Microsoft Repository 1.0

Read-me Contents

This read-me file contains these topics:

What is Microsoft Repository?

Microsoft Repository is an enabling technology for defining and populating information models. It is used by engineering and software development tools to share information about engineered artifacts. Engineered artifacts are things like software components, manufactured components, documents, maps, Web pages - anything about which you might want to share information. Using a repository to store this information enables these tools to cooperate with one another and to share information between tools.

Microsoft Repository stores repository data in a relational database. Two database management systems are supported: Microsoft SQL Server 6.5 and the Microsoft Jet database engine.

Microsoft Repository Components

What are the major parts of Microsoft Repository? They are:

- **The repository engine** offers services to access and manage the repository itself and the information it stores. It provides ActiveX and DCOM interfaces for object-at-a-time navigating and updating of objects, relationships, and properties.

- **The Type Information Model** is the fundamental information model that is used to define all other information models.

- **The Microsoft Development Objects Model** (MDO Model) is an information model that can track Visual Basic projects (and the elements within those projects). Third party developers are working on products that extend the Visual Basic Integrated Development Environment (IDE) to provide greater capabilities to manage and re-use Visual Basic components. Some of these Third party products will use Microsoft Repository.

- **The Repository Add-in for Visual Basic** is an add-in module that automatically populates the MDO Model with information about your Visual Basic projects as you work in the Visual Basic IDE. The Repository Add-in for Visual Basic is not enabled during the Visual Basic installation process, and you cannot enable it via the Visual Basic Add-in Manager. Third party tools that need the Repository Add-in will enable it during tool installation.

- **The Repository Browser** is a basic browser that you can use to explore the information models and information model instance data that is stored in a repository.

The Microsoft Repository Files on Your Hard Disk

These files have been added to your Windows SYSTEM (or SYSTEM32) directory:

- REPODBC.DLL - The repository engine
- REPRC.DLL - Resources for the repository engine

- REPOSTRY.HLP - Online help for the repository engine
- REPOSTRY.CNT - Table of contents for the repository engine online help

In the HELP directory, immediately underneath the directory to which you installed Visual Basic 5.0, you will find these files:
- REPVB.HLP - Online help for the Repository Add-in
- REPVB.CNT - Table of contents for the Repository Add-in help

Some of the Microsoft Repository files are installed in the REPOSTRY directory, immediately underneath the directory to which you installed Visual Basic 5.0. Within the REPOSTRY directory (where this README.WRI file resides), there are two subdirectories:
- BIN - Executable files for Microsoft Repository
- INCLUDE - Include files for the Microsoft Repository API

Within the BIN subdirectory, you will find these files:
- REPVB.DLL - The Repository Add-in for Visual Basic
- REPVBRC.DLL - Resource file for Repository Add-in
- REPVBTIM.DLL - Type definitions for the MDO Model
- REPBROWS.EXE - A basic repository browser

Repository Installation Notes

Microsoft Repository is installed during Visual Basic installation. However, the Repository Add-in for Visual Basic is not automatically activated.

Microsoft Repository requires that the Data Access component is also installed. By default, the Data Access component is installed during Visual Basic installation. However, if you do not install the Data Access component and then install a third party tool that uses Microsoft Repository, you will encounter this error message when you start Visual Basic 5.0:

> "An error occurred while opening the Microsoft Repository database. Specified driver could not be loaded due to system error 126 (Microsoft Access Driver (*.MDB)). Microsoft Repository Add-in for Visual Basic is shutting down."

To correct this problem, install the Data Access component.

How Do I Activate the Repository Add-in for Visual Basic?

If you install a third party development tool that is an extension of the Visual Basic Integrated Development Environment and uses Microsoft Repository, the Repository Add-in for Visual Basic will be activated when you install the third party tool. If you are not using such a tool, we don't recommend that you activate the Repository Add-in for Visual Basic. Doing so will introduce overhead that provides no benefit to you.

The repository engine is capable of running without the Repository Add-in for Visual Basic. The Repository Add-in is only required to keep the MDO Model instance data synchronized with your Visual Basic projects. Consequently, if you install a 3rd party development tool that uses Microsoft Repository but does *not* use the MDO Model, then the Repository Add-in for Visual Basic is not required and will not be activated.

Information for Third Party Tool Developers

If you are a third party tool developer working on a Visual Basic development tool that will use Microsoft Repository, the installation program for your tool must activate the Repository Add-in for Visual Basic. To activate the repository add-in, you must add the following line to the VBADDIN.INI file (if it is not already present):

```
Repository.VBAddIn=1
```

The VBADDIN.INI file is located in the directory to which Windows was installed (for example, C:\WINDOWS).

Repository Browser Command Line Parameters

The Repository Browser program (REPBROWS.EXE) accepts four optional command line input arguments. They are:

Access=*filename***;**
DSN=*dataSourceName***;**

UID=*userID***;**
PWD=*password***;**

Fill in the appropriate values for the italicized items. The semicolon is required on the end of each argument.

Repository API Changes

- TXN_QUERY_TIMEOUT is a new type of option that can be set and retrieved via the **SetOption and GetOption methods** of the IRepositoryTransaction interface. The value for this option specifies the number of seconds that an outstanding query is allowed to execute before it is canceled. The default value for the option is 120 seconds.
- TXN_DBMS_READONLY is a new type of option that can be retrieved via the **GetOption method** of the IRepositoryTransaction interface. The value for this option specifies whether or not you can make changes to the currently open repository database. If the value of the option is zero, you can make changes to the database. If the value is nonzero, you cannot make changes to the database.
- The maximum length of an annotational property is 220 characters, not 255 characters. The constant **PROPVALSIZE** determines this limit.
- The Interface method of the IRepositoryItem interface is used to obtain an interface pointer to an alternate interface that an Automation object exposes. This method will accept one of three different types of input parameter to specify the interface whose pointer is to be retrieved: the object identifier of the interface definition object (a repository object identifier), the interface identifier for the interface (a GUID), or the name of the interface (a character string).
- A new method (**ExecuteQuery**) has been added to the IRepositoryODBC interface. The ExecuteQuery method has this signature:

  HRESULT ExecuteQuery(BSTR QueryString, IObjectCol **ppICol)

  QueryString is a character string that contains an SQL query (or the name of a stored procedure) that is to be executed against the open repository database. The selected columns that are returned by the SQL query must consist of either the internal identifier column (IntID), or the internal identifier column and the type identifier column (TypeID). The results of the query are returned as an object collection, and an IObjectCol interface pointer is returned to the caller via the *ppICol output parameter.
- The **get_Item method** of the IObjectCol interface does not support retrieving objects by name. The get_Item method of the ITargetObjectCol and IRelationshipCol interfaces supports retrieving objects by name, but only when the object collection is a naming collection. The name is not required to be unique.
- The name assigned to a project via the naming relationship of the **Contents collection** of the IMpoProjectItemContainer interface is adorned to ensure that the name is unique among the collection of all Visual Basic projects in the collection. The adornment follows these rules:
  - If the project file is a network file, then the UNC name for the file is used as the name.
  - If the project file is a local file, then the string [computerName]filename is used as the name.
  - If the project is unsaved, then the string [computerName]~processId~projectName is used as the name.

  The FileName property of the IMpoProjectItem interface contains the unadorned file name for a project.
- The IReposProperties interface is a dependent interface that is used to access the collection of repository properties that are attached to any interface that inherits from IRepositoryDispatch. As a dependent interface, it is only accessible by obtaining an interface pointer via the get_Properties method of the IRepositoryDispatch interface. A new method (**get_Type**) is supported on the IReposProperties interface. The get_Type method has this signature:

  HRESULT get_Type(VARIANT *psTypeID)

  The get_Type method retrieves the type of the interface that derives its base behavior from IRepositoryDispatch; that is, it returns the object identifier for the interface definition object to which the derived interface conforms.

Repository Database Restrictions

Microsoft Repository 1.0 can use either Jet or SQL Server to manage the repository database. These restrictions apply:

- If SQL Server is used, the version must be 6.5 or later.
- If Jet is used, care must be taken to avoid large amounts of data being accessed within the scope of a single transaction. The Jet engine uses an in-memory cache to speed up query processing. This cache continues to grow inside of a transaction until a Commit or Abort is performed. If your repository application is accessing a large amount of data within a single transaction (which might be done to isolate the view of the retrieved data from uncommitted changes of other processes), the cache can grow to the point where it consumes enough memory to cause the application to fail. To avoid this situation, periodically Commit your transaction, even if it is a read-only transaction.

Repository Multithreading Restrictions

When programming a multithreaded repository application, take care to synchronize repository database commit operations between your application threads. Specifically, you must synchronize the use of these C++ methods:

- IRelationshipCol::Count
- IClassDef::ObjectInstances
- IInterfaceDef::ObjectInstances
- IRepositoryODBC::ExecuteQuery

If You Used Microsoft Repository from the Visual Basic Beta

- If you install this version of Visual Basic 5.0 over a previous installation of a Visual Basic 5.0 beta release, there are a few Microsoft Repository files that are obsolete and can be deleted to save space. Starting from the directory to which you install Visual Basic 5.0, look for and delete these files:
  REPOSTRY\README.WRI
  REPOSTRY\BIN\BROWSER.EXE
- In your Windows system directory, look for and delete these files:
  REPAUTO.CNT
  REPAUTO.HLP
  REPCOM.CNT
  REPCOM.HLP
- The beta format of the repository database is no longer valid. If you used the beta version of Microsoft Repository, you must delete your old repository database. In the MSAPPS\REPOSTRY directory underneath your Windows directory, look for and delete this file:
  REPOSTRY.MDB

For More Information

You can go to these sources for more information about Microsoft Repository:

- Check for the latest news on our Web site at http://microsoft.com/repository!
- The help files mentioned earlier in this file are online help references for the Microsoft Repository API. Information is provided for both the C++ and the Visual Basic programmer.
- In Visual Basic Books Online, start at the topic *Using the Repository with Visual Basic*. Within this book, you will find the *Repository Programmer's Guide, Repository Reference,* and *Repository Add-in Reference*.
- Check out the samples under the REPOSTRY directory on the Visual Basic CD that contains tools and other utilities.