Product Documentation

ER/Studio® Repository

Installation and Administration Guide

4th Edition

Version 6.0
Published January 2011

Embarcadero Technologies, Inc. is a leading provider of award-winning tools for application developers and database professionals so they can design systems right, build them faster and run them better, regardless of their platform or programming language. Ninety of the Fortune 100 and an active community of more than three million users worldwide rely on Embarcadero products to increase productivity, reduce costs, simplify change management and compliance and accelerate innovation. The company's flagship tools include: Embarcadero® Change Manager™, CodeGear™ RAD Studio, DBArtisan®, Delphi®, ER/Studio®, JBuilder® and Rapid SQL®. Founded in 1993, Embarcadero is headquartered in San Francisco, with offices located around the world. Embarcadero is online at www.embarcadero.com.

January 28, 2011

CORPORATE HEADQUARTERS
100 CALIFORNIA STREET
12TH FLOOR
SAN FRANCISCO, CALIFORNIA
94111 USA

EMEA HEADQUARTERS
YORK HOUSE
18 YORK ROAD
MAIDENHEAD, BERKSHIRE
SL6 1SF, UNITED KINGDOM

ASIA-PACIFIC HEADQUARTERS
L7. 313 LA TROBE STREET
MELBOURNE VIC 3000
AUSTRALIA

Contents

Welcome to ER/Studio Repository

ER/Studio Repository is a server-side model management system that solves the day-to-day challenges of modeling in a team environment, where model collaboration, versioning, security and component reuse are vital. The ER/Studio Repository allows multiple users to be extremely productive while collaborating on data and business process modeling projects with real-time concurrent access that allows team members to share and re-use assets across projects.

- Collaborative modeling with concurrent model and object access
- Model and object version management
- Standardized enterprise data dictionary
- Real-time objects status notification

New Features of ER/Studio Repository 6.0

ER/Studio Repository Performance Improvements

ER/Studio Repository software is now multi-threaded, enabling some processes to occur concurrently. The following describes some important rules that affect multi-threading on the Repository:

- If a Check In transaction is being processed on a given Diagram or Data Dictionary, all other operations on that Diagram or Data Dictionary must wait until the Check In completes.
- If an Add or Check In transaction is being processed, all other Add or Check In transactions must wait until the initial transaction completes. This same rule applies to Set Named Release, Branch and Branch/Merge Check In operations.
- Most other common tasks, including Log Ins, Log Outs, Gets, Check Outs and Security Center operations are processed concurrently by the server, but may be affected by database level-locking mechanisms.

Important Name Change Notice

The following describes recent product names changes in the ER/Studio family

Old Name	New Name	Short Form of New Name
ER/Studio	ER/Studio Data Architect	ER/Studio DA
ER/Studio Repository	ER/Studio Repository	Repository
ER/Studio Portal	ER/Studio Portal	Portal
EA/Studio	ER/Studio Business Architect	ER/Studio BA

About This Document

The *ER/Studio Repository Installation and Administration Guide* is the primary reference for the Repository. It provides both installation and administration information. In HTML Help format it serves as online help within ER/Studio DA. It is also distributed in PDF format for easy downloading and printing.

For pointers to additional information not included in this document, see Additional Product Information.

Additional Product Information

The Embarcadero Web site is an excellent source for additional product information, including white papers, articles, FAQs, discussion groups, and the Embarcadero Knowledge Base.

Go to www.embarcadero.com/support, or click any of the links below, to find:

- Documentation
- Online Demos
- Technical Papers
- Discussion Forums

ER/Studio Repository Installation

NOTE: The following information is for installation only. If you are upgrading ER/Studio Repository, please see the Readme for important upgrade information.

The server-side Repository provides for improved teamwork and enterprise collaboration. ER/Studio Repository is included in the Enterprise edition of ER/Studio. ER/Studio Repository is comprised of two components:

- **Repository Server**: The link between ER/Studio Data Architect and the Repository database. The server is responsible for creating the SQL code necessary to query the database to retrieve data or to insert data into the database. The Repository Server consists of the following components:
  - Repository Communication Server
  - Repository Event and Dispatch Server
  - Repository Database Server
- **Repository Database**: Stores information about the Repository Diagrams and their objects.

This section is comprised of the following topics:

ER/Studio Repository Installation Checklist

Use the following checklist to ensure ER/Studio Repository is correctly installed and configured.

Done	N/A	Requirements
		Download ER/Studio Enterprise, which includes the Repository. You can download a 14-day trial version of ER/Studio Enterprise, which you can license later, from www.embarcadero.com/downloads.

Done	N/A	Requirements
		Review the ER/Studio ReadMe. For the most current installation and usage information, see the ReadMe at docs.embarcadero.com.
		Ensure system compatibility. For system requirements, see System Requirements on page 9.
		Optimize the repository database. For information on how to optimize Repository performance, see Preparing to Install ER/Studio Repository on page 11.
		Plan for repository growth. For information on how to estimate the required size of the Repository database, see Planning for Repository Database Growth on page 11.
		Install ER/Studio Repository For instructions on how to install the Repository, see Installing the Repository on page 13.
		Connect to the Repository For instructions on how to connect to the Repository from ER/Studio Data Architect, see Connecting to the Repository on page 14.
		Run and test ER/Studio Data Architect For instructions on how to use ER/Studio Data Architect, see the *ER/Studio Data Architect Evaluation Guide* at docs.embarcadero.com.

SYSTEM REQUIREMENTS

Review the following requirements before you install ER/Studio Repository. Adhering to these requirements optimizes ER/Studio Repository performance.

	ER/Studio Repository Requirements	
Hardware	**Repository Database Server**	**Repository Server**
	As required for your database platform	
Processor		Pentium 4 or higher
RAM		2 GB
Disk Space		50 MB
DVD Reader		yes
Operating System		Any of the following: • Windows Server 2008 (32-bit and *64-bit modes) • Windows Vista • Windows XP • Windows 2003 (32-bit mode) For 64-bit configurations, the 32-bit database client libraries are required.
Software	Transaction Control Protocol (TCP) up and running Western European character set (UNICODE and UTF8 are not supported)	TCP up and running
	Supported DBMS (see Database Server Requirements on page 10)	
Privileges	DBMS privileges to create objects on the server	DBMS privileges to create objects on the server
	Local administrator privileges	Local administrator privileges

NOTE: The hardware requirements are based on an ER/Studio Repository installation with five to 10 users. For more than 10 users, ensure your hardware surpasses the previously stated hardware requirements to accommodate the number of transactions that increase as the number of users increases.

DATABASE SERVER REQUIREMENTS

The following table lists the databases the Repository supports. In order for the Repository to communicate with the database, the machine running the Repository must also have the corresponding RDMS client utility installed. The RDMS client utilities must also be installed on the machine running ER/Studio Data Architect if you want to use ER/Studio Data Architect to reverse engineer databases originating at any of these platforms.

ER/Studio Repository Database Server Requirements	
Supported Repository DBMS	**Corresponding RDBMS Client Utility Software**
IBM DB2 LUW 7.x *, 8.x, and 9.x Server	Corresponding version of IBM DB2 UDB Client Utilities (7.x, 8.x or 9.x)
Microsoft SQL Server 2000, 2005, and 2008	Corresponding version of SQL Server Client Access (2000, 2005 or 2008)
Microsoft SQL Server 7.0	Microsoft Data Access Component (MDAC) 2.6 or later
Oracle 8.1.x, 9.0.x, 9.2.0.x, 10g, and 11g	Corresponding version of Oracle Client Utilities (8.1.7.4, 9.0, 9.2.0, 10g, or 11g)
Sybase ASE 12.x or 15.x	Sybase 11.9 or later Client Utilities

Important Notes

- ER/Studio Repository has been certified to work with the above database platforms using 32-bit versions of the database client software. The 64-bit clients are not supported.
- If you install the ER/Studio Repository Server and the ER/Studio Repository database on the same machine, consider surpassing the hardware requirements described previously.
- You must use ER/Studio Data Architect 9.x or later with ER/Studio Repository 6.x.
- If you are upgrading ER/Studio Repository and are also using ER/Studio Portal, upgrade ER/Studio Portal to the latest version.
- The RDBMS Client Utility software must be installed before installing the Repository server.
- If you are using Microsoft SQL Server and use case-sensitive nomenclature, you may get connection error messages from the Repository. To avoid this, create an empty database with a non-case-sensitive character set (such as Latin_General_CI_AS), then initialize the database with the Repository Database Maintenance utility.
- For an IBM DB2 UDB 7.x server, the machine where your DB2 instance resides must have the C++ compiler for the Stored Procedures to work correctly.
- For all versions of an IBM DB2 UDB server, to ensure optimal Repository performance, you must have a user temporary tablespace and the database must be tuned so it does not run out of memory or log space. For more information, see Preparing to Install ER/Studio Repository on page 11.
- For all versions of IBM DB2 UDB server, the client connection to the Repository database must be registered as a system ODBC Data Source. The data source name must match the Database Alias specified in the client connection.

PLANNING FOR REPOSITORY DATABASE GROWTH

Regardless of the platform, allocate at least 300 MB of space to the database and implement auto extend so that the database can accommodate increases in the size and number of models. Also, allocate 50 MB of space to the database log file.

The table below describes the initial size of some sample Repository databases before any diagrams are added, and the approximate database sizing for sample small, medium, and large diagrams.

RDBMS	Initial Size (MB)	Small Diagram (MB)	Medium Diagram (MB)	Large Diagram (MB)
Oracle 8	6	0.5	14	64
Microsoft SQL Server 7	7.8	0.7	14.7	40.9
Sybase ASE 12	3.9	0.8	28.9	89.1
IBM DB2 UDB	3.2	0.8	14.6	50

The table below describes the characteristics of small, medium, and large sample diagrams:

Sample Diagrams	Entities	Attributes	Views	Relationships	DM1 File Size (KB)
Small	22	125	2	20	184
Medium	302	2516	32	606	2153
Large	722	10629	0	1542	6967

To ensure the Repository is installed correctly, see the following:

- Preparing to Install ER/Studio Repository on page 11.
- Installing the Repository on page 13.

NOTE: If you are installing ER/Studio Repository 6.x, install ER/Studio Data Architect 9.x or later. Previous versions of ER/Studio Data Architect are not compatible with ER/Studio Repository 6.x.

PREPARING TO INSTALL ER/STUDIO REPOSITORY

For Microsoft SQL Server

If you are using Microsoft SQL Server and case-sensitive nomenclature, you may get connection error messages from ER/Studio Repository. To remedy this, create an empty database with a non-case-sensitive character set (such as Latin1_General_CI_AS), then initialize the database with the Repository utility. Also, ensure you are using Mixed Authentication for server security.

For IBM DB2 for LUW

If ER/Studio Repository will be stored on IBM DB2 for LUW, perform the following procedures to ensure optimal Repository performance:

- Creating an IBM DB2 for LUW User Temporary Tablespace on page 12.
- Tuning the IBM DB2 for LUW Database on page 12.

Creating an IBM DB2 for LUW User Temporary Tablespace

The DB2 UDB database must have a user temporary tablespace with at least a 4 KB page size to create the temporary tables.

TIP: Use a system-managed user temporary tablespace because when managed by the system the user temporary tablespace can grow automatically.

TIP: You can use DBArtisan to create a system managed user temporary tablespace.

Create a user temporary tablespace using the tool of your choice or you can use the following SQL:

NOTE: Before you run this script, modify the path for the container.

```
    CREATE USER TEMPORARY TABLESPACE REPOTEMP
        PAGESIZE 4096
        MANAGED BY SYSTEM
        USING('D:\DB2\NODE0000\SQL00002\SQLT0004.0')
//  here you specify the physical location of the table space
        EXTENTSIZE 32
        PREFETCHSIZE 16
        BUFFERPOOL IBMDEFAULTBP
        OVERHEAD 24.10
        TRANSFERRATE 0.90
    ;
```

Tuning the IBM DB2 for LUW Database

This procedure tunes the DB2 database so that it does not run out of memory and log space.

1 Increase the application heap size and application control heap size to 2048 × 4KB.

TIP: To increase the application heap size, right-click the database in the DB2 Control Center, and then click Configure. Then, switch to Performance and modify the values for Application Heap Size (applheapsz) and Application Control Heap Size (appctlheapsz).

2 Increase the log file space to 10,000 × 4 KB (or 40 MB).

3 Change the number of primary and secondary log files from 3 and 2, to 10 and 10. This provides for a total of 400 MB of primary log space, which should be sufficient for medium to large diagrams.

TIP: To increase primary and secondary log file sizes, use the Logs tab of the Database Configuration page in the IBM Control Center.

INSTALLING THE REPOSITORY

NOTE: You need an account on the DBMS server Database with administrator privileges to create a new Repository database.

1 Log on to Windows with local administrator privileges.

2 Start the **Repository 6.x** installation program.

3 Walk through the installation wizard.

The options are straightforward, but make sure you choose the following option:

- **Create new Repository database**

Notes

The installation adds tables, indexes, views, and stored procedures. The installer is for the most part self-explanatory. Follow the instructions on screen. The following are notes for those parts that require additional explanation:

- **Database name**: To install a Repository server on a DBMS server that already has a Repository server installed on it, enter a unique database name for the new server. Any changes to the database name are also applied to the data and log file names.
- **Automatic Growth for Microsoft SQL Server 7.0 or later**: To allow for automatic growth of the database file, click Allow Growth, and then enter a Growth Rate. Express the growth rate as either a percentage of the current database size or the number of MB to add to the database when the database is becoming full. Select the maximum size of the file as either Unlimited or click SIze, and then enter the maximum file size in MB.
- Use the **New Oracle User** dialog to create a new user and provide temporary tablespace information for the user. The dialog is self-explanatory, except for the following user identification options:
  - **Password**: If selected, indicates that Oracle should identify the user with the password provided.
  - **Externally**: If selected, indicates that Oracle should verify the database user name against an existing operating system user name.
  - **Globally**: If selected, indicates that Oracle will permit access to the user by obtaining user name and password information from the security domain central authority. This option is only available for Oracle 8.

- **Error Log Viewer**: The Error Log View appears if there is a problem creating your Repository database. If you get an error, review the Repository Database requirements to make sure you meet them. For further assistance with creating your Repository Database, contact Embarcadero Technologies Technical Support.

Connecting to the Repository

1 Start **ER/Studio Data Architect** or **ER/Studio Viewer**.

2 From the **Main** menu, choose **Repository > Repository Options**.

3 In the **Server Machine** field, type the host name or IP address of the Repository server machine, and then click **OK**.

4 Log on to the Repository as follows:

Choose **Repository > Log In**, enter your user name and password, and then click **OK**.

You are now ready to use ER/Studio Data Architect with ER/Studio Repository. For details on using ER/Studio Repository, see the *ER/Studio Data Architect User Guide* at docs.embarcadero.com.

UNINSTALLING ER/STUDIO REPOSITORY

From the Windows Add or Remove Programs Control Panel, select Embarcadero Repository X.X, click **Remove**, and then follow the prompts to remove all ER/Studio Repository components.

NOTE: This does not remove the Repository database. The database administrator must remove the Repository database.

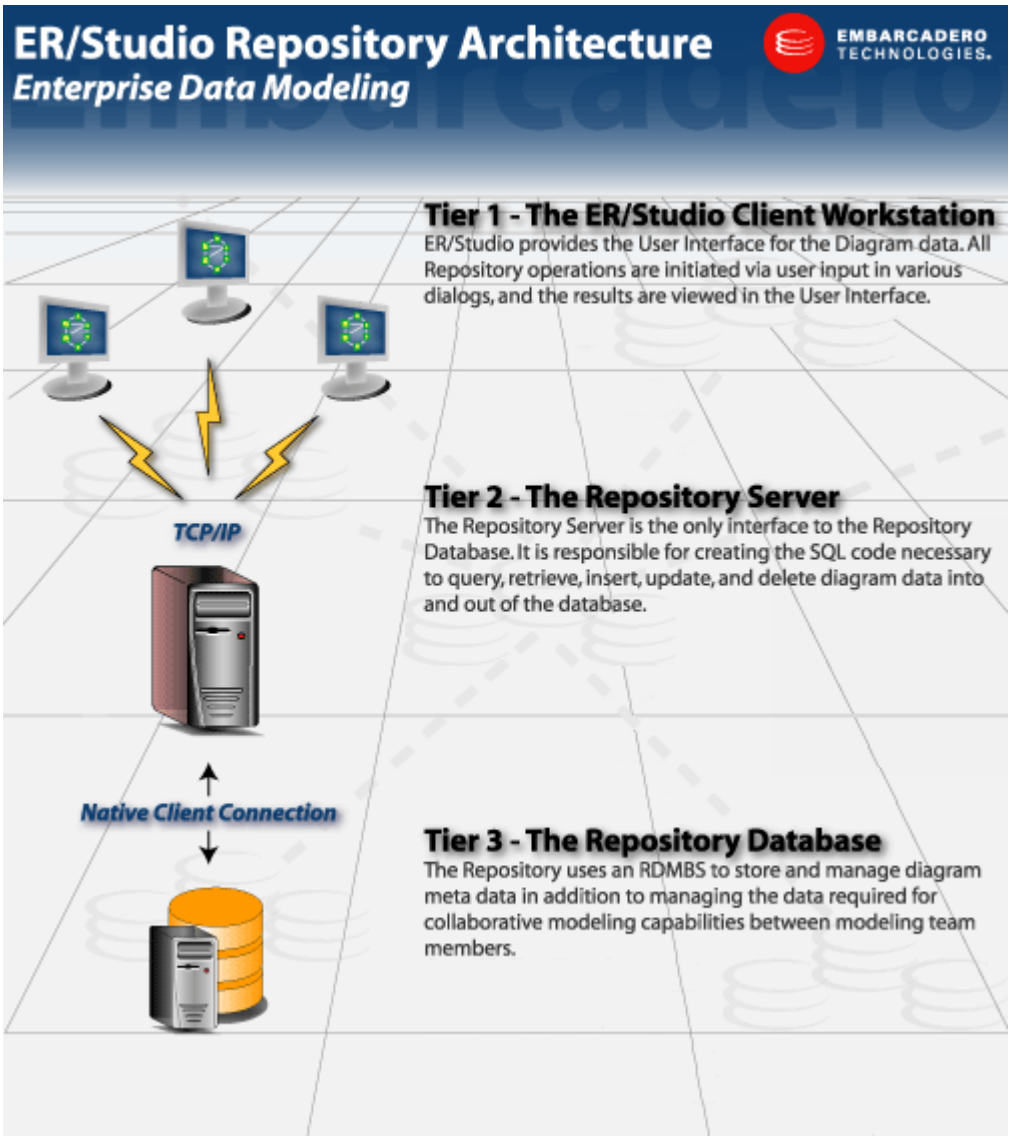
ER/Studio Repository Administration

This section is for the person who installed ER/Studio Data Architect (ER/Studio DA) and who is now configuring ER/Studio DA or administering ER/Studio Repository databases. It contains the following topics:

Understanding and Maintaining the Repository

This section includes the following topics:

REPOSITORY ARCHITECTURE AND DESIGN



The Repository relies on a native client connection and the TCP/IP protocol to send and receive information between ER/Studio DA, the Repository Server, and the Repository Database. Communications between the client and server are secure. SSL support in the client and repository allows you to specify the common URL and port to support secure communications between the client and the repository.

There is a metadata diagram of the Repository schema in the System Models directory (Repository300MetaModel.dm1). It includes relationships between entities as documentation whereas the actual database, for performance reasons, does not include FK constraints. The file shows the propagation chain which could be helpful for writing queries to run against the Repository.

For a complete list of software and installation requirements, and supported Repository Databases, see the Database Server Requirements on page 10.

Repository Client

ER/Studio DA provides the user interface for the Diagram data stored in the Repository. The ER/Studio Repository menu gives you access to all Repository functions and features. To access and use the Repository, you must have ER/Studio DA running.

From the Repository tab, you can access Repository metadata, search and create reports through the ER/Studio Portal.

Repository Server

The Repository Server links ER/Studio DA to the Repository Database. The Repository Server contains the SQL code necessary to query the database to retrieve data or to insert data into the database. The Repository Server runs this code when it receives a request to execute an operation that requires it to read data in the database and return the appropriate information and when it receives a request to execute an operation that requires it to write new data to the database.

> **NOTE:** Because the Repository Server is responsible for inserting data into the database, and retrieving data from the database, you can only have one Repository Server for each Repository Database.

The Repository Server manages the state of the Repository data, which it stores in the Repository database. It records modeling change operations as requested by ER/Studio DA users.

When the server receives a request, it does the following:

1. Checks to make sure the user has the necessary permissions to perform the operation.
2. If the operation requests data from the Repository database, the server retrieves the data from the database and sends it to the ER/Studio DA instance from which the request originated.
3. If the operation attempts to store data in the Repository database, the Repository Server writes the data out to the database.
4. Notifies all ER/Studio DA "clients" of the new state of the Repository and sends them the appropriate information to update the displays.

Managing Repository Files

The Repository Server installation includes several folders, described below. Depending on your configuration options, particularly if the `-deleteoff` option is set on your server, you may want to manually remove older files collecting in your Repo In/Out folders as described here.

It is always safe to delete a .pri or .pro file. It is safe to delete any files with old dates since the files are generally processed immediately upon being written to the directory. This applies to .out and .err files. Files with a .in extension should not be deleted and should not be in the folders for more than the time it takes to process them, unless the Repo Services have been stopped.

Folder	Description
Repository	This folder contains the Repository subfolders, listed below. It also contains the executable files for the three Repository services, along with the Repository Server Licensing application, a copy of the license agreement, and a copy of a compiled help file that contains information about installing and licensing the Repository.
Logs	This folder is a subdirectory of the Repository folder. It is used to store log files that are used by each of the Repository services to store information about events or transactions.
RepoIn	Stores the request files received from an ER/Studio DA client until they are processed by the server. If you have older files collecting here: 1. Open the directory in the Windows Explorer. 2. Sort the files by Date Modified with the oldest files at the top. 3. Select all files starting from the top and stopping wherever you feel safe. A good idea is to select all files modified before the current day. 4. Delete those files.
RepoOut	Stores files being sent from the Repository server to the ER/Studio DA client until they are processed.

When installing the Repository, you can install the server and database on the same machine, or on different machines. Regardless of where you install each component, we recommend that you do a full backup of both components on a regular basis.

Repository Database

The Repository uses a database to store information about the Repository Diagrams and their objects. The database is updated each time a user checks out, checks in, adds, or deletes any element of a Diagram. In addition, the Repository database stores the history of certain Repository operations, as well as all security rules and data.

CAUTION: Do not directly update Repository tables because the database is a transactional system designed to handle collaborative model use. Updating the Repository database outside of ER/Studio DA can corrupt the data. The SQL scripts in the `Embarcadero\Repository\Utilities\MasterScripts` and `SQLScripts` directories are used by the Repository within ER/Studio DA and should be used outside ER/Studio DA only at the request and under the guidance of Embarcadero Support.

NOTE: You can query the data and select data from the Repository for reporting purposes. Commonly used SQL scripts in the `Embarcadero\Repository\Utilities\QueryScripts` and `Embarcadero\ERStudio Data Architect X.X\SQLCode\Repo_Queries` directories have been provided for your convenience. However, you can create your own scripts and to help you do this, a meta model is provided in the `\Program Files\Embarcadero\ERStudio Data Architect X.X\System Models` directory.

Moving the Repository

The following describes how you can move the Repository server and database to a new machine or to a new database platform.

1. Backup the current Repository database or export the current Repository database schema, depending on the DBMS platform.
2. Restore the database on the new Repository database server.
3. Install the Repository application server on a new machine if necessary.
4. Run the program "ERStudio Repository Database Maintenance" on the machine where the new Repository application server is installed.
5. Choose the option "Connect to an existing ER/Studio Repository database".
6. Specify the connection information for the new database.
7. If necessary, provide the new hostname to Repository users so they can connect to the new Repository.

Optimizing Repository Performance

The following tips can help optimize the performance of the Repository:

- **Dedicated Server**: The performance of the Repository is directly affected by the number and size of diagrams and the size of the data transfers between the client and the Repository. We recommend using a dedicated server for the Repository, unless you're using industrial-strength hardware such as multi-processor rack servers that can manage many applications simultaneously.

- **Separate the Repository Server from the Repository Database:** The Repository Server and Repository Database do not have to reside on the same machine, however such a configuration may reduce wait times due to reduced contention. If you decide to separate the Repository Serve and the Repository Database, for optimal performance, the Server should reside in the same physical local as the users.
- **Clearing Repository Folders**: On the Repository Server machine there are `RepoIn` and `RepoOut` folders in the `\...\Embarcadero\Repository\` folder. Sometimes network problems cause entries to be created in these folders. Many files in these folders can hinder Repository performance. Periodically purging the files in these folders should be done frequently, such as nightly, when no one is connected to the repository.
- **Checking In/Out Diagrams versus Checking In/Out Objects**: You can use information in the `RepoSrvDb.Log` to optimize performance and to make recommendations on how users can use the Repository more effectively. For example, checking in or checking out an entire diagram is less efficient than selecting multiple objects for check in or check out. The `RepoSrvDb.Log` file is found on the Repository application server. It provides a record of who is using the Repository and when, the request type initiated, the processing result, and the total time to process the request. Total Process Seconds is listed for each request and includes the time required to parse the request file, insert and/or retrieve data from the database, and build and write out a results file.
- **For Oracle environments**:
  - **Separate Tablespaces**: Create separate tablespaces for Repository tables, `ERSTUDIO_DAT` and indexes, `ERSTUDIO_IDX`. The extent of each of these tablespaces should be 1 MB or larger. In some very large Repository environment it may be beneficial to further separate the larger tables and their indexes creating, for example `ERSTUDIO_DAT2` and `ERSTUDIO_IDX2` in addition to the original tablespaces.
  - **Rebuild Indexes**: The Repository index, `ERSTUDIO_IDX` should be rebuilt at least once a month. Depending on how many diagrams are added, the index may need to be rebuilt on a weekly basis.
  - **Analyze and Compute Statistics**: The response of queries to the Repository can be improved by running DBMS_STATS for recent Oracle version or the Analyze Table and Compute Statistics statements for older versions. These commands gather information about data distribution within the tables and indexes and update the data dictionary with this information. This enables the cost-based query optimizer to make more intelligent decisions about how to efficiently process SQL statements that access the tables or indexes.
  - **Oracle Logs**: Check In operations can cause checkpointing to occur which can interrupt other Repository actions to the detriment of response time. To minimize the effect checkpoint has, increase the size of the logs, set the `log_checkpoint_interval` parameter in `init.ora` to zero, and reduce the size of the log buffer.

Contingency Planning

To ensure optimal Repository performance and availability, backup the Repository schema frequently and designate another server as the backup server in case of hardware failure. Then, if necessary, you can install a new set of Repository services onto the backup server and point the Repository to a freshly restored backup.

Running Reports on the Repository

The Utilities folder in the Repository directory contains many SQL scripts. The SQL Agent is used to run these.

Backing Up and Recovering the Repository

You should implement a backup and recovery process for your Repository Database. Repository stores critical data in the Repository Server Data folder.

When you add your locally created and managed *.DM1 files to the Repository server, the file is managed in the Repository Database (existing within Oracle, Sybase, SQL Server or DB2 tables). You can find these files in the ER/Studio Repository installation path. If you accepted the installation default, the files are located in:

- `...\Embarcadero\ERStudioX.X\Repository\Data`

NOTE: ER/Studio DA no longer uses overflow files. Long files are now added to the Repository Database. For more information, see Repository Database on page 20.

The Repository Server is the only interface to the Repository database. It is responsible for managing the state of the Repository data. The Repository Server is basically a transaction server for the database. The Repository database also stores the transaction history as well as all security rules and data. Because the Repository server is responsible for data transactions with the database there can only be one Repository server for each Repository database.

Querying the Repository

You can query your Repository when you want to run some simple SELECT reports. Below is an example of a simple cross-diagram report that selects a Diagram, Sub Models in the Diagram, and Entities in each Sub Model.

NOTE: Do not use the Repository for editing, deleting, or updating data. ER/Studio DA is the only application you should use for these operations.

Example

We created this example using ER/Studio DA's Repository meta model. This data model is in the installed application directory:

```
...\Program Files\Embarcadero\ERStudio Data Architect
X.X\SystemModels\Repository450MetaModel.dm1

SELECT dbo.DiagramVer.Name as "Diagram Name",
dbo.SubModelVer.Name as "Submodel Name",
dbo.EntityVer.Name as "Entity Name",
dbo.EntityVer.TableName as "Table Name"
FROM dbo.Diagram, dbo.DiagramVer, dbo.Entity, dbo.EntityVer, dbo.Model,
dbo.SubModelVer, dbo.SubModel
WHERE (dbo.Entity.LatestVersionID = dbo.EntityVer.EntityVerID AND
 dbo.Model.ModelID = dbo.Entity.ModelID AND
 dbo.SubModel.LatestVersionID = dbo.SubModelVer.SubModelVerID AND
 dbo.Model.ModelID = dbo.SubModel.ModelID AND
 dbo.DiagramVer.DiagramID = dbo.Model.DiagramID AND
 dbo.Diagram.LatestVersionID = dbo.DiagramVer.DiagramVerID)
```

ESTABLISHING SECURITY FOR THE REPOSITORY

Security of the Repository contents is managed through the Security Center by creating and then assigning users particular Roles and Privileges.

NOTE: To prevent an accidental lockout, the Admin user can't be deleted.

For example, we create a user JimB, give him a Role and call it DBA. The DBA role has many privileges, but none allow the DBA to modify a logical model. A Repository administrator can associate JimB with any model in the Repository and he will not be able to modify the diagram's logical model, but he can view it.

The Administrator can restrict user access to Repository items at the project, diagram, model, submodel, and data dictionary levels. To restrict access to the Repository, the Administrator creates users and then assigns the users a default role, which grants specific permissions to the user, or creates new roles to assign to users. Then the Administrator can grant users full access to some Repository items and restrict or deny access to others items.

Before creating users and roles, the Administrator should understand the following Repository concepts:

- **Cascading Security**: To make it easy for Administrators to assign a role globally to users when there are many diagrams in the Repository, such as when the user must have the same access permissions to many of diagrams, the Administrator can assign a user and role to various 'levels' of the ER/Studio Repository.

  For example, a Project can be created in the ER/Studio Repository that contains many diagrams. If you assign a user a role to this Project, any diagrams contained within the Project will be granted the same permission set. The Repository itself can act as this highest-level point to assign permissions that are cascaded down.

  Higher levels can be overwritten by assigning the same user different roles at a lower level, for example UserA may have the Viewer role at repository level but Modeler role for a specific diagram or submodel.

- **Client Side Security Caching**: To promote the concept of detached modeling collaboration (e.g. being logged out of ER/Studio Repository and possibly detached from the network), all security associated with the user by diagram is cached on the client when the user logs into ER/Studio Repository.

  NOTE: When security changes are made, users must re-login to update their permissions.

- **Super Use**r: Super User is a default Role created upon installation of ER/Studio Repository. The default 'Admin' user is assigned to the Super User role. This user can do anything to diagrams, users and roles managed in the ER/Studio Repository as it is granted all privileges. It is not removable or editable, and is only found at the 'Repository' level of the Repository Security area of the Security Center.

- **No Access**: *No Access* is a role that can be applied only at the project and diagram levels of the Repository. It is a quick and global mechanism to prevent any access whatsoever to diagrams managed in specific projects, or to individual diagrams themselves.

Notes

- When a user gets a file from the Repository, ER/Studio DA tracks the user and machine combination for that particular file. In addition, to enable users to model while they are "offline" or unable to connect to the Repository server, ER/Studio DA stores information about the security rights of the user and the diagram with the file, which has the following effects:
  - When a user is logged into the Repository, and attempts to open a DM1 file that was retrieved from the Repository by a different user/machine combination than the currently logged in user, ER/Studio DA indicates that the file cannot be opened due to the conflict. This ensures that one user does not attempt to work on or check in changes to a diagram that was originally retrieved by a different user. Because the Repository keeps track of object check outs based on the user and machine, only the user on the machine that originally checked out an object can check it back in.
  - Even if a user is not logged into the Repository, Repository DM1 files can be opened and worked on in ER/Studio DA. ER/Studio DA will load the security permission data for the user that originally retrieved the file from the Repository, and that security data will govern the rest of the working session until the user either logs in to the Repository or closes the file. If multiple files are open while the user is NOT logged in to the Repository they must all have been retrieved by the same user. Of course, this also applies when the user IS logged in.
  - The cached security data stored with each diagram is updated whenever an open Repository file is saved while logged in to the Repository, but if the file is not saved before closing it, then any changes to the security permissions for that user and diagram will not be cached with the file. So, if you plan to work offline, and the security settings have changed since the file was last saved, then you should open each file while logged into the Repository and save it before taking the files to work offline.
- Permission settings can be applied at the project or repository level, which might cause a conflict when two files are opened while a user is offline. If two files are opened that both have cached security information for the Repository level or for the same Project, and if the cached data differs, then the most recently saved data will be used and stored in both files if and when they are saved.
- If an admin makes changes to the security UI, such as changing permissions on folders to grant or revoke access or moving diagrams between projects with different permissions, users must re-login to update their permissions.

This section includes the following topics:

- Creating and Managing Roles on page 27
- Creating and Managing Users on page 32
- Granting and Prohibiting User Access to Repository Items on page 33

See Also

"Working with Repository Projects" in the *ER/Studio DA User Guide*

Creating and Managing Roles

ER/Studio DA lets you create customized roles with different sets of permissions. Repository Object Type Permissions are pre-defined privileges to operate on Repository items. You can assign them to a role, which gives users assigned to that role, permission to perform certain Repository operations. You can use the many available privileges to create specific roles to suit your environment.

For example, you can create and assign some roles like the following:

- **Repo Level Basic**, which has all of the Repository level permissions, assigned to the target user at the Repository level.
- **Logical Only Modeling**, which has all of the Diagram, Model, and Submodel level privileges that do not apply to the physical model, assigned to the target user at the diagram level for each diagram in the Repository or at least the ones that the target user is allowed to work on.

With the above roles and assignment, when the target user adds the diagram, the user can check it out, but cannot modify it. To allow the user modify the logical model immediately after adding it, the Admin would have to apply the "Logical Only Modeling" role to the target user at the Repository level.

NOTE: To access, create, update, and delete user information, a Repository administrator must have Repository Object Type Permission, Access Security Info and Update Security Info privileges.

Create, Update, and Delete Roles

1. Choose **Repository > Security > Security Center**.
2. Select the **Manage Roles** tab.
3. Click **New**.
4. Click an item in the **Repository Object Type** list and then in the **Repository Object Type Permission** area, assign the permissions you want to assign to the role for the selected object type.
5. Repeat step 4 for each **Repository Object Type**.
6. Click **Apply** and continue changing security settings and then when finished, click **OK** to exit the security center.
7. To apply the new role to existing users, see Assigning a Role to a User.
8. Inform affected users that they should log out of the Repository and then log in again to receive the security updates.

Notes

- Once created, the Administrator can select the role and update, delete or rename it at any time. Renaming the role does not affect the privileges users have, but if you delete a role users who had access to Repository items through that role, will no longer have access to those items.
- Users cannot modify objects without the necessary permissions, regardless of whether the user has the objects checked out.
- To facilitate immediate access to a newly added diagram (not projects) the Admin should set up privileges as follows: If a user gets a submodel and wants to add an entity, the user must have Create Entity permission in the model containing the submodel, as well as Add Member permission in that submodel. If the user deletes an entity, the user must have Remove Member permission in that submodel. In addition, if the user wants to select the "Delete From Model" check box, the user must have Delete Diagram Object permission in that model.
- Once you delete a role from the Repository, it will no longer be in the database.
- To access/create/update/delete user information, a Repository administrator needs to have the Access Security Info and Update Security Info Privileges applied to the Roles.
- Before you delete a role, you must unlink any diagrams that are assigned to it and delete any users assigned to it.
- The following lists the Repository Object Types, the permissions you can grant, and the common operations these permissions give access to:

Repository Object Types	Repository Object Type Permissions	Permitted Operations
Repository	Access Security Info	**Security Center**: View settings
	Update Security Info	**Security Center**: View and change settings
	Create Diagram	Add Diagram
	Update Diagram	Check Out Diagram, Check In Diagram Check Out Object(s), Check In Object(s) Undo Check Out Diagram, Undo Check Out Object(s) Redo Check Out Diagram, Redo Check Out Object(s)
	Delete Diagram	Delete Diagram If you delete a diagram in Repository, the file itself remains on the local disk.
	Create Enterprise Dictionary	Create Enterprise Dictionary
	Update Dictionary	Check Out Data Dictionary, Check Out Dictionary Object(s), Check In Data Dictionary, Undo Check Out Data Dictionary, Redo Check Out Data Dictionary
	Create Project	Create Project
	Delete Project:	Delete Project
Project	Add Project Member	Add Diagram to Project
	Remove Project Member	Remove Diagram from Project

Repository Object Types	Repository Object Type Permissions	Permitted Operations
Diagram	Bind Enterprise Dictionary	Create New Enterprise Data Dictionary, Bind Existing Enterprise Data Dictionary
	UnBind Enterprise Dictionary	Remove Enterprise Data Dictionary
	Compare Models	Run Compare/Merge Wizard
	Create Physical Model	Create Physical Model
	Delete Physical Model	Delete Model
	Set Named Release	Set Named Release
	Delete Named Release	Delete Named Release
	Rollback Diagram To Named Release	Rollback Diagram
	Update Diagram Properties	Edit Title Block Data, Edit Diagram Properties
	Create Data Flow	Create New Data Flow on the Data Lineage tab
	Delete Data Flow	Delete Data Flow from the Data Lineage tab
Data Dictionary	Create Dictionary Object	**Create**: Attachment Type, Attachment, Default, Rule, Data Movement Rules, Reference Value, User Datatype, Domain Folder, Domain, Reusable Trigger, Reusable Procedure, Library
	Update Dictionary Object	**Edit**: Attachment Type, Attachment, Default, Rule, Data Movement Rules, Reference Value, User Datatype, Domain Folder, Domain, Reusable Trigger, Reusable Procedure, Library
	Delete Dictionary Object	**Delete**: Attachment Type, Attachment, Default, Rule, Data Movement Rules, Reference Value, User Datatype, Domain Folder, Domain, Reusable Trigger, Reusable Procedure, Library
Logical Main Model	Create Diagram Object	**Create**: Entity, View, Relationship, View Relationship, Subtype Cluster, Subtype, Title Block
	Delete Diagram Object	**Delete**: Entity from Model, View from Model, Relationship from Model, View Relationship from Model, Subtype Cluster from Model, Subtype from Model, Title Block from Model
	Update Diagram Object	**Entity Editor**: Create/Modify/Delete: Attribute, Key, Key Attribute, Check Constraint **View Editor**: Modify View; Create/Modify/Delete: View Table, View Column **Key Editor**: Modify Key, Create/Modify/Delete Key Attribute **Relationship Editor**: Modify Relationship **Subtype Cluster Editor**: Modify Subtype Cluster **Edit Model**: Options, Properties
	Create Submodel	Create Submodel
	Delete Submodel	Delete Submodel

Repository Object Types	Repository Object Type Permissions	Permitted Operations
Logical SubModel	Add Member	**Submodel Editor**: Add to Submodel
	Remove Member	**Submodel Editor**: Remove from Submodel, Delete Entity from Submodel Remove Database View **Delete**: Relationship from Submodel, View Relationship from Submodel, Subtype Cluster from Submodel, Subtype from Submodel
	Update Display Properties	**Move/Resize**: Entity/Table, View, Title Block, Text Block, Subtype Cluster **Color/Font Changes**: Entity/Table, View, Title Block, Text Block, Relationship Line, View Relationship Line, Subtype Cluster **Move**: Relationship Line, View Relationship Line **Create/Modify/Delete**: Text Block Change Model Notation, Perform Layout, Zoom, Align Objects
Physical Model	Create Diagram Object	**Create**: Table, View, Relationship, View Relationship, Schema Object, Title Block
	Delete Diagram Object	**Delete**: Table, View, Relationship, View Relationship, Schema Object, Title Block
	Update Diagram Object	**Create/Modify/Delete**: Column, Index, Index Column, Check Constraint, View Table, View Column, Key Attribute, Key, Relationship, Subtype Cluster **Edit Model**: Options, Properties Change Database Platform
	Create Submodel	Create Submodel
	Delete Submodel	Delete Submodel
Physical SubModel	Add Member	**Submodel Editor**: Add to Submodel
	Remove Member	**Submodel Editor**: Remove from Submodel **Delete**: Table from Submodel, View from Submodel, Relationship from Submodel, View Relationship from Submodel, Schema Object from Submodel
	Update Display Properties	**Move/Resize**: Entity/Table, View, Title Block, Text Block, Physical Schema Object **Color/Font Changes**: Entity/Table, View, Title Block, Text Block, Relationship Line, Physical Schema Object **Move**: Relationship Line, View Relationship Line **Create/Modify/Delete**: Text Block Change Notation, Perform Layout, Zoom, Align Objects

Repository Object Types	Repository Object Type Permissions	Permitted Operations
Data Flow Model	Create Data Flow Object	**Create**: Data Flow, Data Lineage Component, Data Stream, Source
	Update Data Flow Object	**Edit** Data Flow, Object, Transformation, Data Stream, Source **Check In**: Data Flow, Object, Transformation, Data Stream, Source **Check Out**: Data Flow, Object, Transformation, Data Stream, Source **Undo Check Out**: Data Flow, Object, Transformation, Data Stream, Source **Redo Check Out**: Data Flow, Object, Transformation, Data Stream, Source **Note**: The user must have permission to update the diagram in order to update data flow objects in the diagram.
	Delete Data Flow Object	**Delete**: Data Flow, Data Lineage Component, Transformation, Data Stream **Note**: Deleting a table\entity component from the Data Lineage window does not delete the table\entity from the model. **Note**: The user must have permission to delete the diagram in order to update data flow objects in the diagram.
Data Flow Display	Update Display Properties	**Move/Resize**: Transformation, Component, Data Flow **Color/Font Changes**: Data Lineage Background, Component, Transformation, Data Stream **Diagram And Object Display Options**: Change any option in this dialog to control how the data lineage diagram displays. Perform Layout, Zoom, Align Objects, Layout Data Stream, Straighten Data Stream, Remove All Bends

Assigning a Role to a User

Assigning a role to a user grants the user all the permissions selected in the role.

1 Choose **Repository > Security > Security Center**.

2 *If the user is not currently assigned a role*, on the **Repository Security** tab, click the user name in the **Available Users** column and drag it onto a role name in the **Available Roles** column.

If the user was already assigned a role, on the **Repository Security** tab, click the user name the **Available Roles** column and drag it onto another role name in the **Available Roles** column.

3 Click **Apply** and then click **OK** to exit the **Security Center**.

4 Notify users that the must log out and then log in again to receive the security updates.

Creating and Managing Users

Managing users entails creating, changing, and deleting user account permissions and logins, checking in a users checked out documents under unusual circumstances. You can control the privileges granted to a user for a specific diagram, model, submodel or Enterprise Data Dictionary by associating the user with a role. Using Roles, you can choose the permissions granted a user for the Repository item.

NOTE: To access, create, update, and delete user information, a Repository administrator must have Repository Object Type Permission, Access Security Info and Update Security Info privileges.

Create, Edit, Delete, Deactivate, Reactivate, Log Out, and Check In Users

1. Choose **Repository > Security > Security Center**.
2. Click the **Manage Users** tab.
3. On the **Manage Users** tab, you can manage users by clicking their name in the list and then clicking a management option and following the prompts as required.
4. Continue making security changes and then click **OK** to exit the security center.
5. To assign permissions to users, see Assigning a Role to a User on page 31.
6. Notify users that they must log out and then log in again to receive the security updates.

   The following describe options that require additional explanation:

User Management area

- **New**: Click New to access the Create Repository User dialog.
  - **Directory Service User**: Select this option If you want the new user to login to the Repository using their Windows credentials. In this case ER/Studio DA accesses the LDAP server on the network to verify the user's credentials. Directory service users can check the Log in using current Windows account option on the Repository login dialog to use their Windows user ID and password to log into the Repository.
- **Force Check In Repository User**: When you use the Check In User option, you can choose how to handle the checked out objects.
  - **Convert to non-exclusive**: Enables the user to continue making changes to the checked out items. Other users can then check out the same items. When checking in non-exclusively checked out items, if the item was changed by another user since it was checked out, the user can resolve the differences and choose to override the other user's changes or to update his local version with the changes found in the Repository.
  - **Undo Check Out**: When the user next connects to the Repository, the status of his items will be updated to show that they are not checked out.

Notes

- You cannot delete or deactivate the Admin user, but you should change the Admin password from the default, which is Admin.

- The new user appears in the user list with a star indicating that this user does not yet have any roles assigned to it for any Repository items.
- Users must log out and log in again to receive any security updates.
- The Admin does not need to know the user's password to create a new password for the user.
- The Admin can look to the Manage Users tab to find out which users are logged in to the Repository and whether or not they currently have items checked out. Knowing if users have items checked out is particularly useful when upgrading the Repository, because after before upgrading the Repository the users should check all their items in and then do a clean get and check out of the items when the upgrade is complete. Knowing which users are logged in is useful when changing security center settings because after making any setting changes, the Admin should request online users to log off and log in again to receive the security center updates.
- The Administrator may want to user log a user out to receive security center updates. For example, a new diagram has been added to a project and the diagram shouldn't be accessible by all the users who have access to the project. The Administrator can make the necessary security changes for the project and then log out the users, forcing them to log in again to receive the security updates. Other unusual circumstances may arise when it is necessary for the Administrator to log a user out of the Repository.

Granting and Prohibiting User Access to Repository Items

On the Repository Security tab of the Security Center, users with the appropriate privileges can assign other users the appropriate access privileges for different objects, granting them permission to perform specific operations on a Repository item and prohibiting them from performing other operations on the same object.

Assigning Users Roles for Accessing Repository Items

1. Log in to the Repository.
2. Choose **Repository > Security > Security Center**.
3. Click the **Manage Roles** tab; explore the available roles to determine if an existing role provides the access you want to grant the user. If necessary, create a new role with the appropriate permissions.
4. Click the **Repository Security** tab.
5. In the **Repository Object** area, select a Repository item (diagram, object, submodel, or data dictionary) to which you want to grant or prohibit access.
6. Beneath **Available Users**, select a user to which you want to assign the role to the selected object and drag it onto the appropriate role in the **Available Roles** area.

   TIP: To remove the role assignment of a user to a Repository item, click the Repository item, drag the user name from under role onto another role in the Available Roles area.

Notes

- A user can only have one role for each object. But a user can have different roles for different objects.
- If a user is listed in the Available Users area, but is unselectable, the user has been deactivated and must be reactivated before you can modify the user.

Prevent Users from Accessing Specific Repository Items

The system-defined No Access role can prevent a user from accessing selected projects or diagrams

1. Log in to the Repository.
2. Choose **Repository > Security > Security Center**.
3. In the **Repository Object** area, navigate to and then click to select a project or diagram you want to secure from specific users.
4. From the list of users in the **Available Users** area, click a user name or **CTRL-click** several names and then drag the users onto the **No Access** role in the **Available Roles** area.
5. Repeat step 3 and step 4 as required to secure other diagrams and projects.
6. Click **Apply**, continue changing security settings if required, and then click **OK** to exit the security center.

Change User Access to Repository Items

To change user access to Repository items, change the permissions of the role the user has for the Repository item or assign another role to the user for that particular Repository item. If required, you can force the user log out, and then change the user's access privileges.