository applications. If a new model format does not offer compelling advantages, you should retain your existing information models for use with a 3.0 repository engine and databases.

For more information about upgrading any element of your Meta Data Services installation, see <u>Retaining Legacy Components in a Repository</u>.

#### See Also

**Information Models** 

**Repository Engine** 

**Repository Databases** 

What's New in Meta Data Services

# **Retaining Legacy Components in a Repository**

While upgrading every component of an installation has its advantages, in practice many installations retain at least one component that is not the latest version.

The following table describes how different versions of these components work in combination and how to upgrade to another version.

Topic	Description
Upgrading the Repository	Explains how to upgrade the repository
Engine	engine.
Upgrading and Migrating a	Explains how to upgrade a repository
Repository Database	database and migrate it to a different
	DBMS format, and how to use a
	nonupgraded database with an upgraded
	repository engine.
Upgrading an Information	Explains how to upgrade an information
<u>Model</u>	model using Extensible Markup
	Language (XML).
Using Repository Engine	Describes the availability of some
Features with Older Databases	features depending on the kind of DBMS
	and operating system you are using.

## See Also

**Information Models** 

**Repository Engine** 

**Repository Databases** 

What's New in Meta Data Services

# **Upgrading the Repository Engine**

Repository engine 3.0 is the newest version of the <u>repository engine</u>. It supports new features that improve your ability to fully define <u>information models</u> and program against a Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services <u>repository</u>.

Only one version of the repository engine can be installed on each computer. You can upgrade to version 3.0 to use the new features, or you can continue to use the version you already have installed. If you keep the previous version, you cannot upgrade a repository database. Use the newest version of the Meta Data Coalition (MDC) Open Information Model (OIM), or use the newest version of Extensible Markup Language (XML) interchange functionality.

The following table lists the versions of database, information models, and XML interchange formats that can be used with each repository engine.

<b>Engine version</b>	<b>Database version</b>	OIM version	XML version
2.0	2.0	OIM 1.0	XML
			Interchange
			Format (XIF)
3.0	2.0 and 3.0	OIM 1.0 and MDC	XIF and MDC
		OIM	XML

## **How to Upgrade the Repository Engine**

SQL Server 2000 uses repository engine 3.0 . As a result, upgrading to repository engine 3.0 is accomplished when you upgrade an existing SQL Server or create a new SQL Server 2000 installation.

To take advantage of new repository engine features, we recommend that you upgrade the repository database to the latest format. Upgrading adds tables and columns that support new features. For more information, see <u>Upgrading and Migrating a Repository Database</u>.

## See Also

**Repository Engine Modeling Enhancements** 

**Repository Engine Programming Enhancements** 

Retaining Legacy Components in a Repository

**Upgrading an Information Model** 

**Upgrading from Earlier Versions** 

**Using Repository Engine Features with Older Databases** 

# **Upgrading and Migrating a Repository Database**

Upgrading a repository database updates the repository schema with new tables and columns that support repository engine 3.0 features. If you have upgraded to repository engine 3.0, you should upgrade your database so that it corresponds to the engine.

Upgrading and migrating a database are separate, optional tasks. For any repository database, you can do all or none of the following:

- Upgrade the repository schema to the repository engine 3.0 format.
  - After you install repository engine 3.0, you can choose whether to upgrade all or some of your repository databases. After you upgrade, however, you cannot open the database using a version 2.0 engine. Upgrading is unrelated to database migration. You can upgrade a repository database that you created in Microsoft® SQL Server<sup>TM</sup> version 6.5, for example, without having to migrate your SQL Server 6.5 database to a later version of SQL Server.
- Migrate a repository database so that it runs on a more recent or different DBMS.

You can migrate a database if you require performance improvements or the view generation features that are only available on SQL Server 2000.

## **How to Upgrade a Repository Database**

You can upgrade a repository database by passing the REPOS\_CONN\_UPGRADE flag when you open the repository database using repository engine 3.0. For more information about REPOS\_CONN\_UPGRADE, see ConnectionFlags Enumeration.

If you are upgrading a SQL Server 6.5 or 7.0 database, you can either use the REPOS\_CONN\_UPGRADE flag or Meta Data Browser. To upgrade the database through the browser, you must edit the properties of a registered repository database. For more information, see <u>Working with Repository</u>

#### Databases in Meta Data Browser.

## **Backward Compatibility with 2.0 Repository Databases**

The version of the repository engine that you use to initially populate the database determines the version of the repository database. For example, if you used repository engine 2.0 to populate the repository database, the repository database is version 2.0.

Repository engine 3.0 is backward compatible with version 2.0 functionality and interfaces. When you use a 2.0 repository database, typically only the 2.0 engine features of repository engine 3.0 will work. However, two repository engine 3.0 features, **IReposProperty2** and handling of <u>bit</u> properties, are available to 2.0 repository databases.

Most new features are provided through new interfaces. If you inadvertently invoke a repository engine 3.0 feature on 2.0 repository database, **QueryInterface** returns E\_NOTIMPL for the interface that provides the new feature.

For more information about feature restrictions for an upgraded database, see <u>Using Repository Engine Features with Older Databases</u>.

## **How to Migrate a Repository Database**

You can migrate a repository database from one DBMS version to another. Your DBMS provides these features.

To convert from a Jet database to a SQL Server database, you must copy the data in the Jet database to the SQL Server database using features provided by SQL Server and Microsoft Access. Next, you must set REPOS\_CONN\_RECOMPUTE on the **Open** command to add definitions that SQL Server 2000 Meta Data Services requires for SQL Server databases. For more information, see <u>ConnectionFlags Enumeration</u>.

#### See Also

Connecting to a SQL Server Repository Database
Generating Views

Repository SQL Schema
Retaining Legacy Components in a Repository
Upgrading the Repository Engine

# **Upgrading an Information Model**

This topic provides information about using different versions of information models and Extensible Markup Language (XML) interchange formats. Generally, you can omit upgrading (that is, replacing) an existing information model with a newer format of the same information model if you do not require the additional meta data types that the newer format provides.

If you do choose to replace an older information model format with a newer format, you must copy your object instance data to the new information model. XML interchange provides the means to move your data.

## **Open Information Model Formats**

The Open Information Model (OIM) is a generic information model. OIM 1.0 and the Meta Data Coalition (MDC) OIM are two alternate modeling formats of the OIM. MDC OIM is an enhanced version of OIM 1.0 that includes Unified Modeling Language (UML) 1.3 support. OIM 1.0 and the MDC OIM use the same repository tables and cannot be installed into the same repository database. If you want to use the MDC OIM, you must copy your OIM 1.0 meta data to the new information model.

- OIM 1.0 is supported by repository engine 2.0 and repository engine 3.0. A version of OIM 1.0 is distributed with Microsoft® SQL Server<sup>TM</sup>.
- MDC OIM is supported by repository engine 3.0, and it is the newer of the two formats. A version of MDC OIM is distributed with the SQL Server 2000 Meta Data Services Software Development Kit (SDK).

You can exchange and migrate meta data between these two model formats using either XML interchange format. The following section discusses backward compatibility in more detail. For more information about XML Interchange Format (XIF) and backward compatibility, see <u>Using XML Encoding</u>.

## **XML Interchange Formats**

XML interchange formats define the way in which you can exchange meta data

in XML with other tools and repositories. You can choose between two alternate XML interchange formats.

- XIF is supported by repository engine 2.0 and repository engine 3.0. It expresses meta data in entity-normal format.
- MDC XML Encoding is supported by repository engine 3.0, and it is the native format of the MDC OIM. It expresses meta data in attribute-normal format. MDC XML Encoding conforms to the XML specification published by the World Wide Web Consortium (W3C).

XIF and MDC XML Encoding are two alternate encoding mechanisms. They are not compatible.

**Note** In practice, because MDC OIM and MDC XML Encoding are not supported by repository engine 2.0, you cannot pair OIM 1.0 with MDC XML Encoding unless you are running repository engine 3.0. MDC OIM and XIF are not compatible. You cannot pair MDC OIM with XIF under any circumstances.

The following table lists which versions of the OIM and XML interchange format you can use with each repository engine version.

Engine version	OIM version	XML version
2.0	OIM 1.0	XIF
3.0	OIM 1.0	XIF or MDC XML Encoding
	MDC OIM	MDC XML Encoding only

## **Meta Data Interchange Combinations**

The following table recommends an XML interchange format for each information model source-target combination, for all versions of the repository engine.

	_	0	Recommended XML interchange
2.0	OIM 1.0	OIM 1.0	XIF

3.0	OIM 1.0	OIM 1.0	MDC XML Encoding
3.0	OIM 1.0	MDC OIM	MDC XML Encoding
3.0	MDC OIM	MDC OIM	MDC XML Encoding
3.0	MDC OIM	OIM 1.0	MDC XML Encoding

## **See Also**

**Information Models** 

OIM in Meta Data Services

Retaining Legacy Components in a Repository

**Upgrading from Earlier Versions** 

**Upgrading the Repository Engine** 

XML in Meta Data Services

# Using Repository Engine Features with Older Databases

Upgrading a repository database to version 3.0 makes repository engine 3.0 features available to the database. Depending on the DBMS you use to manage the database, however, you can encounter some exceptions. A few newer features are not supported on older versions of DBMS products. This topic provides more information about these exceptions.

## **DBMS Version Exceptions**

The majority of repository engine features work identically for all supported DBMS products and versions.

The following table details which features are unavailable for certain DBMS versions when you upgrade a repository database to repository engine 3.0.

Database type	Features
Microsoft® SQL Server <sup>TM</sup>	Bit properties cannot be added to existing
6.5	tables.
	View generation and view-based queries are
	not supported.
SQL Server 7.0	View generation and view-based queries are
	not supported.
SQL Server 2000	All features are supported.
Microsoft Jet 3.5 and later	Bit properties cannot be added to existing
	tables.
	View generation and view-based queries are
	not supported.
	Some performance optimization techniques are
	not supported.

**Note** Converting a Microsoft SQL Server database to a more recent version of SQL Server does not automatically generate views. Generating views and database conversion are separate tasks.

## **About Operating Systems**

You can run SQL Server 2000 Meta Data Services on the following Microsoft Windows® operating systems: Windows 98, Windows NT® 4.0, and Windows 2000. Integration with Microsoft Distributed Transaction Coordinator (MS DTC) runs only on a computer that is running Windows 2000. For more information, see Integration with Distributed Transaction Coordinator.

#### See Also

**Generating Views** 

IReposProperty2 Interface

**Optimizing Repository Performance** 

**Repository Databases** 

Retaining Legacy Components in a Repository

**Upgrading and Migrating a Repository Database** 

**Upgrading the Repository Engine** 

# **Using Meta Data Browser**

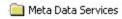
Microsoft® SQL Server™ 2000 Meta Data Services supports browsing of registered repository databases through Meta Data Browser. You can use Meta Data Browser in the following ways:

- From within SQL Server 2000 using SQL Server Enterprise Manager
- As a stand-alone snap-in that you add through Microsoft Management Console (MMC) and run separately from SQL Server Enterprise Manager

The way you use the browser determines the set of features and functionality that you can work with.

## Using Meta Data Browser in SQL Server Enterprise Manager

To run the browser from within SQL Server Enterprise Manager, click **Meta Data Services**.



In SQL Server, Meta Data Services provides storage service for SQL Server meta data, including meta data associated with specific Data Transformation Services (DTS) packages and all online analytical processing (OLAP) meta data. Repository tables are included in the SQL Server msdb system database to support management and storage of SQL Server meta data.

In SQL Server Enterprise Manager, Meta Data Browser enables you to view meta data that you create and store in **msdb**. You can view this data in the Contents folder.

When you use Meta Data Browser in SQL Server Enterprise Manager, you function in End User mode. End User Mode provides read-only access to the repository database. You can view information about any meta data that you store in the repository.

SQL Server Enterprise Manager purposely restricts the functionality of Meta Data Browser to protect the meta data that it uses. Modifying or deleting native

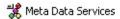
meta data can corrupt your SQL Server installation. For this reason, actions that put your meta data at risk are not supported in this mode.

## **Running Meta Data Browser Separately**

Meta Data Services can be run separately from SQL Server Enterprise Manager as a stand-alone MMC snap-in. You can add Meta Data Services to a console to work with other SQL Server repository databases, using a wider range of functionality.

After you add Meta Data Services to the MMC, you can run the Meta Data Browser to register the repository databases you want to work with. You can work with any SQL Server repository database created in SQL Server version 6.5 or 7.0, or in SQL Server 2000. However, you must use version 3.0 of the repository engine that is distributed with SQL Server 2000.

To run the browser in stand-alone mode, click **Meta Data Services**.



#### See Also

Viewing Meta Data in Meta Data Browser

Working with Contents in Meta Data Browser

Working with Information Models in Meta Data Browser

Working with Repository Databases in Meta Data Browser

Meta Data Fundamentals

Repository Databases

Storage Strategy in a Repository Database

# **Viewing Meta Data in Meta Data Browser**

Meta Data Browser presents content in different ways, depending on whether you run the browser from within SQL Server Enterprise Manager or in standalone mode. In addition, display options that you select can expand the kinds of meta data that you can view. Icons are used to visually identify the kinds of meta data displayed in the browser.

## Viewing Meta Data in SQL Server Enterprise Manager

Meta data in SQL Server Enterprise Manager is presented in the Contents folder. In SQL Server Enterprise Manager, repository tables are defined within the **msdb** database. In SQL Server Enterprise Manager, **msdb** is the only database that provides viewable content in Meta Data Browser.

Initially, the **msdb** database does not contain any meta data for you to view. However, after you add content to a repository, you can view it by expanding the Contents folder in Meta Data Services.

Adding content to a repository occurs when you choose to save to Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services (for example, when saving Data Transformation Services (DTS) packages).

## **Viewing Meta Data in Stand-Alone Mode**

When you run Meta Data Browser as a separate, stand-alone Microsoft Management Console (MMC) snap-in, you can work with multiple repository databases, and you can view content and perform tasks that are not available otherwise.

Meta Data Browser organizes content by repository database. Depending on the browse mode you select for your database, content can be further organized into folders named Contents and Information Models. Within each folder, objects and collections are arranged within a hierarchy to facilitate browsing.

• The Contents folder shows meta data stored in the repository database. Meta data varies depending on the database. In some repository databases, for example, meta data can be application objects, database

objects, or some other software artifact.

• The Information Models folder shows the <u>information models</u> that are installed in a repository database, and the classes and relationship definitions that they contain. Information models can be viewed only in Administrator browse mode.

## See Also

Icons Used in Meta Data Browser

Selecting Browse Mode in Meta Data Browser

Setting Display Options in Meta Data Browser

**Using Meta Data Browser** 

Working with Contents in Meta Data Browser

Working with Information Models in Meta Data Browser

# **Icons Used in Meta Data Browser**

This table shows and describes the icons used in Meta Data Browser.

Icon	Description
品	Information model icon
<b>*</b> \$	Class icon
. <u>J</u>	Connection object icon
-0	Interface icon
⊶	Relationship icon
	DTS package icon
•	Generic object icon
<b>3</b>	Data source icon
귤	OLE DB provider icon
å.	Class diagram icon
•	Attribute icon
•	Method icon
F:	Tagged value icon
	Generic collection icon

## **See Also**

Setting Display Options in Meta Data Browser

<u>Using Meta Data Browser</u>

# Working with Repository Databases in Meta Data Browser

When you run Meta Data Browser as a stand-alone snap-in, you can register any Microsoft® SQL Server™ database that contains repository tables. Registered repository databases are listed in the SQL Server 2000 Meta Data Services folder. All registered repository databases are grouped together on the same level.

## **Upgrading Databases**

After you register a database, you have the option of upgrading the database to the latest format. Upgrading applies the most recent repository SQL schema to a repository database that you formatted using a version of the repository engine that is older than the current repository engine. Although you can register repository databases created with earlier versions of the repository engine, upgrading gets the most current repository SQL schema tables that support new repository engine features. During the upgrade process, new tables are added, and data from existing tables is copied to the new table format. Custom tables that you create remain unchanged.

To upgrade a SQL Server repository database, right-click the registered database and then click **Properties**. This opens the **Repository Properties** dialog box. Providing the required information in this dialog box completes your option selections.

#### **More Database Related Tasks**

The following table contains links to topics that can help you learn more about related tasks. For more information about repository databases, see <u>Repository Databases</u>.

Topic	Description
Selecting Browse Mode in	Describes how to select End User, Power
Meta Data Browser	User, or Administrator browse mode.

	When working with a repository database, the scope of actions that are available to you depend on the browse mode you choose for the database.
Setting Display Options in	Describes how to make optional items
Meta Data Browser	viewable in the browser.
Registering a Repository	Explains how to add a new repository
Database in Meta Data	database to the list of registered databases.
<u>Browser</u>	
<b>Editing Registration Properties</b>	Explains how to edit registration
<u>in Meta Data Browser</u>	properties.
Deleting Registration	Explains how to delete registration
Properties in Meta Data	information and thereby unregister a
<u>Browser</u>	database.
Exporting to XML	Explains how to copy repository data to an
	XML file.

## **See Also**

Repository Properties Dialog Box

Repository SQL Schema

<u>Upgrading From Previous Versions</u>

**Using Meta Data Browser** 

Working with Contents in Meta Data Browser

Working with Information Models in Meta Data Browser

# Registering a Repository Database in Meta Data Browser

When you run Meta Data Browser as a stand-alone snap-in, you can register repository databases. You can register only repository databases that you created in Microsoft® SQL Server<sup>TM</sup> 6.5 or 7.0, SQL Server 2000, or the SQL Server Runtime Engine. Registering a repository database makes it available to users of Meta Data Browser. Later, if you want to make the database unavailable, you can unregister it by deleting registration information.

When you register a repository database, you must provide connection information similar to SQL Server 2000 database registration.

Before you can register a repository database, it must already exist. SQL Server 2000 Meta Data Services does not create repository databases.

After you register a database, you can upgrade it to use the newest features of the repository engine. For more information, see <u>Working with Repository</u> Databases in Meta Data Browser.

To register a repository database, right-click **Meta Data Services**, and then click **Register Database**. This opens the **Database Registration Properties** dialog box. Providing the information requested in this dialog box completes the database registration.

#### See Also

Deleting Registration Properties in Meta Data Browser

Editing Registration Properties in Meta Data Browser

**Database Registration Properties Dialog Box** 

Selecting Browse Mode in Meta Data Browser

# **Selecting Browse Mode in Meta Data Browser**

When you run Meta Data Browser as a stand-alone snap-in, you can set a browse mode at the repository database level to determine the scope of actions that are available to you. Browse modes include End User, Power User, and Administrator. You can choose different browse modes for each database.

The scope of actions for each browse mode includes the following:

- Creating, editing, and deleting registration information can be performed in all browse modes.
- Viewing the Contents folder and setting display options can be performed in all browse modes.
- Creating, editing, and deleting objects and object properties can be performed in Power User and Administrator browse mode.
- Viewing the Information Models folder can be performed in Administrator browse mode. In addition, only Administrators can view repository identifiers.

Browse modes apply exclusively to repository databases and have no impact on Microsoft® SQL Server<sup>TM</sup> 2000 user modes, Microsoft Windows NT® 4.0 user modes, or Microsoft Windows® 2000 user modes. In other words, being a repository Administrator does not confer administrator rights in SQL Server.

You cannot change the browse mode if you are running Meta Data Browser from within SQL Server Enterprise Manager. From within SQL Server Enterprise Manager, Meta Data Browser can only be run in End User mode.

To set a browse mode, you must select it during database registration. The browse mode you select is part of the registration information. To select a different browse mode, you must edit registration properties. For more information about the scope of actions each user mode supports, see <a href="Database Registration Properties Dialog Box">Database Registration Properties Dialog Box</a>.

## **See Also**

Editing Registration Properties in Meta Data Browser
Registering a Repository Database in Meta Data Browser
Working with Contents in Meta Data Browser
Working with Information Models in Meta Data Browser

# **Setting Display Options in Meta Data Browser**

For the meta data that you view in Meta Data Browser, you can set display options to alternately show or hide collection and inherited property information. If you show collections, you can set additional options to show or hide empty collections and reverse relationships.

The following figure shows a collection icon and a relationship icon, respectively.



0-0

Display options are context-sensitive. The options you set affect the current selection and all child nodes that branch from it. You can set display options for all or part of the selected database. The display options you select remain in effect until you reset the options. Display options are independent of browse mode selections.

To set display options, right-click a registered repository database or an object within the database, and then click **Browse Options**. This opens the **Meta Data Services Browser Display Options** dialog box. Providing the required information in this dialog box completes your option selections.

#### See Also

Meta Data Services Browser Display Options Dialog Box

# **Editing Registration Properties in Meta Data Browser**

When you run Meta Data Browser as a stand-alone snap-in, you can edit the properties of a registered repository database to change connection information, choose a different database, or change browse mode.

To edit database registration, right-click a repository database from Microsoft® SQL Server™ 2000 Meta Data Services, and then click **Edit Database Registration**. This opens the **Database Registration Properties** dialog box. For more information about repository database properties, see <u>Database</u> <u>Registration Properties Dialog Box</u>.

#### See Also

Deleting Registration Properties in Meta Data Browser

Registering a Repository Database in Meta Data Browser

# Deleting Registration Properties in Meta Data Browser

When you run Meta Data Browser as a stand-alone snap-in, you can remove a repository database from the Meta Data Services folder by deleting its registration information. To delete a repository database, use the database deletion features of Microsoft® SQL Server<sup>TM</sup> 2000.

To delete database registration, right-click a repository database from SQL Server 2000 Meta Data Services and then click **Delete**. Meta Data Services prompts you to confirm the deletion.

#### See Also

<u>Editing Registration Properties in Meta Data Browser</u>

Registering a Repository Database in Meta Data Browser

# **Exporting to XML**

You can export any kind of meta data stored in the repository to an XML file, including instance data from the Contents folder and information model elements from the Information Models folder.

The format of the XML is defined by MDC OIM XML Encoding. For more information, see <u>Using XML Encoding</u>.

To export to XML, right-click an element and then click **Export to XML**.

The scope of an export varies depending on whether the selected element is related to other elements. For elements that are related through containing or aggregate relationships, the XML export file includes the related data. In the Contents folder, where the relationship between parent and child elements is a containing relationship, the export file includes multiple XML tagged elements. In contrast, relationships between elements in the Information Models folder are typically not containing relationships. As a result, XML export files for elements in the Information Models folder include XML data for only the selected node.

#### See Also

Working with Contents in Meta Data Browser

Working with Information Models in Meta Data Browser

# **Working with Contents in Meta Data Browser**

You can use Meta Data Browser to discover facts about objects already implemented in, designed for, or defined by an application. If you are an application developer, browsing the contents of a repository can help you identify the best object to choose when building a new application. You can also browse the contents to view details about an existing Data Transformation Services (DTS) package or application component.

The kind of content that you can view varies depending on how you run the browser and the options you select. For more information, see <u>Using Meta Data Browser</u> and <u>Viewing Meta Data in Meta Data Browser</u>.

#### **Inside the Contents Folder**

The Contents folder shows object instance data in a repository database. You can view objects, object properties, collections, and <u>relationships</u>. Your browse mode selection determines how you interact with the Contents folder. Your display option selections determine what is visible. Except for sequenced relationship collections, the order in which objects appear is undefined. If a database does not show a Contents folder, it is probably not a repository database.

**Note** The browser does not track workspace or <u>repository object</u> version data. Object properties do not indicate the workspace to which an object belongs. However, some version information is surfaced through property information. Version labeling can be particularly helpful in identifying how a specific object version is deployed.

## **Objects**

Objects can contain collections and other objects. If you choose not to display collections, you can more easily view object relationships. In the browser, object relationships are rendered hierarchically. Expanding an object brings its related objects into view. For example, expanding a **Car** object can show additional objects for **Engine**, **Body**, and **Tires**. Expanding **Engine** can show an additional object for **Parts**.

The following figure shows a generic object icon.

For each object, you can view properties that tell you a great deal about each object. Properties are listed in alphabetical order. For more information about properties, see Working with Object Properties in Meta Data Browser.

If you are a repository Administrator, you can delete, rename, and remove objects from collections. For more information, see <u>Deleting, Renaming, and Removing Objects from Collections</u>.

#### **Collections**

Collections can contain objects or be empty. When you display collections, you can see how objects are grouped. For example, because **Engine**, **Body**, and **Tires** are elements of **Car**, displaying collection information can reveal that **Engine**, **Body**, and **Tires** are members of an **Elements** collection under **Car**.

The following figure shows a generic collection icon.



#### See Also

<u>Selecting Browse Mode in Meta Data Browser</u> <u>Setting Display Options in Meta Data Browser</u>

# Working with Object Properties in Meta Data Browser

You can display object properties to view the characteristics of individual objects.

To view properties, right-click an object, and then click **Properties**. This opens the **Repository Object Properties** dialog box. For more information about the options on this dialog box, see <u>Repository Object Properties Dialog Box</u>.

In addition to viewing properties, when you run Meta Data Browser as a standalone snap-in, you can edit property values of objects in the Contents folder. You must be a repository Administrator or Power User to edit property values. Editing property values can only be performed on objects in the Contents folder. The Information Models folder is always read-only.

To edit a property value, select a property for an object in the Contents folder and type a different value in the **Value** column. For more information about how to become a repository Administrator or Power User, see <u>Selecting Browse</u> Mode in Meta Data Browser.

## **Understanding Property Information**

Object properties expose **Class Name** and a property set that reveals details about a particular object. Each object supports a property set that varies from object to object and folder to folder.

In the Contents folder, **Class Name** identifies the information model class used to create the object. If you understand information models, knowing about the class reveals abstract data about an object that can be useful. The property set defines object characteristics. Property values are instance data about the object. The property set originates from the properties supported by the abstract class.

In the Information Models folder, **Class Name** identifies the repository API definition used to build the object class. A repository API definition can be an **InterfaceDef**, a **ClassDef**, or some other definition. The property set defines object class characteristics. Type information about the object class is expressed through property values. The property set originates from the properties

supported by the repository API definition.

You can search SQL Server Books Online for more information about repository API definitions and properties. For example, you can search for **InterfaceDef**. You can also search for specific properties, such as **CreateByUser**, **Name**, and **VersionLabel**.

## **Special Properties**

Custom objects support a property set that is determined by a model designer. For more information about custom object properties, check with the author of the model.

Enumeration objects are associated with property definitions. The purpose of an enumeration object is to provide a list of predefined values for a given property. In the browser, when you view a property that is enumerated, you can select one of the enumerated values from a predefined list. The list appears in the **Values** column of the property.

Virtual properties are a special category of properties that typically do not have persistent data. However, if the aggregation objects are stored in the repository database, virtual properties will appear in the Meta Data Browser.

#### See Also

**Creating Objects** 

Deleting, Renaming, and Removing Objects from Collections

Repository API Reference

Working with Contents in Meta Data Browser

Working with Information Models in Meta Data Browser

# **Creating Objects**

You can populate a repository with new object instance data that you create. The objects you create must be compatible with the definitions of an installed information model. For example, to create a new table object, the underlying information model must support a table object definition.

To create objects, right-click the parent object under which the new object is to reside, and then click **Create Object**.

To create objects, the following conditions must be satisfied.

Criteria	Conditions
Environment	Stand-alone snap-in. You cannot create objects from
	within Enterprise Manager. For more information
	about stand-alone mode, see <u>Using Meta Data</u>
	Browser.
Browser mode	Administrator or Power User. For more information,
	see Selecting Browse Mode in Meta Data Browser.
Kind of meta data	Contents folder meta data. You cannot create objects in
	information models. For more information about
	Contents, see <u>Viewing Meta Data in Meta Data</u>
	Browser.

Creating an object requires choosing a collection, choosing a class, and defining a name. Your choices determine the characteristics assumed by the object.

- Collections that you can choose from belong to the parent element. For example, if the parent object is a **Table** object, you can choose from the collections that belong to the **Table** object (in this case, the **Columns** collection). You cannot create collections. You can only choose from collections that are provided for the parent object by way of the information model.
- Class selections are derived from the collection. If a collection supports multiple classes, you must choose which class to use.

• Names are object names that you define. The name you define must be under 255 characters in length. Names can include spaces.

After you create the object, you can edit properties. For more information, see <u>Working with Object Properties in Meta Data Browser</u>.

## See Also

Deleting, Renaming, and Removing Objects from Collections
Working with Contents in Meta Data Browser

# Deleting, Renaming, and Removing Objects from Collections

When you run Meta Data Browser as a stand-alone snap-in, you can delete, rename, or remove object instance data from collections. You must be a repository Administrator to perform these actions. You can only perform these actions on object instance data that appears in the Contents folder.

When you delete, rename, or remove an object from a collection, your changes are immediately saved in the repository database. Except for renaming, you cannot reverse these changes using the Meta Data Browser.

To perform these actions, right-click an object in the Contents folder, and then click **Delete**, **Rename**, or **Remove**.

- **Delete** permanently deletes the object instance data from the repository database.
- **Rename** activates an in-place editor so that you can type over the existing name. If you are naming an object in a collection that requires unique names, Meta Data Services does not allow you to duplicate a name in that collection. Otherwise, duplicate names are supported.
- **Remove** deletes reference information that associates an object with a collection. The object instance data is not deleted.

For more information about how to become a repository Administrator, see <u>Selecting Browse Mode in Meta Data Browser</u>.

#### See Also

**Creating Objects** 

Working with Contents in Meta Data Browser

Working with Object Properties in Meta Data Browser

# Working with Information Models in Meta Data Browser

If you are an application developer or model designer, you can use Meta Data Browser as a visual tool for tracking your inventory of information models. You can also discover facts about the objects, collections, and properties that make up an information model.

The Information Models folder shows the information models that are installed in a repository database. The Information Models folder is visible only to repository Administrators, and it is available only when you run the browser as a stand-alone snap-in. Except for sequenced relationship collections, the order in which information model objects appear is undefined.

Information models are the blueprints of items you see in the Contents folder. In its native format, an information model is typically a network of related objects. In Meta Data Browser, information models are depicted hierarchically. When you expand an object, the child nodes that appear are the objects related to the expanded object. Depicting a network structure in a hierarchical format means that some objects appear multiple times. For example, in a relationship, each object will appear as a child node of the other object.

You can install ready-to-use information models (stored as .rdm files) using Meta Data Browser or a separate installation tool that comes with the Microsoft® SQL Server™ 2000 Meta Data Services Software Development Kit (SDK). Installing an information model extracts information from a model and places it into tables in the repository database. How an information model is used from that point forward can vary considerably across tools, users, and environments.

For more information about how to become a repository Administrator, see <u>Selecting Browse Mode in Meta Data Browser</u>.

## **Inside the Information Models Folder**

In the browser, you can expand an installed information model to view the objects and collections it contains. You cannot create, modify, or delete an

information model, its objects, or its properties. To perform those tasks, you must use modeling tools.

You can expand an information model to do the following.

- View objects, collections, interfaces, and members.
- View the inheritance tree for each interface.
- View read-only object properties to see how an object is defined.

#### See Also

**Information Models** 

**Information Model Fundamentals** 

**Installing Information Models in Meta Data Browser** 

Working with Contents in Meta Data Browser

Working with Object Properties in Meta Data Browser

# **Installing Information Models in Meta Data Browser**

When you run Meta Data Browser as a stand-alone snap-in, you can install ready-to-use information models to the repository database, making them available to applications, application developers, and model designers. You must be a repository Administrator to install an information model. For more information about how to become a repository Administrator, see <a href="Selecting Browse Mode">Selecting Browse Mode in Meta Data Browser</a>.

After you install an information model, it remains in the repository database. Deleting an information model is not currently supported by Microsoft® SQL Server<sup>TM</sup> 2000 Meta Data Services.

To install an information model, right-click the Information Models folder of the database in which you want the model to reside, and then click **Install Information Model**. This opens the **Install Information Model** dialog box so that you can choose a Repository Distributable Model (RDM) file to install. RDM files are compiled information model files. RDM files are generated from a modeling tool in the Meta Data Services Software Development Kit (SDK).

**Note** It is also possible to install information models programmatically or by command line, without using Meta Data Browser. For more information about this alternate approach, see <u>Installing Information Models</u>.

#### **See Also**

**Information Model Fundamentals** 

**Information Models** 

Meta Data Services SDK

# Meta Data Browser User Interface Reference

Meta Data Browser includes dialog boxes. Several of these dialog boxes support direct access to context-sensitive Help topics.

To open a context-sensitive Help topic, click **Help** or press F1 when the dialog box is open. You can also choose a topic from the following list.

Dialog box	Description
Meta Data Services Browser	Contains options that you can set to increase
<b>Display Options Dialog Box</b>	the amount of viewable content.
Repository Object Properties	Displays the properties of an information
Dialog Box	model, object, or collection.
Database Registration	Connects a repository database so that you
Properties Dialog Box	can view meta data in the browser. It also
	includes options for selecting a browse
	mode.
Repository Properties Dialog	Displays the server name, database name,
Box	and repository database version. It also
	includes an option for upgrading the
	database format.
Create New Object Dialog	Populates an information model with new
Box	object instance data.

## See Also

**Using Meta Data Browser** 

# Meta Data Services Browser Display Options Dialog Box

Use this dialog box to set display options that filter collection and property data in Meta Data Browser.

This dialog box appears when you right-click any item within the Meta Data Services folder, and then click **Browse Options**. The display options that you select apply to the current item and all items that branch from it.

## **Collection Options**

These options enable you to view collections, and to select whether empty collections and reverse relationships are visible.

#### **Display Collections**

Specifies whether object and relationship collections are displayed. The following icon identifies a collection.



Selecting this option enables the following additional options.

## **Show Empty Collections**

Specifies whether empty collections are displayed. Empty collections contain no members.

## **Show Reverse Relationships**

Specifies whether reverse relationships are displayed. Reverse relationships show a relationship from the opposite perspective and may contain additional data not otherwise available. For example, given this one-to-one relationship, *car contains engine*, the reverse relationship is *engine contained by car*. The reverse relationship represents a data definition that shows whether one type of engine is used in many types of cars.

## **Properties Options**

These options enable you to show or suppress inherited property values for objects in the Contents and Information Model folders.

### **Show All Class Properties**

Specifies whether inherited properties are displayed. An inherited property is an alias of a property on a base interface. When you show inherited properties, you see both the alias and the base property from which the alias is derived.

Consider a **Person** object that has a property named **Phone Number**. Through aliasing, you could base a new property of a different object on **Phone Number**. For example, a **Customer** object could have a property named **Customer ID** that is derived from **Phone Number**. When you select the **Show All Class Properties** option, both the inherited property (**Customer ID**) and the base property (**Phone Number**) are presented as sibling properties.

#### See Also

**Repository Collections** 

Repository Relationship Objects

Setting Display Options in Meta Data Browser

Working with Object Properties in Meta Data Browser

# **Repository Object Properties Dialog Box**

Use this dialog box to view properties of a repository object. The name of the current object determines the name of the dialog box. For example, if the object name is **TblCustomer**, the dialog box name is **TblCustomer Properties**.

If you are running Meta Data Browser as a stand-alone snap-in, and you are running in either Power User or Administrator browse mode, you can edit property values.

This dialog box appears when you right-click an object in the Contents folder or in the Information Models folder, and then click **Properties**.

## **Options**

#### Name

Displays the name of the object or information model.

#### **Class Name**

Displays the name of the class upon which an object is based. The class name is a repository API class that provides meta data about the object.

#### **Columns**

#### Name

Shows the name of the property.

## **Data Type**

Shows the data type of a property.

#### **Value**

Shows the user-defined value of a property. Depending on your browser mode, you can click the value to invoke an in-place editor used to modify the value.

#### See Also

Editing Registration Properties in Meta Data Browser
Repository API Reference

Selecting Browse Mode in Meta Data Browser

Working with Object Properties in Meta Data Browser

# **Database Registration Properties Dialog Box**

Use this dialog box to register a new repository database with Meta Data Browser. You should only register repository databases. Databases that do not contain the standard repository SQL tables do not have viewable content in Meta Data Browser.

This dialog box is available when you run Meta Data Browser as a stand-alone snap-in. It is not available for use within SQL Server Enterprise Manager. For more information, see <u>Using Meta Data Browser</u>.

This dialog box appears when you right-click the Meta Data Services folder or an existing registered repository database and then click **Register Database**. You can also open this dialog box by clicking **Edit Database Registration**.

## **Options**

#### Server

The name of a registered computer running Microsoft® SQL Server™ 2000.

#### Connection

Provides two authentication approaches that are identical to SQL Server database registration. For more information, search on "registered SQL Server properties" in SQL Server Books Online.

#### **Database**

The name of the repository database you want to register. Be sure to choose a database that has the repository SQL tables. These tables provide data for the Contents and Information folders.

#### **Browse Mode**

Specifies the browse mode for browsing contents and information models. The default selection is End User. Power User and Administrator modes provide additional browsing capability.

#### **End User**

The default user mode selection. End User mode supports read-only access to the Contents folder.

#### **Power User**

Power User mode supports read/write access to the Contents folder. In the Contents folder, Power Users can display and edit object properties.

#### **Administrator**

Administrator mode supports read-write access to the Contents folder, read-only access to the Information Models folder, and support for creating, editing, and deleting repository database registration. Repository Administrators can also install information models and view object identifier properties (such as **ObjID** and **InternalID**) in the **Repository Object Properties** dialog box.

Browse mode selection is available during new database registration. The browse mode that you select determines your degree of interaction with the database. To change the browse mode, you must edit the registration properties.

The browse mode selection you make applies to you. You cannot set or predetermine the browse mode for other users.

#### See Also

Editing Registration Properties in Meta Data Browser

Registering a Repository Database in Meta Data Browser

Selecting Browse Mode in Meta Data Browser

# **Repository Properties Dialog Box**

Use this dialog box to specify an upgrade option and view version information about the repository database. The name of the current repository database determines the name of the dialog box. For example, if the repository database name is DevTools, the dialog box name is **DevTools Properties**. If you open this dialog box from within Enterprise Manager, the dialog box name is **Meta Data Services Properties**.

This dialog box appears when you right-click a repository database and then click **Properties**.

## **Options**

#### Server

The name of an installed instance of Microsoft® SQL Server™ 2000.

#### **Database**

The name of the repository database.

## **Repository Database Version**

Shows a version number and a point release number, if applicable. Version information identifies which version of the repository engine was used to create the database. The outcome of creating a database varies depending on the DBMS you are using. For a SQL Server database, creating a database causes SQL Server to create repository SQL tables in an empty database that you provide. For more information, see <a href="Connecting to a SQL Server Repository Database">Connecting to a SQL Server Repository Database</a>.

## **Upgrade**

Updates repository SQL schema tables in a repository database so that you can use new repository engine features. After you upgrade a repository database, you cannot work with it using previous versions of the repository engine.

This button is enabled only when the database version is less than the current version of the repository engine.

# See Also

Registering a Repository Database in Meta Data Browser

Repository SQL Schema

**Upgrading from Earlier Versions** 

# **Create New Object Dialog Box**

Use this dialog box to create a new repository object and add it to the specified collection. You can add repository objects instance data in the Contents folder. You cannot create new objects for items in the Information Models folder.

This dialog box is available when you run Meta Data Browser as a stand-alone snap-in, in either Power User or Administrator browse mode. This dialog is not available within SQL Server Enterprise Manager. For more information, see <u>Using Meta Data Browser</u>.

This dialog box appears when you right-click an item in the Contents folder and then click **New Object**. The item that you select determines what type of object you can create and where it is located.

## **Options**

## Collection to add object to

Lists the collections defined on an object. Objects are always added to collections. If collections are visible (that is, the **Display Collections** option is enabled), the collection is selected for you. If collections are hidden, you must select the collection in which to place the object.

## **Object Type**

Lists the object types that are allowed for the collection. Only valid object types are available for selection.

## **Object Name**

The *objName* of the object. This name is required when adding objects to a naming relationship. The name of the object should reflect its context. For example, a name of an interface member should include the package, interface, and member name. Names composed of multiple parts must be separated by colons (for example, BaseInterfaces:IUMLPackage:Visibility).

# See Also

**Creating Objects** 

Working with Object Properties in Meta Data Browser