The Encryption Wizard for Oracle

User Manual

For 10g, 11g, 12c ,18c and 19c Oracle Databases

Version 9

Relational Database Consultants, Inc.

The Encryption Wizard for Oracle

User Manual

Relational Database Consultants, Inc. (RDC)
12021 Wilshire Blvd
Suite 108
Los Angeles, CA. 90025
310-281-1915

www.relationalwizards.com

The Encryption Wizard for Oracle

User Manual

Table of Contents

Introduction - Database Encryption

In this age value is stored within the world's information systems. Before the advent and proliferation of computers, corporate value took the form of money and physical documents. Yet just as capital seeks to automate labor – capital has automated exchange value itself, thus replacing physical money with credit and other forms of virtual money.

Today not only is money value stored within the worlds information systems, but the actual marketplace, where the velocity of capital turnover is determined, has been abstracted to a virtual space within the computer and network infrastructure of the world's information systems – thus today it becomes paramount to protect this real value, just as a bank in the past would protect its physical deposit of money and its ability to conduct business transactions.

Data Encryption and Obfuscation is the ultimate protection against the potential theft or destruction of online marketplaces where goods and services are exchanged and the actual money result of those transactions are stored. The reason being is that these forms of value and capital accumulation must be stored as exact information. If this information is obfuscated - it loses its value, until such a time that it is unobfuscated, where it again becomes a value.

To encrypt data is a magical process. It would be like a banker of old being able to transform currency into obfuscated pieces of worthless paper. Thus any thief would only be stealing the product of our forests and not much value at that. Today we possess this magical art of obfuscation and encryption - thus it is inevitable that capital will employ this practice to protect data, since data content not only determines the money result in the circuit of production and consumption, but the actual marketplace velocity itself.

The Encryption Wizard for Oracle offers users the ability to encrypt relational data - an emerging necessity in the networked global economy and among the governments that comprise the superstructure of this epoch in history. With the Encryption Wizard, data administrators can easily transform valuable corporate data into useless bytes of information – until such a time as they choose to re-establish the value within their data.

Try an experiment if you are not convinced. Create a small tablespace in your Oracle database and create a table containing some hypothetical valuable information. Now read the small tablespace in a text-editor. You will see how easy it is to obtain your corporate data. Imagine if someone had used a tool such as FTP to transfer your tablespace to their computer – would you be comfortable with the information they now had?

The Encryption Wizard will eliminate these real fears. Simply encrypt your valuable corporate data so anyone downloading our hypothetical tablespace will not possess your corporate value but will instead own useless information. Not employing Encryption techniques is akin to leaving money and valuable documents out in the open. Thus - the lack of encryption methodology for *relational* corporate data is a primary cause of the billions of dollars lost in Information Systems each year. By leaving money, credit, and valuable information unobfuscated - hacking and corporate theft is encouraged. Use the encryption techniques of the Encryption Wizard and protect the value of your corporate data at a fraction of the cost of other less-secure methods.

Chapter One – Installation

There is no database encryption tool on the market that is easier to install than the Encryption Wizard for Oracle. This is because the Encryption Wizard runs exclusively on the Oracle RDBMS and accesses all information directly within the SGA. Therefore no external libraries or OS dependent files are there to hamper your installation efforts.

The Encryption Wizard requires the following to be implemented successfully on your Oracle platform(s):

1. An Oracle database of version 10g or above with the package DBMS_Obfuscation_Toolkit currently installed.

Contact us for backward compatible Oracle 9i and 8i versions of the Encryption Wizard.

2. To run the user-interface of the Encryption Wizard you will need the Java JRE 1.4 or higher installed on your client(s). If you plan on using the Encryption Wizard API Library exclusively, you do not need Java installed on your client.

Installation of the Encryption Wizard starts by downloading the product from:

http://www.relationalwizards.com/html/oracle_encryption.html

1. Client Installation

The Encryption Wizard for Oracle download is offered as either a Windows .EXE installation file or as a .ZIP file, which can be installed on any OS file-system running Java.

Tip:

Installing the Encryption Wizard on a Windows client does not preclude its use on non-Windows Oracle servers such as Unix or Linux platforms.

The Client Installation is performed when you download the Encryption Wizard for Oracle and extract the *.exe file or the .zip file.

If you choose to run the EncryptionWizardJava.exe Windows installation file, the Encryption Wizard will create a Windows program group and install the appropriate documentation and files needed for the Oracle Server Installation.

2. Database Installation

Once you have completed the client installation for the Encryption Wizard, you are ready to begin the Oracle database installation. You can start the Encryption Wizard database installation from SQL*Plus directly by running the installation script rdc_encrypt_install.sql.

```
SQL>@rdc_encrypt_install.sql
```

Or, if you are running a Windows client with SQL*Plus, you may simply click on the Icon labeled "Database Installation" in the Encryption Wizard program group created by the Windows client installation.

During the server installation the Encryption Wizard will prompt you for these following five items:

1. The System Password – This value is not stored and is used to create the Encryption Wizard user account.

2. The SQL*Net connection string used by SQL*Plus to connect to the database where you are installing the Encryption Wizard.*

3. The new password you would like for the Encryption Wizard account: *rdc_encrypt_user.*

4. The default tablespace you would like to assign to the Encryption Wizard (Optional)

5. The Temporary tablespace you would like to assign the Encryption Wizard (Optional)

**Use may manually remove this parameter from the installation script if you do not use SQL*Net.*

Once you have supplied the Encryption Wizard installation script with these values, the script will create the Oracle account *rdc_encyrpt_user* and compile the intelligent PL/SQL packages that make up the Encryption Wizard for Oracle.

Tip:

Using Oracle's DBMS_Crypto AES Encryption: (Optional)

During installation, you will be prompted to execute the following SQL connected as SYS:

SQL>grant execute on DBMS_Crypto to RDC_Encrypt_User;

This is required for AES encryption. If you need AES encryption are not implementing DBMS_Crypto, see section four of this chapter: *Installing AES Encryption With GNU Crypto.*

Once your installation is complete, a list of valid encryption types will be displayed as part of the output. When the installation script finishes, you should receive this message:

Installation successful...

The Installation log is spooled to the file rdc_encrypt_install.log

The Encryption Wizard user *rdc_encrypt_user* is granted these following privileges by SYSTEM:

Connect	*Lock Any Table*
Resource	*Create Any Trigger*
Execute_Catalog_Role	*Alter Any Trigger*
Select_Catalog_Role	*Drop Any Trigger*
Select Any Table	*Create Any View*
Insert Any Table	*Drop Any View*
Analyze Any	*Create Any Index*
Global Query Rewrite	*Drop Any Index*
Update Any Table	*Create Public Synonym*
Delete Any Table	*Drop Public Synonym*
Alter Any Table	*Grant Any Object Privilege*
Select Any Dictionary	

3. Java UI Configuration and Client Verification

The main menu of the Encryption Wizard UI uses the Java class called EncryptionWizard.class, which exists in the EncryptionWizard installation file EncryptionWizard.jar.   To successfully run the Encryption Wizard UI, a Java JRE (1.4 or higher) must be installed on your client.

SQLNET.ORA - Java Configuration (Oracle 12c and above)

To run the Encryption Wizard (thin-client) UI, add this line to the sqlnet.ora file on the database server:

```
sqlnet.allowed_logon_version=8
```

The sqlnet.ora file is usually in the directory $ORACLE_HOME/network/admin.

To test that you have the proper Java version in your command path, issue this command from either your Windows command prompt or Unix/Linux shell:

```
>java –version
```

Make sure the version of the java runtime is version 1.4 or higher. If java cannot be recognized by the operating system, you will need to include the correct JRE in your OS path. Oracle already has the Java JRE installed by default on its databases and can be added to the path.

Setting the proper Java path for Windows Clients:

1. Navigate to Control Panel | Administrative Tools | Computer management
2. Choose: Actions | properties | Advanced | Environment Variables
3. Click on the system variable in the list called "path" and click on EDIT to modify the path to include the proper java path.

Example: (*mypath* is already defined)

```
Path = c:\mypath;C:\Program Files\s1studio_jdk\j2sdk1.4.1_02\bin
```

For Oracle users who don't want to install the SUN jdk, use the Oracle JRE 1.4. In this example ORACLE_HOME is defined as c:\oracle\ora10g:

```
Path = c:\mypath;c:\oracle\home\sample\jdk\jre\bin;
```

Once you have edited the path information and clicked OK, open a new command prompt and run the test:

java –version again.

Setting the Proper Path for Unix/Linux Clients:

After you have extracted the Encryption Wizard for Oracle .ZIP file, move all of the files up to a single Unix/Linux directory where you wish to start the Encryption Wizard.

Depending on your FTP or file transfer commands, you may need to set the EncryptionWizard.sh as an executable script as such from your shell:

```
$chmod 744 EncryptionWizard.sh
```

4. Installing AES Encryption With GPL GNU Crypto (Optional)

For Oracle licenses that do not have access to DBMS_Crypto, the Encryption Wizard now integrates with the free GPL GNU Crypto java library to offer AES256 and AES128 ciphers for all customers.

After Encryption Wizard Database Installation, follow these steps to install GPL GNU AES:

1. Upload gnu-crypto.jar and JavaEncrypt.class from the Encryption Wizard installation directory to your database server.

2. Run the Oracle command-line utility *loadjava* from the server's OS prompt:
(For Unix users make sure the ORACLE_SID is set before running *loadjava)*

```
 loadjava -u rdc_encrypt_user/<password>@sid -v -resolve -f gnu-crypto.jar
JavaEncrypt.class
```
(You may notice a few errors for unrelated libraries)

3. Issue this command from sqlplus (or other SQL tool) logged in as rdc_encrypt_user:

```
dbms_output.put_line(rdc_encrypt_util.InstallJavaAES);
```

To verify the GPL JavaEncrypt.class was loaded properly issue this query from rdc_encrypt_user:

```
Select * From    User_Objects
Where  Object_Type = 'JAVA CLASS'
And      Object_Name = 'JavaEncrypt'
And      Status = 'VALID';
```

To verify the new GPL installed AES encryption ciphers were recognized, issue this query from rdc_encrypt_user:

```
select * from valid_encryption_types;
```

The Free GNU Public License (GPL) software notice:
www.gnu.org/copyleft/gpl.html

Source code for GNU Crypto is available at:
www.gnu.org/software/gnu-crypto/#downloading

Caution:

Make sure to decrypt all GNU AES encrypted columns *before* using these commands in the future to load a new version of GNU Crypto. See the Encryption Wizard readme.txt for migration notes between versions of the Encryption Wizard.

5. Starting the Encryption Wizard User Interface

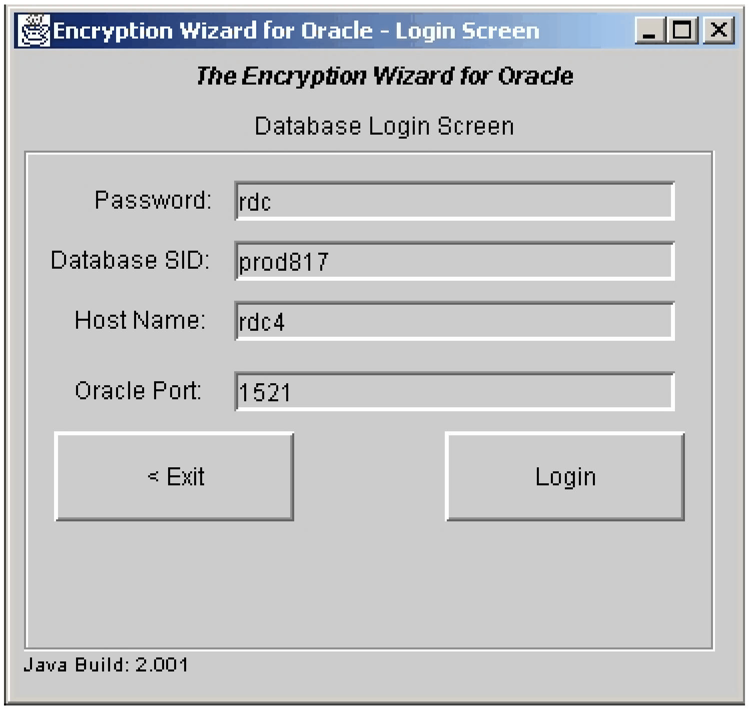
To invoke the Encryption Wizard Main Menu from Windows, simply click on the icon labeled: *Encryption Wizard User Interface*.

If you are running Unix or Linux, simple run the provided shell script that you modified in the directory containing the Encryption Wizard install:

```
$./EncryptionWizard.sh
```

After you start the User Interface, you will be prompted for your database login to the Encryption Wizard account *rdc_encrypt_user.* Make sure you remember the password that you assigned during installation step 2 and simply supply the login information as such.

In most Oracle configurations, the port is either 1521 or 1522; if this is not the case, check your tnsnames.ora file and the listener.ora file on the server to find the correct port.


Encryption Wizard for Oracle - Login Screen

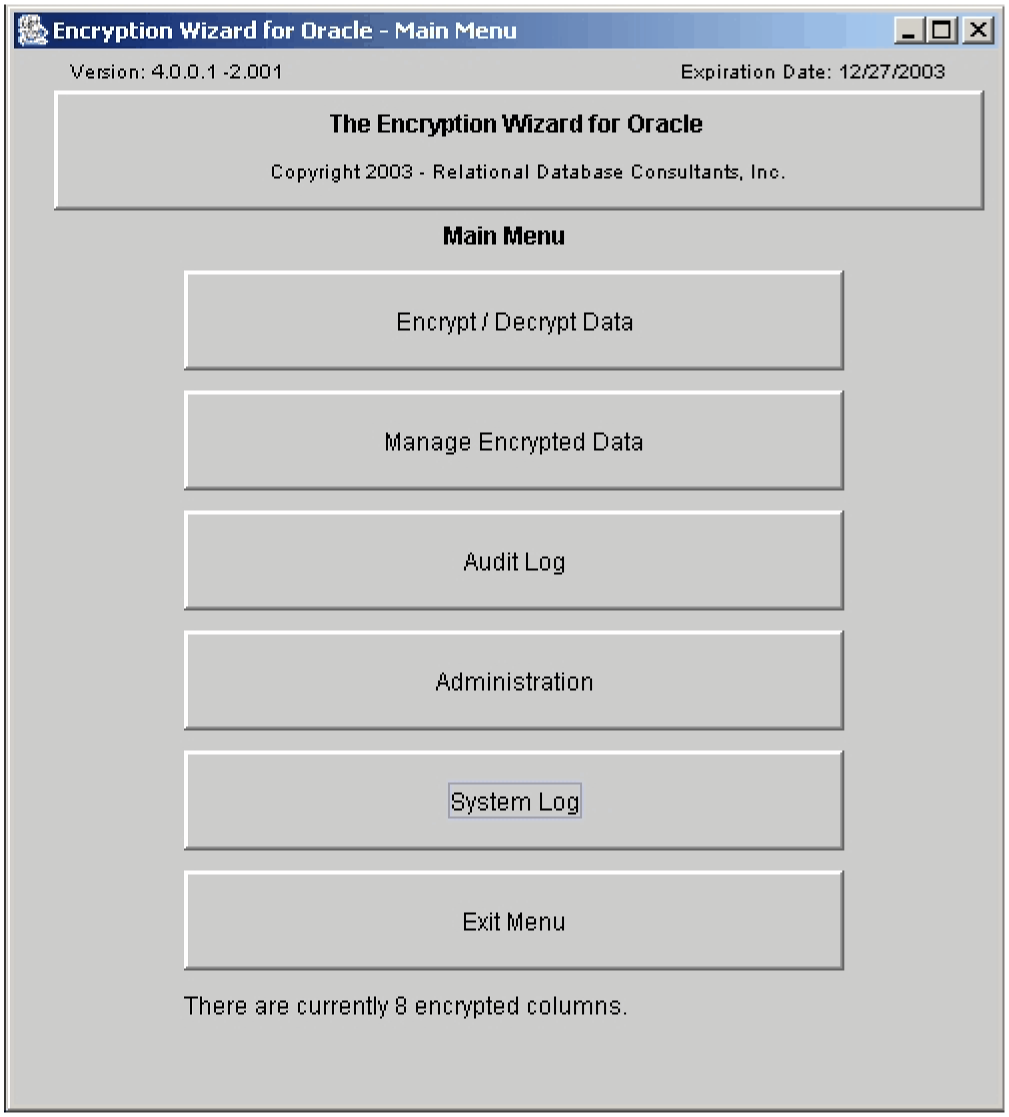
The Encryption Wizard for Oracle

Database Login Screen

Password: rdc
Database SID: prod817
Host Name: rdc4
Oracle Port: 1521

Java Build: 2.001

If your connection succeeds, then you should see the Encryption Wizard Main Menu. It is from this window that you will navigate through the various functions of the Encryption Wizard:


Encryption Wizard for Oracle - Main Menu
Version: 4.0.0.1 -2.001
Expiration Date: 12/27/2003
The Encryption Wizard for Oracle
Copyright 2003 - Relational Database Consultants, Inc.
Main Menu
Encrypt / Decrypt Data
Manage Encrypted Data
Audit Log
Administration
System Log
Exit Menu

There are currently 8 encrypted columns.

When you first enter this menu - you will be instructed to enter a demo license code which can be obtained from ewdemo@relationalwizards.com. After you receive your code, you can register the Encryption Wizard by clicking on Administration, option 4 of the main menu with the instructions provided.

Chapter Two - Encryption Wizard Overview

The Encryption Wizard for Oracle allows you to physically encrypt the data that resides within your Oracle RDBMS. You can specify this physical encryption at the schema, table, or column level.

1. Encryption Types (Ciphers)

The Encryption Wizard gives you a number of encryption ciphers to choose from to protect your data:

I. Obfuscation – Obfuscation is not technically encryption. Obfuscation simply *obscures* and makes your data apparently useless. Advanced decryption techniques can break obfuscation, yet obfuscation makes casual data theft unlikely among threats inside or outside your organization unless sophisticated and time-consuming techniques are employed to break the obfuscation keys.

II. DES Encryption – DES Encryption is the certified encryption standard provided by the Oracle Corporation through their package DBMS_Obfuscation_Toolkit. The Encryption Wizard utilizes a 64 bit key to protect your data.

III. Triple DES Encryption – Triple DES Encryption (DES3) is a response to advanced techniques used to break standard DES encrypted data. With Triple DES, a data value is encrypted recursively using three 64-bit keys to insure an almost infinite number of key combinations. Currently the Encryption Wizard uses the Triple DES scheme:

C=Ek3(Dk2(Ek1(P)))

IV. AES 128-bit Encryption

AES encryption is available to Oracle 10g users through the DBMS_Crypto package or optionally with GNU Crypto. AES encryption is recognized as more secure than DES Encryption and we have tested it as 20% faster.

AES encryption is not available as of release 5.0.0.1 for CLOB or BLOB datatypes.

V. AES 256-bit Encryption (CBC or ECB)

256-bit AES encryption uses large 32 byte encryption keys. This encryption type also is called through Oracle's certified DBMS_Crypto package and/or GNU Crypto.

AES encryption is not available as of release 5.0.0.1 for CLOB or BLOB datatypes.

VI. Triple AES 256-bit Encryption

Triple AES-256 consists of three rounds of AES-256 encryption. Each encryption round is performed by Oracle's DBMS_Crypto package and/or GNU Crypto.

VII. Blowfish 448-bit Encryption

This older cipher is available when you have loaded GNU Crypto. Blowfish uses a large 448 bit key, but is slower than the other ciphers available.

VIII. None

Choosing “None” as your encryption type allows auditing, restricted user lists, and password protection for non-encrypted columns.

The Encryption Wizard stores your encryption keys in the Oracle database as a hash value and then generates a second hash at runtime in memory using a large 2048 bit key. Encryption Keys can either be specified by the user or automatically generated by the Encryption Wizard.

For character data, salt can be specified which obfuscates identical data with a random string concatenation before encryption.

2. Key Management

I. Key Values are Stored in the Oracle RDBMS and Protected

All keys values are stored as 2048 bit raw variables within the Oracle RDBMS - the potential mathematical seed of the eventual key to be utilized. At runtime this key matrix is again encrypted and cached in user memory. This allows for a hidden mutating key strategy for the Encryption Wizard’s eventual sets of 256 bit keys. Thus, the Encryption Wizard uses an algorithm to choose mutating 256 bit subsets of a 2048 bit key per encryption round which is AES protected at runtime using Oracle’s certified DBMS_Crypto package.

II. User-Defined Keys

All user-defined keys, or pass-phrases, are expanded to the 256 byte key value and obfuscated using AES256 and then again obfuscated at runtime. Yet with a user-defined key, recovery is possible if the key is lost. This recovery can only be performed using the Encryption Wizard API.

III. Key Backup

The Encryption Wizard for Oracle now offers complete or partial key backups to a flat-file. This will allow users to backup database encryption keys to removable disks and/or use removable disks as a requirement to read encrypted data.

The encryption wizard also allows you to password protect your backups and insures that backups from one Oracle database cannot be loaded into another database. Because the Encryption Wizard performs another encryption round on these stored keys at runtime, it is not necessary to encrypt the backup file - unless you want another level of protection.

The Encryption Wizard always generates one unique key per database column. This makes unauthorized decryption much more difficult for large data sets containing many columns.

3. Data Types to be Encrypted

The Encryption Wizard for Oracle allows you to encrypt these basic types of data:

Character Data

Varchar2 and Character data types can be encrypted using any of the three above methods. If DES, AES-128 bit or AES-256 bit encryption is utilized, the Encryption Wizard will modify the column lengths of your character data upwards to the nearest multiple of 8, 16 and 32 characters respectively.

Number and Date Data

Date and Number data can only be obfuscated. Large floating point numbers over 38 places are rounded in accordance with Oracle's *to_number* function default settings.

Large Object Binaries

The Encryption Wizard is able to encrypt both BLOB and CLOB Oracle data types. There are no size restrictions, yet large objects take considerable time to encrypt and decrypt. Be sure to test the time of encryption before attempting to encrypt a large table of BLOB or CLOB data. CLOB data is trimmed of trailing blanks.

Null Data

The Encryption Wizard can be instructed to encrypt null data. With a call to the API Library you can set the Encryption Wizard to encrypt null values for *varchar2*, *char*, and *nchar* datatypes. For instruction on enabling or disabling this feature, please refer to chapter seven, section 4 of this document.

The Encryption Wizard will not encrypt Primary, Unique, or Foreign Keys, nor does it encrypt columns with default values or condition constraints, aside from the popular "Not Null" constraint. The Encryption Wizard default settings also prohibit encryption of Oracle compressed tables. Write us at ewdemo@relationalwizards.com for instructions on how to enable this feature.

4. Transparent Decrypted Views

To allow for applications to access physically encrypted data, the Encryption Wizard Administrator can optionally create *decrypted views* against any table with encrypted data. Decrypted Views display data in unencrypted format, and thus allow applications to seamlessly read and/or write to the decrypted data objects through the use of an automatically generated instead-of database trigger created for each view. Decrypted views can be dynamically created and dropped at any time through the Encryption Wizard User Interface or the Encryption Wizard API.

Typically, the application base table is renamed and then encrypted. After this, a decrypted view is created with the original table name. This allows for transparent access to the same database object by applications.

5. Function Indexes

To allow for fast access of decrypted views, the Encryption Wizard now offers point and click function index creation to speed up access to your encrypted data. Function Index data is stored in Reverse/Compress format to minimize data search time in the index tablespace.

6. Session Auditing

The Encryption Wizard also allows the user to specify session auditing. Session auditing, will record all distinct Encryption and Decryption (read/write) requests for all sessions. Session Auditing allows you to see who has had access to your encrypted data down to the Terminal ID and database column level. The Encryption Wizard also supplies you with management reports to trace user activity against your encrypted data.

7. Restricted User Lists and Runtime Passwords

A Restricted User List allows you to specify which users you wish to grant the ability to read and/or write encrypted data. You can specify user lists for a given schema, table, or column. If there is no user list specified, then a user's ability to access encrypted data is based on traditional Oracle grants.

Runtime passwords protect your data by requiring all users to login and supply a password before viewing encrypted data objects. The actual login method is a PL/SQL call to the Encryption Wizard API and can be embedded in any SQL script or application. A runtime password can be assigned for each column, table, or schema depending on your preference.

You may also add password protection for any user defined in a restricted user list. This requires the user to "*login"* with the password before they can view or change the encrypted data domain.

Restricted User Lists and Runtime Passwords allow you to block out any Oracle user from viewing your encrypted data, even a DBA user such as SYS. To block DBA access to data, use Restricted User Lists and / or Runtime Passwords in conjunction with the Administrative Password which is discussed below.

8. Administrative Password

To prevent Database Administrators unauthorized access to the Encryption Wizard, the Encryption Wizard Administrator can set an optional password required to Encrypt and Decrypt data and use the Encryption Wizard interface. Likewise, the Encryption Wizard administrator does not require DBA privileges to use the Encryption Wizard.

9. Data Consistency

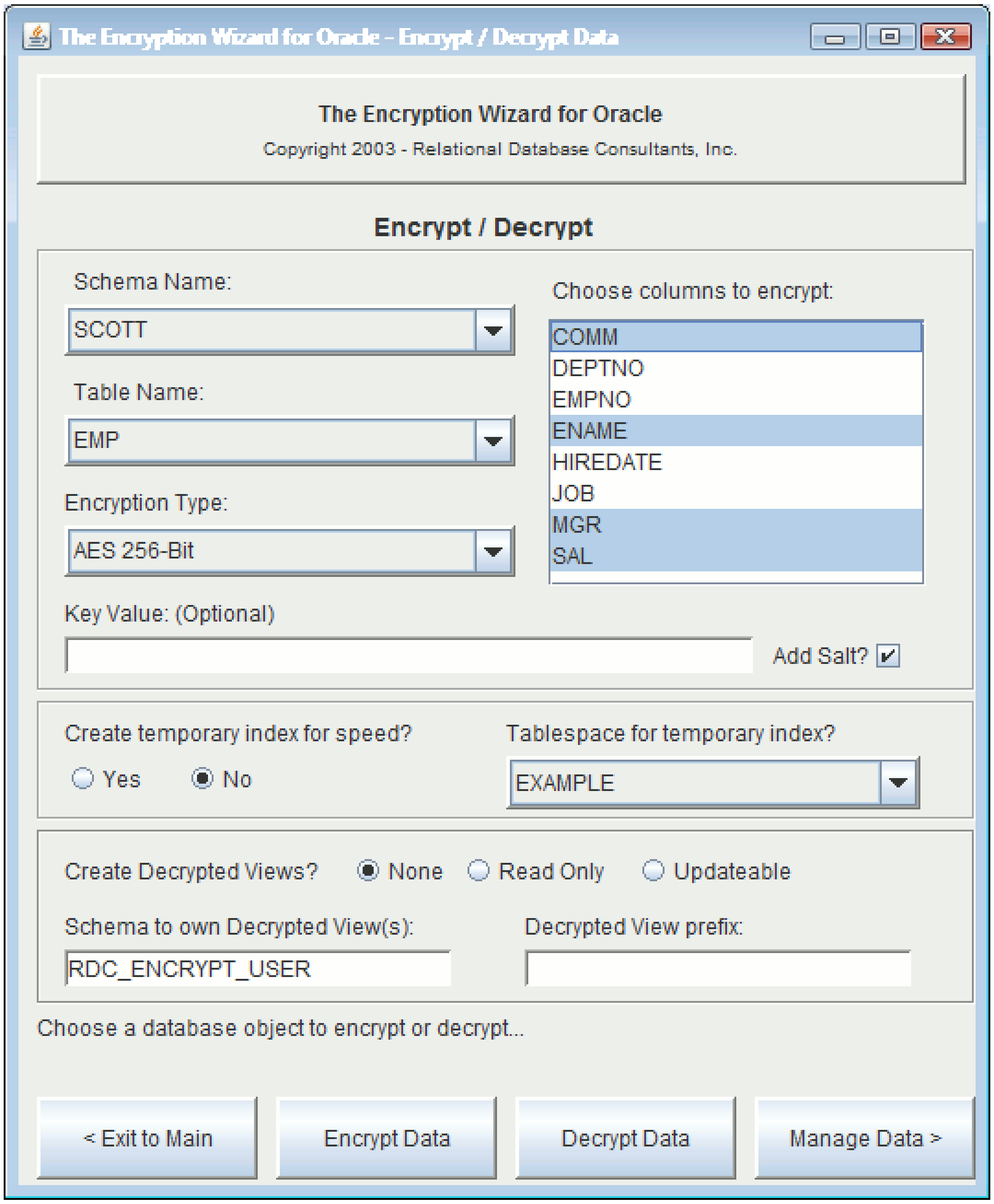
The Encryption Wizard also employs intelligent recovery operations if any Encryption or Decryption attempt fails. The Encryption Wizard will self-diagnose any incomplete Encryption or Decryption operation and allow the administrator to simply continue the process or back out. This helps guard against serious data inconsistency due to partially completed operations that may occur due to an unexpected database event like a shutdown.

10. Key Backup & Recovery for Continuity Planning

The Encryption Wizard allows users to backup encryption keys to a flat-file. These files may be password protected and cannot be used on any other database than from where they were created. This enhances the security of your key backup and recovery operations.

Chapter Three – Encrypting Table Data

To Encrypt Data within your Oracle RDBMS, click on the top-most option of the Encryption Wizard Main Menu labeled*: Encrypt/Decrypt Data*. At this point you will enter the Encrypt / Decrypt Data Screen:


The Encryption Wizard for Oracle - Encrypt / Decrypt Data

The Encryption Wizard for Oracle

Copyright 2003 - Relational Database Consultants, Inc.

Encrypt / Decrypt

Schema Name: SCOTT

Table Name: EMP

Encryption Type: AES 256-Bit

Choose columns to encrypt:
- [✓ COMM]
- DEPTNO
- EMPNO
- [✓ ENAME]
- HIREDATE
- JOB
- [✓ MGR]
- [✓ SAL]

Key Value: (Optional)

Add Salt? ✓

Create temporary index for speed? Yes [✓ No]

Tablespace for temporary index? EXAMPLE

Create Decrypted Views? [✓ None] Read Only Updateable

Schema to own Decrypted View(s): RDC_ENCRYPT_USER

Decrypted View prefix:

Choose a database object to encrypt or decrypt...

Step One - Choosing a Data Set for Encryption

From the Encryption / Decryption Screen we can encrypt database data at the schema, table, or column level. These three levels define our data domain for encryption:

Schema Level Encryption

Schema Level Encryption is the most powerful of the three levels of Encryption. When using Schema Level Encryption, all one must do is specify a schema by using the uppermost pick-list that appears on the Encryption / Decryption screen. Simply leave the Table Name list blank.

When we choose Schema Level Encryption, we are instructing the Encryption Wizard to encrypt all valid columns for that schema. Encryption of this type should not be performed until you are comfortable using the Encryption Wizard.

Note:

When you choose Schema or Table Encryption, all valid date and number fields are obfuscated, regardless of the encryption type you request.

Schema encryption does not mean that your complete schema will always be encrypted. If new tables are added to your schema, the Encryption Wizard will not be aware of them unless you choose schema encryption again to protect the remaining tables.

Table Level Encryption

When you choose to encrypt a single table without selecting any columns to your right, the Encryption Wizard will, of course, attempt to encrypt every valid column within that table. As discussed in the previous chapter, columns that are primary keys, unique keys, foreign keys, default values, and conditional constraints are skipped.

When employing Table Encryption, specify a schema and table name by using the two uppermost pick list that appears on the Encryption / Decryption screen. Choose both the schema owner and table name of the object that you want encrypted. After table encryption, all columns that appear on the right of the screen will be encrypted.

Column Level Encryption

Column Level Encryption is the fine-grain level of encryption for the Encryption Wizard. With column encryption you specify and encrypt distinct column(s) only.

To define encryption at this level, first chose the table owner and table name from the two uppermost pick-lists that appear at the top of the Encryption / Decryption Screen. Once you have chosen the appropriate table to encrypt, click on those column(s) you wish to encrypt that appear in the column list on the right of screen. To choose multiple columns, press down on the CRTL key as you click with your mouse.

We recommend that the Encryption Wizard Administrator first employ Column Encryption before attempting table or schema encryption.

Tip:

Hold-down the CTRL-key on your keyboard as you right-click on the mouse to select multiple columns or to unselect a column.

Step Two - Choosing an Encryption Type

After we have defined our level of encryption, we must now instruct the Encryption Wizard as to what type of encryption to perform on our given data set. For a discussion of the various encryption types please refer to Chapter 2 - Encryption Wizard Overview. To choose an encryption type, simply use the pick-list labeled: *Encryption Type*.

Warning: *If you chose AES encryption for a Blob or Clob data type, the data will be encrypted using DES3 technology. AES encryption is not yet available.*

Optionally, you may check the flag *Add Salt*. Salt makes detection of identical data more difficult by adding random bytes to data. Leaving this box unchecked will save the data storage space used to add salt.

Step Three - Key Generation

After we have chosen our encryption type, we may optionally enter a key for encryption. A key is like a password - and can be used in recovery of data if your *rdc_encrypt_user* schema becomes corrupt. You can also choose to have your key automatically generated; inn this case simply skip this step.

Step Four – Specify a Temporary Index for Speed (Optional)

If you are encrypting a very large table over 10 million rows, you may choose to have the Encryption Wizard build a Temporary Binary Index for performance during encryption. Only use this option if encryption proves to be too slow for a given database object.

Step Five – Transparent Decrypted View Creation (Optional)

A *Decrypted View* is a duplicate projection of any table that the Encryption Wizard partially or completely encrypts. Decrypted Views allow users to transparently perform SQL operations to read and/or write data to the encrypted table. Data *appears* in its original form when using a Decrypted View.

If you plan on migrating applications or SQL statements to your new encrypted data sets, using Decrypted Views is the easiest option. Refer to chapter 6 – Migrating Applications to Encrypted Data Tables for more discussion of this topic.

Tip:

If you haven't thought about a decrypted view strategy yet, simply skip this step and create your decrypted views later in the Manage Encrypted Data screen, option 2 from the main menu.

If you specify *read-only* Decrypted Views, the Encryption Wizard will create a decrypted view that gives authorized users a transparent view of your encrypted data, but will not allow them to update, delete, or insert rows using that decrypted view.

If instead, you select *updateable* decrypted views, the Encryption Wizard will create decrypted views through which authorized users may insert, update, or delete encrypted records via *instead of triggers* automatically generated by the Encryption Wizard supporting the decrypted view.

After specifying which type of decrypted view(s) you would like the Encryption Wizard to create, we need to choose a schema for that view and, optionally, a prefix for the decrypted view. By default, all decrypted views are created in the Encryption Wizard schema, *rdc_encrypt_user.* You may instead assign the schema owner of your data or another schema, in this case SCOTT, as the owner of your decrypted view(s). Simply change the current schema owner name in the field under the label: *Schema to Own Decrypted Views* to the owner of the table(s) you are about to encrypt.

By default, decrypted view will have the same name as their base table, unless they are created in the schema of the base table. In this case, the base table can be renamed prior, to allow for the decrypted view to have the base table name. Strategies for decrypted view creation are discussed in length in Chapter 6 – Migrating Applications to Encrypted Data.

If we choose to create our decrypted view in the same schema as our encrypted table we must first assign a prefix to it - so the view name will not conflict with the base table name. Later we can rename our decrypted view to any valid object name using the View Management screen.

To enter a decrypted view prefix, enter a short prefix in the field under the label: *decrypted view prefix*.

Example:

If the table we are encrypting is *scott.emp,* we might enter a decrypted view prefix like: V_. In this case our decrypted view would have the name: *v_emp* and would exist in whatever schema we choose for it. The decrypted view prefix is optional if you are creating the view in a separate schema from the base table.

Caution:

If you are creating a decrypted view in another schema, make sure grant select,update,delete, insert privileges on your table(s) to that schema.

Note:

When encrypting an entire schema, the decrypted view prefix will be applied to all tables.

If you are not sure of a decrypted view naming strategy - don't be concerned. The Encryption Wizard allows you to drop, rename, or recreate any decrypted view dynamically using the Encrypted Data Management Screen discussed in Chapter Five – Managing & Auditing Encrypted Data.

Step Six - Encrypting Data

After we have specified the above options, we are ready to encrypt our data. Encrypting data is a very disk and CPU intensive task. When you encrypt data, all rows of the specified data set are encrypted. Encryption also involves key-generation activities that are very CPU intensive. Thus it is ideal to start encryption during off-hours. Before encrypting data make sure no users are modifying the data tables being encrypted.

Caution:

Always back-up your data before encryption!

Note:

If, for some reason, you need access to your encrypted data table while the Encryption Wizard is encrypting the data, you may delete records and/or insert records into your partially encrypted tables without any data loss. If you encounter a lock-contention error, these errors do not affect the integrity of the encrypted data set and can be corrected by restarting the encryption process again. Ideally, one should not attempt this on a regular basis.

When you are ready to encrypt your data simply click on the button at the lower part of the Encryption / Decryption Screen labeled: *Encrypt Object.*

At this point, the Java interface will lose control of the screen and the Encryption Wizard will encrypt, table by table, the data set you specified. At the end of the encryption round a message will appear giving a total number of rows that were successfully encrypted.

Once the Encryption Wizard notifies you that your tables and/or columns have been encrypted, it is vital that you now point all users and applications to the decrypted views and not to the original encrypted tables.

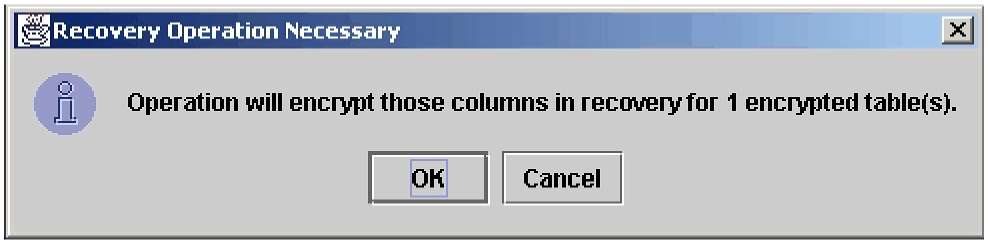
Encryption Recovery

If something happens during the encryption process, such as your database shuts down, or another user exclusively locks the table being encrypted, the Encryption Wizard provides simple recovery from these potential disasters.

In these above cases, if the Encryption Wizard is interrupted, your data will be in an intermediate state. Some of the rows will be encrypted and some will not be. Thus it is important to perform *encryption recovery* if your encryption process is interrupted.

To perform encryption recovery, simply pick again the schema, table, and/or column which you originally encrypted that is in need of recovery. To recover this object, you can either choose decryption or encryption. Decryption will back-out your partially encrypted table to its original form. Encryption will complete the encryption of the partially encrypted table.

Whenever you choose to encrypt or decrypt a schema, table, or column that needs recovery, the Encryption Wizard will perform the recovery task by notifying you with the following message:



Operation will encrypt those columns in recovery for 1 encrypted table(s).

You can always view which data objects need recovery by using the Encrypted Object Management screen discussed in [Chapter 5 – Managing & Auditing Encrypted Data]().

Note:

If only some of your columns need recovery for a given table, you may need to encrypt or decrypt the remaining columns after recover since recovery operations only modify columns that need recovery. You can check which columns require recovery from the Encrypted Data Management Screen, Option 2 from the main menu.

Chapter Four – Decrypting Table Data

Decrypting table data using the Encryption Wizard is a simple process. When we decrypt data we return it to its original readable form. To perform this task, simply enter the Encryption / Decryption Screen of the Encryption Wizard from the Main Menu and choose the schema, table, or column object you wish to decrypt.

After choosing your decryption domain, simply click on the button labeled *Decrypt Object* to decrypt the data. All other information on the screen will be ignored by the Encryption Wizard since it is already stored, along with the key, for any given encrypted column.

Note:

If you need to decrypt data quickly, then use Schema Level Decryption. Decrypting a schema will decrypt all tables and columns that you previously encrypted within that schema.

If there is a problem during decryption, such as a database shutdown or lock contention error, simply click on the *Decrypt Object* button again to complete the decryption – or click on the *Encrypt Object* button to back out and re-encrypt the complete table.

If you are decrypting a table in partial recovery, you may have to decrypt twice if some of the columns are not in recovery. The reason for this is that recovery only repairs inconsistent columns that have been partially encrypted or decrypted. To view the recovery status of any encrypted database object, simply choose option 2 from the Main Menu.

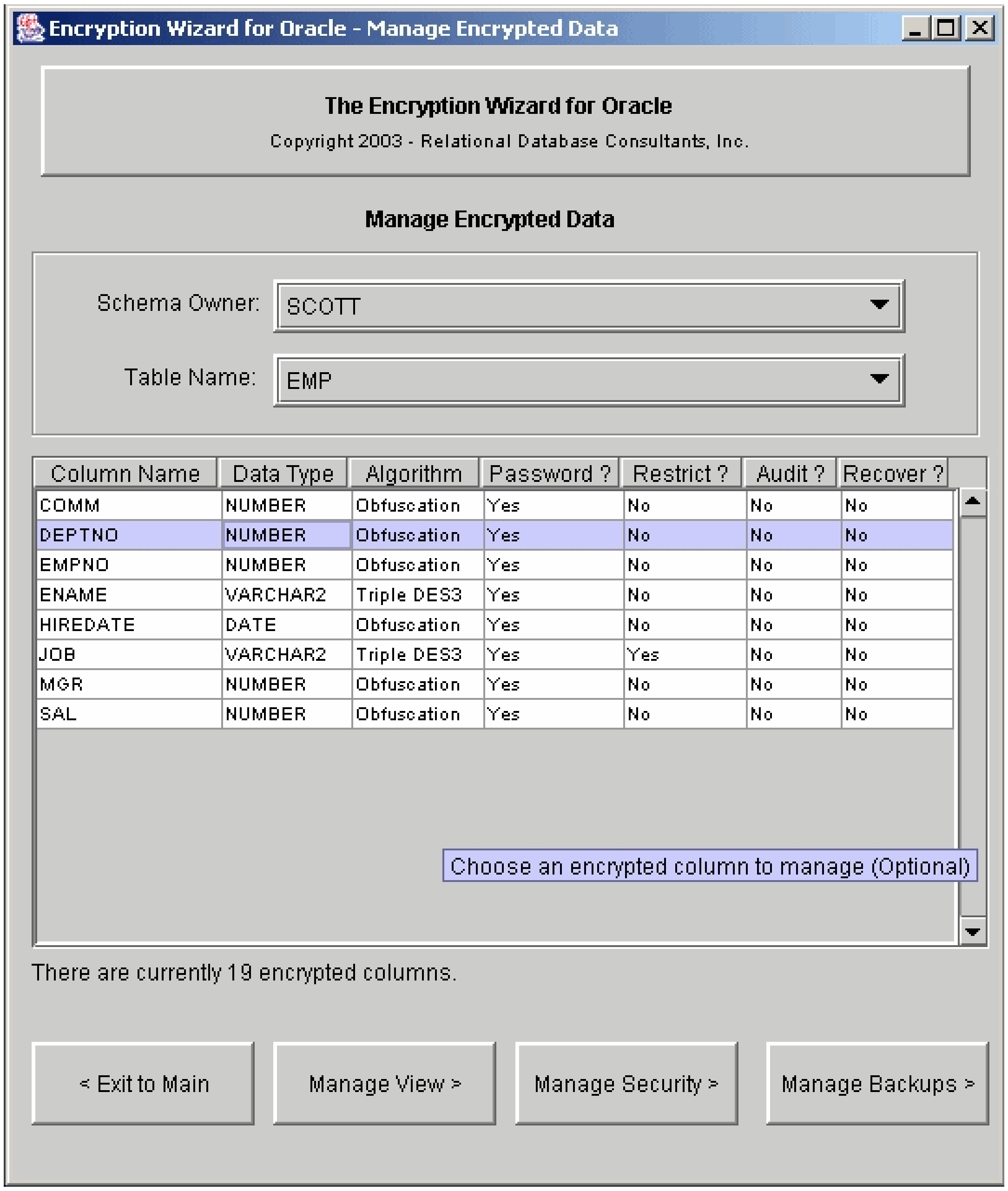
Decryption of a complete table will also drop any decrypted view created for that table.

Warning:

If you are a demo version user of the Encryption Wizard, decryption will not operate after your trial license expiration date – this is because, by the terms of the Encryption Wizard License Agreement, demo users can only encrypt and decrypt worthless data of no value.

Chapter Five – Managing Encrypted Data

The real value of the Encryption Wizard for Oracle is its easy-to-use and well-organized management features. To enter the Manage Encrypted Data Screen, simply click on the button on the main menu labeled: *Manage Encrypted Data.*


Encryption Wizard for Oracle - Manage Encrypted Data
The Encryption Wizard for Oracle
Copyright 2003 - Relational Database Consultants, Inc.
Manage Encrypted Data
< Exit to Main
Manage View >
Manage Security >
Manage Backups >

Schema Owner: SCOTT

Table Name: EMP

Column Name	Data Type	Algorithm	Password ?	Restrict ?	Audit ?	Recover ?
COMM	NUMBER	Obfuscation	Yes	No	No	No
DEPTNO	NUMBER	Obfuscation	Yes	No	No	No
EMPNO	NUMBER	Obfuscation	Yes	No	No	No
ENAME	VARCHAR2	Triple DES3	Yes	No	No	No
HIREDATE	DATE	Obfuscation	Yes	No	No	No
JOB	VARCHAR2	Triple DES3	Yes	Yes	No	No
MGR	NUMBER	Obfuscation	Yes	No	No	No
SAL	NUMBER	Obfuscation	Yes	No	No	No

Choose an encrypted column to manage (Optional)

There are currently 19 encrypted columns.

Once we enter this screen, we notice that the Manage Encrypted Data Screen has a similar format to the Encryption / Decryption Screen. You may manage encrypted data at the schema, table, or column level. To manage a given encrypted table - all we need to do is optionally select the schema owner and table name of the given object.
After choosing a table name from the two pick-list, the Encryption Wizard will show us the encryption details for each column of our encrypted table. To highlight a column, simply click on

the column name with your mouse. Use the CTRL key with the mouse to un-select a given column.

From the Encrypted Data Management Screen we can view the following information for each encrypted column. Each database column has six parameters listed with the *column name*; these values give us an overview of our encryption efforts for any particular column.

1. **Data Type**

   This displays the data type of our database column.

2. **Algorithm**

   This field records the type of encryption used to protect a given column.

3. **Password ?**

   If this value is set to “Yes” - a *Default Runtime Password* has been assigned to a particular database column. Users viewing this data will need to *authenticate* the password before performing encryption or decryption operations. We will learn how to set Runtime Passwords in the Security Manager.

4. **Restricted ?**

   If this value is set to “Yes”, encryption and decryption of this column is restricted to a specific set of users as defined in the Security Manager. These lists of users are called “Restricted User Lists”.

5. **Audit ?**

   If this value is set to “Yes”, users performing encryption and/or decryption activities will be audited. Auditing is also defined in the Security Manager.

6. **Recover ?**

   If this value is set to “Yes”, the table or column in question is only partially encrypted. You should perform encryption, decryption or re-keying of this object again to back-out of the partially encrypted state. Usually this value is set to “No”.

Security Manager Overview

From the Manage Encrypted Data Screen we can enter the Security Manager by clicking on the button labeled “Manage Security”.

To enter the security manager, simply click on the button labeled ‘Security Manager from the Manage Encrypted Data screen. You may define security setting at the schema, table or column scope:

Schema Level Scope

When you define security at this level, the Encryption Wizard will mark every current encrypted column for a given schema with the security specifications you choose. All previous security specifications for tables or columns in the schema are overwritten.

To enter the Security Manager at this level, simply choose a schema name from the pop-up list at the top of screen, leave table name blank – then click on the button labeled “Manage Security”.

Table Level Scope

Table level scope will define security for all encrypted columns within a given table. All previous security parameters set at the column level for the chosen table will be overwritten. Make sure you have chosen a table name from the pick-list before you choose this option.

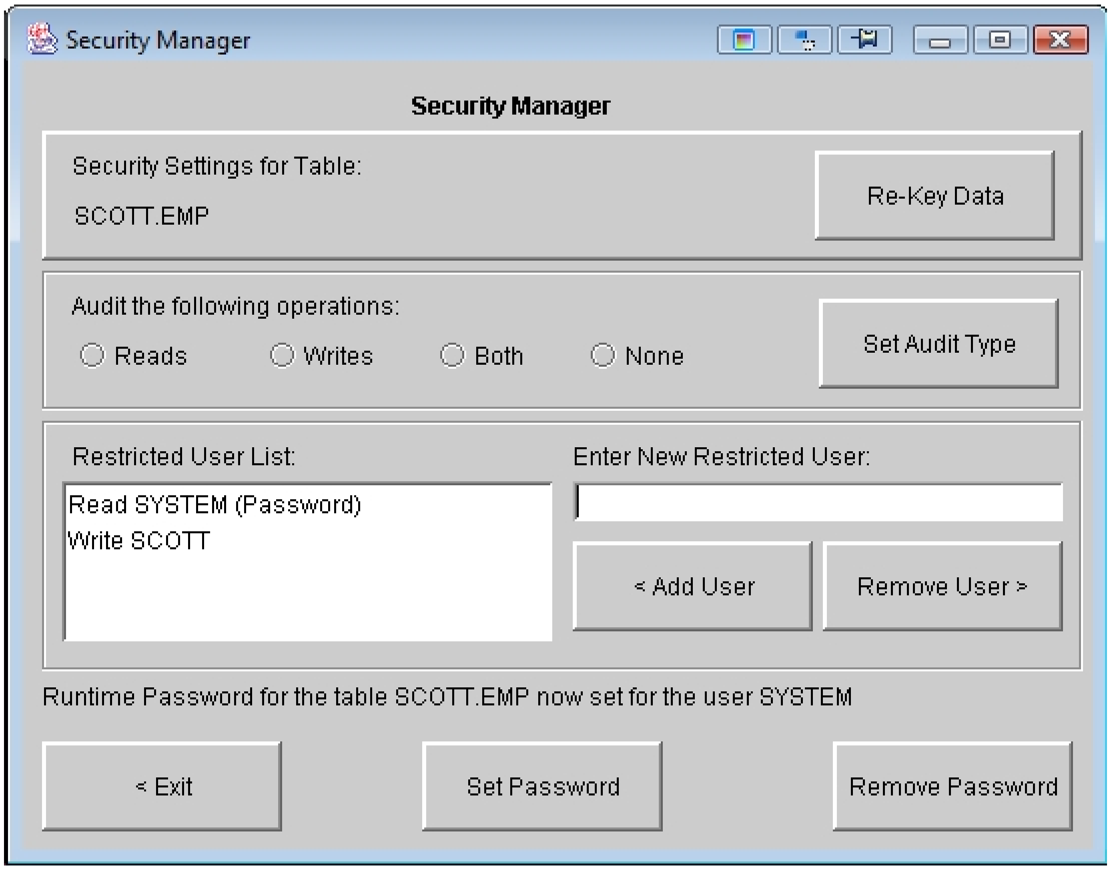
To enter the Security Manager at this level, simply choose a schema name and a table name from the pop-up list at the top of screen – then click on the button labeled “Manage Security”.

Column Level Scope

Column level scope will define security for a single encrypted column. Be sure to chose a schema owner and table name and then click on the given column’s row before clicking on the label “Manage Security”.

Security Manger Screen

Once you have chosen your Security Scope, click on the Manage Security button from the management screen to bring up the Security Manager:



Security Settings for Table:
SCOTT.EMP

Audit the following operations:
( ) Reads ( ) Writes ( ) Both ( ) None

Restricted User List:
Read SYSTEM (Password)
Write SCOTT

Enter New Restricted User:

Runtime Password for the table SCOTT.EMP now set for the user SYSTEM

The Encryption Wizard's security manager allows you to perform three basic security tasks. At the top of the screen you can re-key data. Below re-keying, auditing can be set or unset for any encrypted object and below auditing, you can set up restricted user lists and/or runtime passwords required to authenticate access for the chosen schema, table, or column.

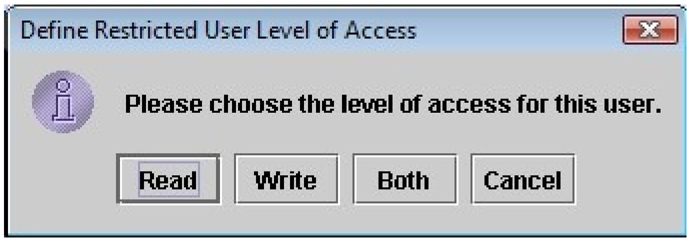
1. Restricted User Lists for Secure Access

The Encryption Wizard allows you to define *restricted user lists* for any given schema, table, or column. A restricted user list instructs the Encryption Wizard to only grant those database accounts listed the ability to encrypt and/or decrypt data. You may also specify a *Runtime Password* that these users must supply before they can view encrypted data. This feature used in conjunction with the Administrative Password discussed in Chapter Seven, allows you to protect encrypted data even from a user with full DBA privileges.

In the above example, we have restricted access to our chosen data object: *Scott.Emp*. Only the users *System* and *Scott* are able to read (decrypt) the data of this column. Not even the user SYS will be able to decrypt this data or read the *decrypted view* generated by the Encryption

Wizard, instead SYS will receive a fatal exception when attempting data decryption or encryption; furthermore an auditing record will record the failed transaction.

To add users to a Restricted User List - simply type in a valid Oracle account in the field labeled 'Enter new restricted user'. After this, click on the button labeled 'Add User'. At this point you will be prompted as to what type of restricted access you would like to grant:



Please choose the level of access for this user.

Read: Select (Read) operations on object are allowed. Update, Insert denied.

Write: Update, Insert (Write) operations are allowed. Select denied. This option cannot be used in conjunction with decrypted views because instead-of triggers must populate with decrypted data.

Both: Select (Read) and update, insert (Write) operations on object are allowed.

To remove a user from a Restricted User List, simply choose a user name from the Restricted User List and click on the button labeled: 'Remove User'.

2. Runtime Passwords for User Authentication

Runtime Passwords add a further layer of protection for your encrypted data. Runtime Passwords require users to validate a password before they can gain encryption and/or decryption privileges for a given schema, table, or column. There are two types of Runtime Passwords:

1. Default Runtime Passwords

Default Runtime Passwords are assigned to a given database object, such as a schema, table or column. After the Default Runtime Password is defined, *all* users must authenticate the password before they can have access to the given database object protected.

To assign a Default Runtime Password, enter the Security Manager for a given schema, table, or column scope. Once you have done this simply click on the button labeled "Set Password" at the bottom of the Security Manager screen. At this point, you may enter a default password of up to 30 characters.

To check if a given column or set of columns for a table have a Default Runtime Password, query the Manage Encrypted Data Screen for the table in question and look under the heading labeled "Password?". If there is a Default Runtime Password, that field will be labeled "Yes".

2. User-Specific Runtime Passwords

User-specific runtime passwords are assigned on databases objects to users on a *restricted user list*. User-Specific Runtime Passwords, will take precedence over any Default Runtime Password assigned for a given database object.

Once you have a Restricted User List created, from the Security Manager Screen, simply click on the user you wish to define a user-specif password for and then click on the button below labeled: “Set Password”. Passwords are always case-sensitive.

If you wish to periodically change the Runtime Password for a selected object or user, simply click on “Set Password” again and supply a new password. To remove a password for a given user simply choose the user from the restricted user list and then click on the button labeled: “Remove Password”.

Password protection insures that a malicious individual with DBA privileges cannot access any Oracle account and view encrypted data. Use this in conjunction with the Administrative Password to protect your encrypted data from malicious hacks into your Oracle RDBMS.

In the previous example we have set a *runtime password* for the user SYSTEM. This requires the SYSTEM user to login to Encryption Wizard runtime library for authentication before the user can view or change encrypted data. In this case SYSTEM would need to login to the Encryption Wizard as such before they could view the table SCOTT.EMP:

SQL>exec **rdc_encrypt_runtime.login**(‘TIGER’,’SCOTT’,’EMP’);

See the Encryption Wizard API Reference Manual for further details on the runtime login method that is used to authenticate restricted users.

3. Session Auditing for Security History

Session Auditing records all attempts at encryption or decryption of your protected data. This information is valuable for reporting and/or tracing any unwanted access to encrypted data.

If a user selects 1000 decrypted rows of sensitive data from an encrypted table, the Encryption Wizard will simply add one auditing record per column to report that a given user has *decrypted* that given column.

There are the three types of auditing you can chose for a schema, table, or column:

Audit Encryption

The Encryption of new data occurs when a user inserts or updates an encrypted column using either a Decrypted View or the Encryption Wizard API Library. When auditing is set to this type, updates and inserts against encrypted columns are tracked for all sessions.

Audit Decryption

Decryption occurs whenever a user selects your decrypted data, either through issuing Select statements against a decrypted view or directly decrypting data through the Encryption Wizard API Library.

Audit Both

This will employ both of the above auditing types for the columns effected.

Use the Session Auditing radio buttons to set the auditing type of the object you wish to track to *Audit Encryption*, *Audit Decryption*, or *Audit Both*. After you have changed the radio button settings, click on the button labeled 'Set Auditing'. The new settings will go into affect for all new sessions that access the encrypted data specified in your auditing scope.

4. Re-Keying of Data for Secure Keys

Many encryption standards such as PCI compliance, now require the definition of a key lifecycle. The concept being, that using the same encryption key on your database column for years increases the possibility of long-term attempts to find patterns in the encrypted data for the purpose of data-theft.

Although the Encryption Wizard for Oracle further obfuscates keys at runtime, re-keying is the best practice for a highly secure environment. For instance, if re-keying is done once a week, a data thief working to crack a key will have only a week to accomplish the reverse cryptography involved.

When data is re-keyed, all the encrypted rows for a given database object (schema, table, or column) are updated with a new encryption key, thus this process will take approximately a little longer than the initial encryption of the same object.

To re-key data, enter the *Security Manager Screen*, by choosing a schema and then optionally a table and column for re-keying. This is the *scope of re-keying* you are choosing as discussed in the overview for this chapter above.

Once you have chosen a schema, table or column to re-key and have navigated to the security manager, simply click on the re-key data button at the top right of the window. Screen control will be lost, until the object(s) you have specified are re-keyed.

Caution:

Make sure no users are updating the data that you have defined for re-keying. Users will be locked out of encryption/decryption operations while each table is being re-keyed. Users must reconnect to the Oracle database after re-keying to avoid changing data with old buffered keys.

During a re-keying event, the Encryption Wizard will store the new and old key value seeds in the system log (Internal_Audit). Please make sure to verify the re-keyed data with a Decrypted View or an API call before clearing the System Log after a re-keying event.

Re-key Recovery

If for some reason, during a large re-keying operation, the database crashes or a user obtains a lock on a table being re-keyed, simply enter the re-keying screen later and re-key the object in question again. The Encryption Wizard will do a partial re-keying in this case, updating only those rows that were not re-keyed before the event.

Table encryption and decryption will not be allowed for partially re-keyed tables and other users will not be able to encrypt/decrypt data. Partially re-keyed tables will show up as needing recovery in the main listing of the *Manage Encrypted Data* screen as shown above.

5. Default Masks for Error Handling

In many cases, applications need to query and modify table data with encrypted columns. For instance, a utility company may employ data entry clerks to update customer data – yet may not want them to view private information. Using the Encryption Wizard, we accomplish this division of tasks by using a default mask.

With a default mask, an unauthorized user, querying a *decrypted view* of an encrypted base table, will not receive an Oracle error; instead a constant string will be returned, *masking* the actual data. For instance, we could return this default mask for a customer bank account number:

XXX-XXX-XXX

Using a default mask allows end-users and applications to work around encrypted data, and work with other columns within sensitive encrypted tables.

Without a default mask, a query by an unauthorized user, results in the Encryption Wizard returning an Oracle error and auditing the event. With a default mask, an end-user simply is returned a meaningless string.

To set a default mask, call this Encryption Wizard API procedure from SQL*Plus:

```
SQL>exec RDC_Encrypt_Object.SetDefaultMask('???', 'SCOTT', 'EMP', 'ENAME');
```

This will set a default mask for the encrypted column ENAME that will return '???' to all unauthorized users attempting decryption. You can also set a default mask to simply return a NULL value:

```
SQL>exec RDC_Encrypt_Object.SetDefaultMask(Null, 'SCOTT', 'EMP', 'ENAME');
```

Encrypted number columns can also have a numeric default mask:

```
SQL>exec RDC_Encrypt_Object.SetDefaultMask(0, 'SCOTT', 'EMP', 'SAL');
```

Dates can also have a default mask of the type date, either a date constant or the current SYSDATE can be returned as the mask:

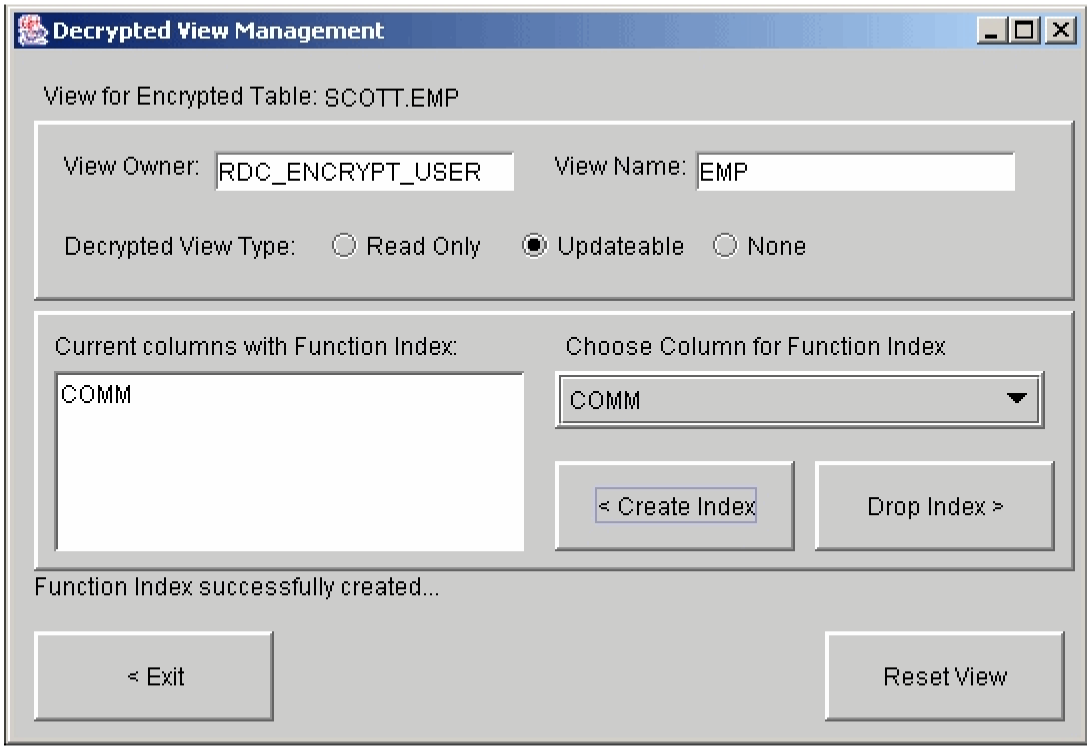
```
SQL>exec RDC_Encrypt_Object.SetDefaultMask('SYSDATE', 'SCOTT', 'EMP', 'HIREDATE');
```

If you are not using *restricted user lists* or *runtime passwords*, you do not need default masks. Users will be authorized to view decrypted data based on Oracle privileges only. For more on setting, changing, and removing default masks see Section One of the Encryption Wizard API Reference Guide.

Decrypted Views for Application Transparency

Decrypted views are objects that *appear* as regular tables with unencrypted data to applications or users with the appropriate privileges. We can create decrypted views in any schema with the appropriate privileges to view the underlying encrypted base table. This will allow applications with the appropriate privileges to view or change encrypted data.

From the Manage Encrypted Data Screen, we can create, modify, or drop a Decrypted View by first choosing a base table of your decrypted view, and then clicking on the button labeled *Manage View*. At this point, you will enter the Decrypted View Management Window:



View for Encrypted Table: SCOTT.EMP

View Owner: RDC_ENCRYPT_USER
View Name: EMP

Decrypted View Type: Read Only [✓ Updateable] None

Current columns with Function Index: COMM

Choose Column for Function Index: COMM

Function Index successfully created...

From the above window you may create a decrypted view against the chosen base table by simply entering a schema owner and view name for the decrypted view. Once you have defined the view, simply click on a Decrypted View Type of either Read-Only or Updateable and then click on the button at the bottom of the screen labeled “Reset View”.

From this screen we can change the owner, view_name, and view_type at any time.

Function Indexes for Performance

If you have encrypted a very large table, decrypted views will be very slow against SQL that refers to an encrypted column. To avoid this you may create function indexes that will allow your Decrypted view to use function index searches when encrypted data is queried. The drawback of function indexes is that column values will be stored in reverse compressed and not encrypted format within the index tablespace.

To create a function index, simply choose a column name using the drop-down list, and then click on the button labeled “Create Index”.

To drop a function index, click on the column name from the list on the left and then click on the button labeled “Drop Index”.

To use function indexes the following parameters must be set in your init.ora file:

```
query_rewrite_integrity=trusted
query_rewrite_enabled=TRUE
```

You will then need to run statistics against the table you have indexed, to insure that the Oracle optimizer will choose to use the new function index.

Tip:

We also recommend setting the init.ora parameter *optimizer_index_cost_adj* to a value between 1 and 30 for optimal use of function indexes.

Key Backup and Recovery for Continuity Planning

The Encryption Wizard for Oracle offers you the ability to backup the vital encryption keys, audit rules, runtime passwords, decrypted view definitions and restricted user lists of your encrypted data. The backup of the Encryption Wizard always originates from an Oracle RDBMS to a flat-file on your operating system. For security reasons the following rules apply to backups:

1. Backups may never be transferred between machines or databases or versions of the Oracle RDBMS.

2. Users may protect backups with a user-defined password

3. Backup and Recovery operations will protect your restricted user lists and Runtime Passwords.

4. Recovery operations will re-create decrypted views and their triggers.

5. Function-Index definitions are not backed-up.

6. Key backup files are not encrypted again to insure integrity of data and to encourage users to use another encryption tool on the server, i.e. PGP or WinZip with AES encryption to protect your key backup files with multiple points of failure.

7. Key backup files are restricted by the Oracle parameter UTL_FILE_DIR - this parameter allows you to control if and where on the file system backup and recovery operations will be allowed.

Preparing the Backup Manager for Use

To use the backup and recovery features of the Encryption Wizard you will need to set the parameter UTL_FILE_DIR to a valid directory or a wildcard, this allows the PL/SQL engine to write data to the file system on your server.

For example, if you want to only allow backups to occur to a certain directory on your file system, you might set the INIT.ORA parameter UTL_FILE_DIR as such:

```
utl_file_dir=/u01/encrypt/backups
```

If you don't want a restriction on the location of backup key files, you may enter the following wildcard as your parameter

```
utl_file_dir=*
```

Backup and Recovery Levels

The Encryption Wizard for Oracle generates small backup file because no data is actually stored in this backup - simply the encryption keys that will transform the encrypted data into useful data. Therefore it is equally important to have a standard data backup procedure also in place to backup your encrypted data.

The Encryption Wizard for Oracle can perform the key backup process at four levels or *backup domains*:

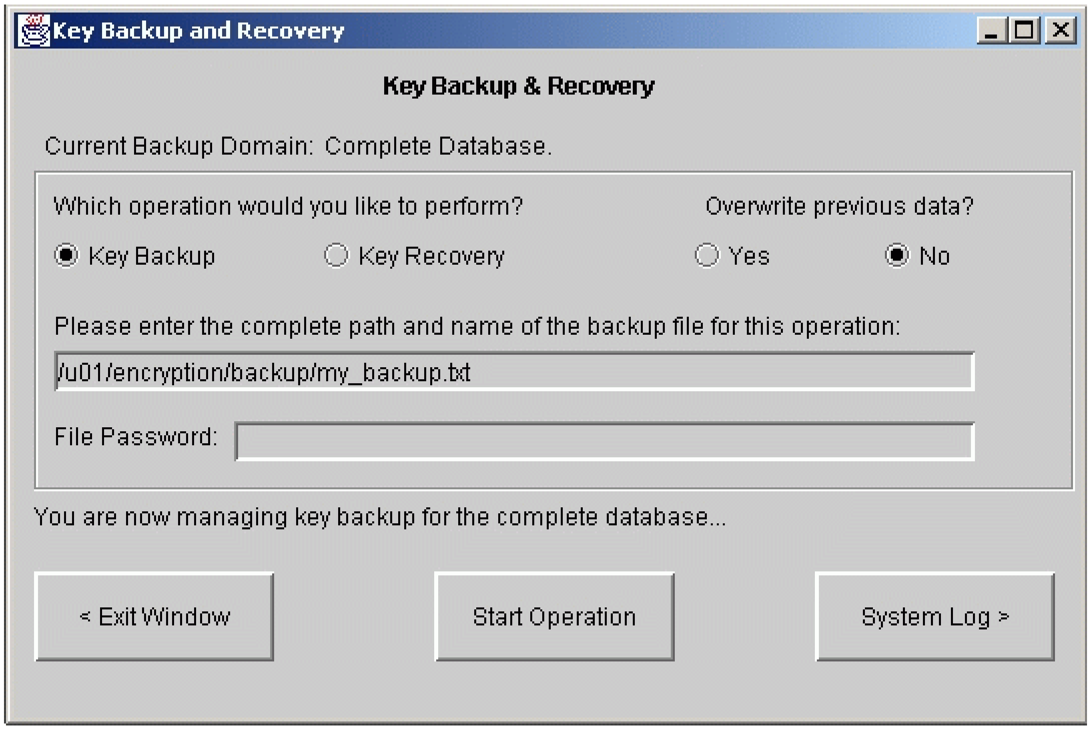
1. Complete Database Backup – To choose this level un-select any chosen schema within the Manage Encrypted Data Screen and then click on the button labeled "Manage Backups". This *backup domain* will backup all encryption keys within the database

2. Schema Backup – To choose this level, select a given schema with the Manage Encrypted Data Screen and then click on the button labeled "Manage Backups". This *backup domain* will backup all encryption keys within the chosen schema.

3. Table Backup – To choose this level, select a given table with the Manage Encrypted Data Screen and then click on the button labeled "Manage Backups". This *backup domain* will backup all encryption keys for a given table.

4. Column Backup – To choose this level, click on a given database column within the Manage Encrypted Data Screen's detail listing and then click on the button labeled "Manage Backups". This *backup domain* will backup one encryption key for the chosen schema.

Once you have clicked on the button labeled “Manage Backups” you will enter the key backup and recovery screen of the Encryption Wizard:


Key Backup and Recovery

Key Backup & Recovery

Current Backup Domain: Complete Database.

Which operation would you like to perform? [✓ Key Backup] Key Recovery

Overwrite previous data? Yes [✓ No]

Please enter the complete path and name of the backup file for this operation: /u01/encryption/backup/my_backup.txt

File Password:

You are now managing key backup for the complete database...

From the Key Backup and Recovery window we can backup all or a subset of our keys based on the *Current Backup Domain* which is the topmost field in the window.

Creating a Backup of Your Encryption Keys

To create a backup of your encryption keys simply enter the complete path of a file you wish the Encryption Wizard to create or overwrite. This file will contain your encryption key seeds in un-encrypted format. This is the same format the data appears in the database, these keys are further encrypted at runtime.

You may also optionally enter a password below the file name to further protect users from restoring these keys. In all cases, these files cannot be restored by any other database than the database for which the backup was created. For further protection we recommend using a third-party encryption tool like PGP or WinZip with AES to protect your backup files.

Once you are ready to backup your encryption keys simply click on the button labeled “Start Operation”. To overwrite an existing flat-file simply you may click on the “Yes” radio button labeled “Overwrite previous data”.

Recovering Encryption Keys from a Backup

To recover or restore lost keys simply click on the radio button labeled “recovery”. After this, enter the filename and any password that may have been assigned to that file. If you wish to overwrite existing keys in the Encryption Wizard schema, you may click on the “Yes” radio button labeled “Overwrite previous data”.

The recovery operation will perform these tasks

- Load encryption keys of the current backup domain into the Encryption Wizard schema.
- Load any restricted user lists, audit rules, and runtime passwords into the database.
- Re-create decrypted views of the affected tables.

Once you have entered the flat-file full path name you wish to perform recovery on and any password assigned for that file, simply click on “Start Operation” to begin recovery. After each recovery operation the count of how many columns, tables, and user authorizations that were restored are displayed.

Using Restore Column to Retrieve Old Keys

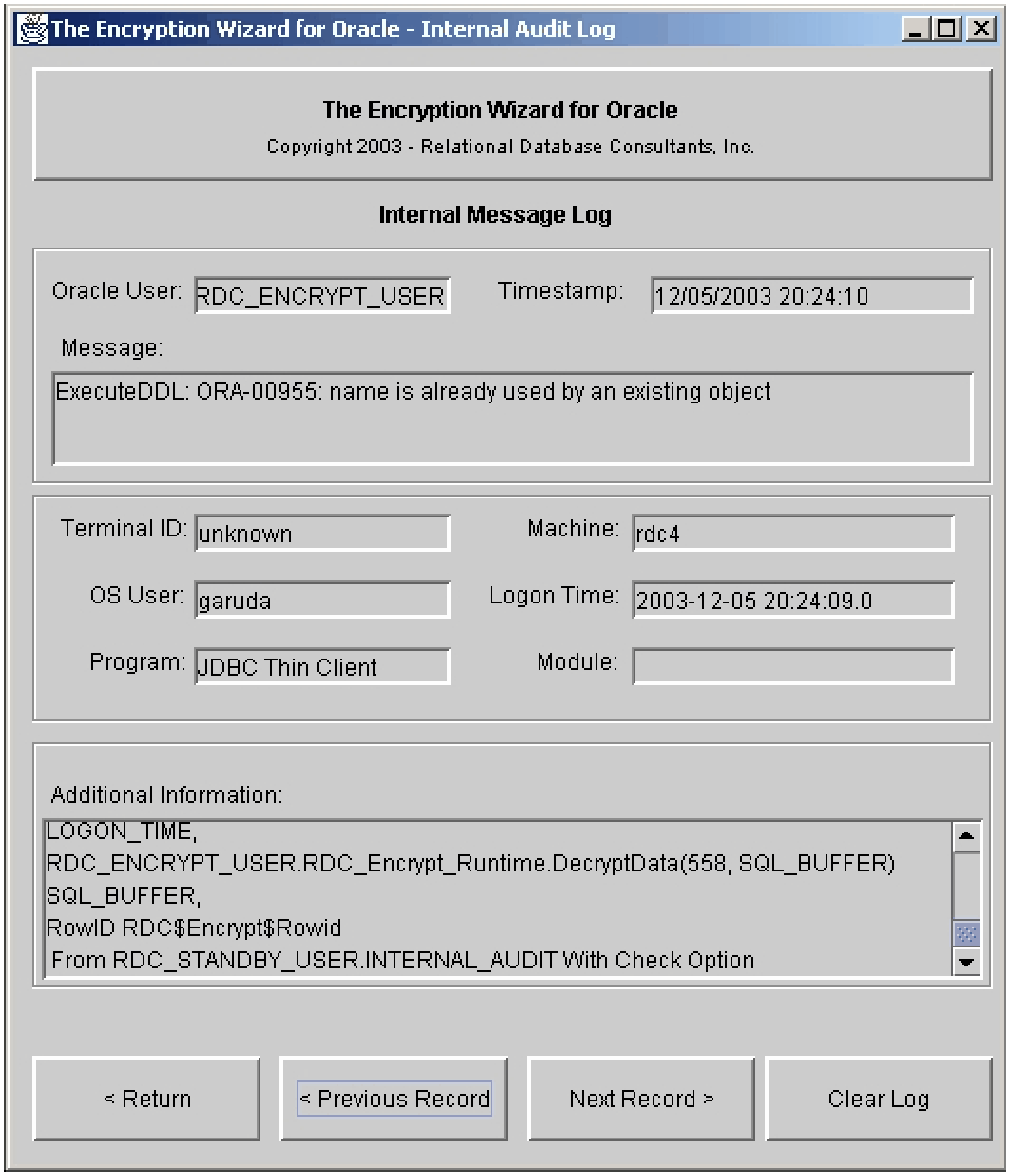
When a column key is deleted from the Encryption Wizard key table, the key is sent to a history table: *encrypted_column_history*. To retrieve a column key from the history table, use the API procedure call: *RestoreColumn.* This method will restore the key to the Encryption Wizard key table, but will not encrypt/decrypt any data.

For Instance, if we wish to use an encrypted backup of a table no longer encrypted, we can restore the previous key in the Encryption Wizard schema using *RestroeColumn* to give us access to the encrypted data.

For more on RestoreColumn, please reference the Encryption Wizard API Reference Guide.

System Log for Debugging

To debug any errors in the Encryption Wizard User Interface, the API, or the Runtime Library, first check the System Log. To do this, simply click on the button labeled "System Log" from the main menu. From here we will enter the System Log window:


The Encryption Wizard for Oracle - Internal Audit Log
The Encryption Wizard for Oracle
Copyright 2003 - Relational Database Consultants, Inc.
Internal Message Log
< Return
< Previous Record
Next Record >
Clear Log

Oracle User: RDC_ENCRYPT_USER
Timestamp: 12/05/2003 20:24:10
Message:
ExecuteDDL: ORA-00955: name is already used by an existing object

Terminal ID: unknown
Machine: rdc4
OS User: garuda
Logon Time: 2003-12-05 20:24:09.0
Program: JDBC Thin Client
Module:

Additional Information:

```
LOGON_TIME,
RDC_ENCRYPT_USER.RDC_Encrypt_Runtime.DecryptData(558, SQL_BUFFER)
SQL_BUFFER,
RowID RDC$Encrypt$Rowid
 From RDC_STANDBY_USER.INTERNAL_AUDIT With Check Option
```

It is from this screen that we can scroll through any errors or messages and view the details of the event. These records are all stored in the table *Internal_Audit* owned by the user RDC_ENCRYPT_USER. Feel free to periodically clear the System Log by clicking on the button labeled "Clear Log".

Chapter Six - Migrating Applications to Encrypted Data Tables

The Encryption Wizard makes it easy for the user to encrypt and decrypt data, yet migrating applications to encrypted data always takes some planning. The easiest way to migrate applications to the Encryption Wizard's encrypted tables – is through the use of *decrypted views*. Decrypted views are views on encrypted tables that *appear* to the user as encrypted data in its original form.

1. Decrypted View Strategy

Decrypted Views appear like to contain the original data because data is decrypted by the view. Users can read, and if specified, write to these views just as if they were base tables. When using decrypted views there are 3 basic strategies you can employ to migrate your applications to encrypted data tables:

Rename Application Base Tables

This method entails renaming your base tables *before* encryption and then instructing the Encryption Wizard to create views with the names of your original table names.

To name a decrypted view to an exact table name, simply create the decrypted view in your application schema with a temporary prefix as you are encrypting your data. After verifying the integrity of your encryption, rename the view in the Encrypted Data Management Screen discussed in Chapter 5 – Managing & Auditing Encrypted Data to the name of the original base table or another name of your choice.

The only catch to this methodology is that when you rename a table you will invalidate all triggers, procedures, functions and views to that table. Make sure you evaluate these objects to see if they are still needed for your encrypted data, and more importantly, make sure these objects will not corrupt your encrypted data. Any insert or update of encrypted data should be done through the use of decrypted views or the Encryption Wizard API.

Create Decrypted Views in Application Schema with Prefix

This second method is easy to implement with the Encryption Wizard. Simply create your decrypted views in the same schema as your encrypted application. In this case you will need to use a *decrypted view prefix* so your decrypted view name will not conflict with the original base table. You can rename any view after your initial encryption of table data using the Decrypted View Manager discussed in Chapter 5 - therefore this prefix is not binding.

Using this method we might assign a decrypted view prefix of 'VE' for the encrypted table EMP. In this case our decrypted view will have the name: VE_EMP and will be owned by the schema owner SCOTT. When accessing data, we now refer to the new decrypted view in our SQL as in these examples:

*Select * from VE_Emp;   Update VE_Emp Set Salary = Salary + 100;*

The drawback of this method is that your applications will now have to point to the new table names, in this case the names assigned to your decrypted views.

Create Decrypted Views and Use Synonyms

This is the easiest path if you use public synonyms. By default, the Encryption Wizard will create all decrypted views in its own schema, *rdc_encrypt_user.* Using this migration method, a developer would encrypt a table, let us say table EMP in the SCOTT schema.

After completion, the developer would point his or her applications to a public synonym named emp that refers to the decrypted view and not the table.

Create public synonym emp for rdc_encyrpt_user.emp;

Now users can seamlessly use this new synonym, as if they were referring to the original table EMP.

2. Database Trigger Migration

There are two concerns regarding database triggers that are defined on tables that are encrypted:

Data Corruption - Before Update or Insert Row Level Triggers

These triggers fire *after* the Encryption Wizard decrypted view issues an encrypted write to the encrypted table. Therefore these triggers *cannot modify any column* that is encrypted unless they are modified with the Encryption Wizard API.

Invalid Selects - All Other Table Triggers

Other table triggers, such as after triggering event triggers, might still corrupt other tables or make incorrect business decisions *if they reference any decrypted column*. These triggers still fire *after* the decrypted view issues an encrypted write and therefore might *read* obfuscated data that will make no logical sense, unless of course the Encryption Wizard API is used to decrypt these trigger variables.

RDC will gladly modify sample triggers of your choosing with the Encryption Wizard API, as examples for use.

Tip:

You can insert your trigger logic into the Instead Of Trigger generated by the Encryption Wizard. Use your favorite database trigger editing tool, and insert your trigger logic *before* the Encryption Wizard's routines encrypt the table data. These triggers are always owned by the Encryption Wizard under the schema owner *rdc_encrypt_user.*

3. SQL Migration

The only SQL migration that is necessary when using the Encryption Wizard is for tables with no primary key and wild card SQL Inserts or fetches.

With a primary key, the Encryption Wizard adds a *rowid* column to the decrypted view necessary for DDL operations. For instance, consider the SQL statement on a decrypted view called emp:

```
Insert Into Backup_Emp
Select * From Emp;

Or

Insert Into Emp
Values (a,b,c)

Or

Select * From Emp
Fetch Into A, B, C;
```

If these three statements refer to a table without a primary key, then they will fail. The reason is that the additional rowid column needed by the Encryption Wizard will not exist in the above example table.

If you do have potential SQL that uses wildcards in DDL, it is recommended a primary key be created for a base table being encrypted, if one is not already present. The syntax to add a primary key to a table is:

Alter Table *Table Name* Add Constraint *ConstraintName* Primary Key (*ColumnName*);

ConstraintName can be any string such as: MyConstraint123.

Multiple column names can also be used to define primary keys:

Alter Table *Table Name* Add Constraint *Constraint Name* Primary Key (*Col1, Col2, Col2*);

Warning:

The column(s) used in a primary key cannot be encrypted.

If you do not use primary keys, SQL such as the above statements, will need to be rewritten to explicitly name every column they reference. For example the second statement should read:

Insert into Emp (a, b, c, d) Values (a, b, c, d);

Generally speaking, wildcard DDL is rare in SQL and is considered bad practice.

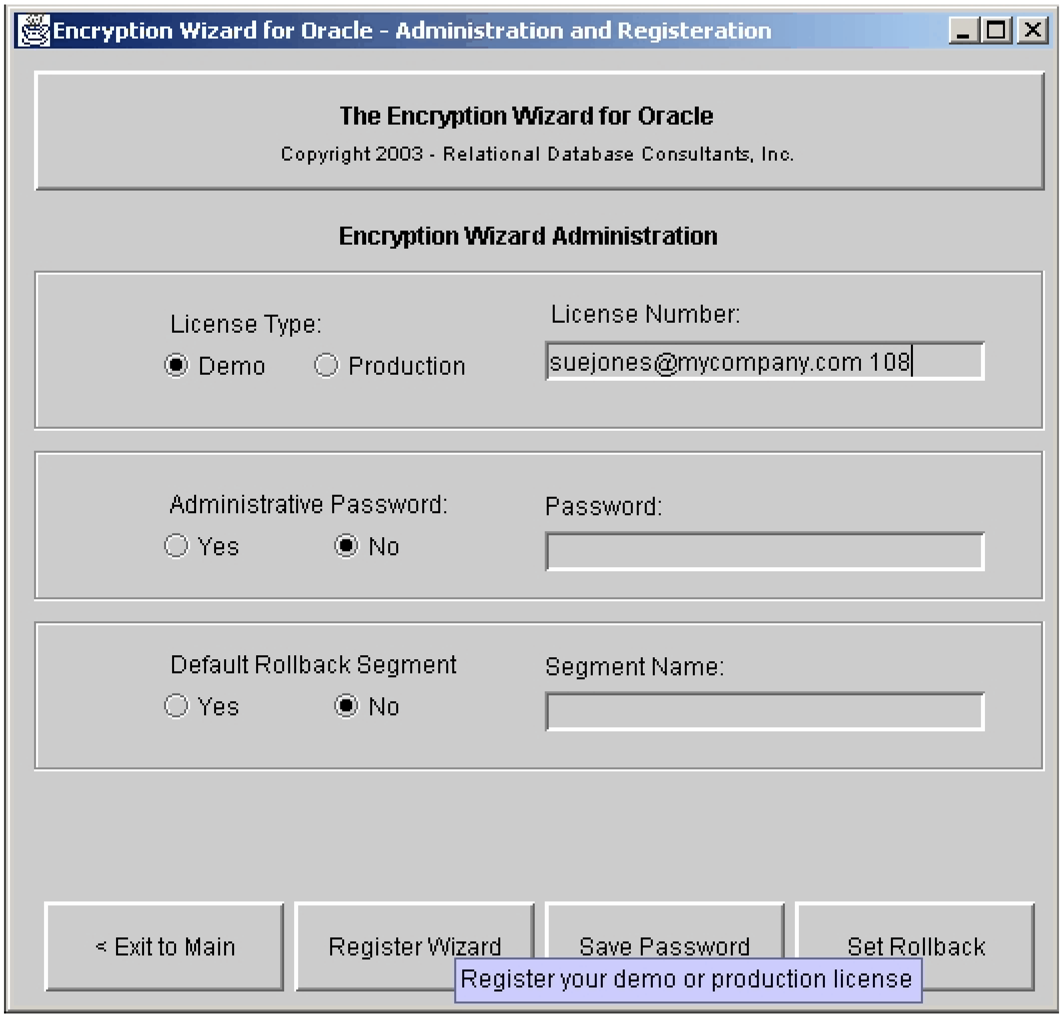
Tip:

If you would like the Encryption Wizard to add a rowid to all decrypted views and not use a primary key, even if one is present, issue this SQL after installation:

Update rdc_encrypt_license set use_rowid_for_views = 'X'; commit;

Chapter Seven – Encryption Wizard Administration

By using the Encryption Wizard Administration Screen we can set an administration password for the Encryption Wizard and define a default rollback segment for large encryption efforts. To enter the Administration screen, simply click on the main menu item labeled *Administration*:


Encryption Wizard for Oracle - Administration and Registeration

The Encryption Wizard for Oracle
Copyright 2003 - Relational Database Consultants, Inc.

Encryption Wizard Administration

License Type: [✓ Demo] Production
License Number: suejones@mycompany.com 108

Administrative Password: Yes [✓ No]
Password:

Default Rollback Segment: Yes [✓ No]
Segment Name:

Register your demo or production license

1. Creating an Administrative Password

Because any DBA can access all objects in your database, the Encryption Wizard allows you to choose an administrative password that will be required to enter before a user can use the Encryption Wizard or its underlying API. The administrative password will stop unauthorized privileged users from using the Encryption Wizard.

To enter an administrative password, simply follow these steps

Step One:

Click on the radio button labeled “yes” for the question: *Require Administrative Password?*

Step Two:

Enter an administrative password of up to 30 characters.

Step Three:

Click on the button labeled “Save Password” – You will be asked to verify your entry.

After you have completed these steps, users will be required to enter this password before using the Encryption Wizard user interface or the Encryption Wizard API.

To remove the use of the administrative password, simply click on the radio button labeled “no” next to the question: *Require Administrative Password?* After this, simply click on the button labeled *Save Password* to remove the administrative password requirement.

2. Setting a Default Rollback Segment for Transactions

Even though the Encryption Wizard performs a commit every 4,096 rows, if you are encrypting a table with many large columns and not using undo tablespaces, you may need to assign a larger rollback segment for the Encryption Wizard’s use than supplied by your default Oracle installation.

To accomplish this task, simply type in a valid, currently online rollback segment - and then click on the button labeled *Set Rollback Segment.* Now, Encryption and Decryption operations will be set to use the rollback segment you specified.

3. Setting Encryption to Hide Null Values

To hide null character data, simply make a call to the Encryption Wizard API from SQL For specifications of the API call refer to the description of *EncryptNullData* in the API Library Reference. This setting has a similar effect as the constraint NOT NULL and may affect out-joins. To set null encryption as the default, issue this call before encrypting a data set:

```
SQL>Exec RDC_Encrypt_Util.EncryptNullData('Yes');
```

To turn off this feature, which will only affect future data to be encrypted, issue this call:

```
SQL>Exec RDC_Encrpyt_Util.EncryptNullData('No');
```

4. Registering the Encryption Wizard After Purchase

After your purchase of the Encryption Wizard, simply provide your sales representative with the current database ID of your database. This value is displayed when you enter the Administration Screen of the Encryption Wizard.

Your sales representative with give you a new production license number and registration. To purchase the Encryption Wizard for Oracle, contact us at ewdemo@relationalwizards.com.

Chapter Eight – HSM Interface

The Encryption Wizard now ships with a Java interface to Hardware Security Modules (HSM), thus providing compliance for remote key storage and management. The Encryption Wizard stores one master key on the HSM and using SHA hashing techniques the HSM-stored key is used to generate all of the Encryption Wizard's runtime keys and passwords. Thus encrypted data will not be accessible unless the Encryption Wizard can access the HSM.

Valid HSM Types:

1. Luna PCI 7000

Make sure you have decrypted all data before installing the Encryption Wizard HSM interface. You will also need to re-register with both your demo and/or production license codes after the HSM installation – since all keys are reset.

1. Oracle JVM Jar & Class File Installation

Certain java classes must be loaded into the Oracle JVM from the command-line using Oracle's loadjava utility. Only SYS is able to load the HSM DLL(s) at runtime.

For Luna PCI 7000 users:

OS>loadjava -u sys -v -resolve -f *ibmpkcs.jar*

The jar file ibmpkcs.jar can be obtained at the following link:

IBM JCE

OS>loadjava -u sys -v -resolve -f *LunaJCASP.jar LunaJCESP.jar*

These Jar files are provided by Safenet, Inc. with the Luna installation.

OS>loadjava -u sys -v -resolve -f *RDC$JavaHSM.class*

This file is included with your Encryption Wizard installation.

2. Encryption Wizard HSM Installation

The Encryption Wizard HSM installation is always performed as SYS, since SYS is the only Oracle user that can load the HSM API's (LunaAPI.dll). To run the installation script, you will need the following information:

1. The HSM Key Name you would like to assign the master key of the Encryption Wizard.
2. The Password that you used on your HSM to create a key store/partition.
3. The Slot Number your HSM is installed on. (Optional, login can search)
4. The case-sensitive valid Card Type. For example, 'Luna PCI 7000'
5. The file LunaAPI.dll. You will prompted by this script to copy it into one of a choice of directories that are in the *java library path* of your Oracle instance.
6. You will also need the Encryption Wizard Oracle user password and optionally the demo license code you installed the product with.

3. Running the installation script

Once the above prerequisites are met, HSM installation is completed by running the script: r*dc_hsm_install.sql*, directly from the Encryption Wizard installation directory. (*C:\Program Files\Encryption Wizard for Oracle* or your .Zip file extraction.)

```
SQL>@rdc_hsm_install.sql
```

Output of the script will go to its .lst extension.

After you have copied the HSM API file(s) to Oracle's java library path and successfully entered the parameters of your HSM card, you should see output something like this:

```
HSM Parameters Successfully Installed:

NAME                          VALUE
----------------------------- -----------------------------
Card Type                     Luna PCI 7000
Cipher Algorithm              HmacSHA1
Cipher Provider               SunJCE version 1.4
Input Key Algorithm           GenericSecret
Input Key Length              256
Inserted Key Algorithm        HmacSHA256
Inserted Key Length           256
Java Library Path             c:\oracle\prod10g\bin;.;C:\WIN
Java Version                  1.4.2_04
...                           ...
```

If your installation fails, you can start this script over as many times as you would like, keeping in mind the Encryption Wizard does not delete old key names from the HSM. Installation errors are stored in the Encryption Wizard System Log as discussed in Chapter Five.

Make sure to manually drop the table *hsm_parameters*, if you are not running this script for the first time and the table was created. This is a safeguard against unintentionally overwriting a master key and/or its parameters.

At the end of the installation, you will be prompted for your demo license code. If you wish to register later you may hit return.

4. HSM Maintenance Tasks

Once the Encryption Wizard HSM interface is successfully installed, all passwords and data will be dependent on the successful functioning of the HSM card. Make sure to use your HSM vendor's backup utilities.

The two maintenance tasks regarding your HSM card and the Encryption Wizard are the following:

1. Informing the Encryption Wizard of a new HSM slot number.
2. Informing the Encryption Wizard of a new HSM password.

Both of these functions are accomplished using the Encryption Wizard API:

SQL>exec RDC_Encrypt_HSM.ChangeParameter('Slot Num','2','*HSMPassword*');

SQL>exec RDC_Encrypt_HSM.ChangeParameter('Password,'*NewHSMPassword*','*OldHSMPassword*');

Refer to section 3 of the Encryption Wizard API Reference Guide for specifications on the HSM API calls.

All runtime errors regarding the HSM card are stored in the Encryption Wizard System Log in the table *internal_audit.*

5. Un-installing the HSM Interface

To un-install the Encryption Wizard HSM interface, simply decrypt your data and run the original installation script: rdc_encrypt_install.sql.

You may also want to drop these objects from SYS;

```
drop function SYS.RDC$ObfuscateSeed;
drop function SYS.RDC$InitMasterKey;
drop function SYS.RDC$GetJavaParameter;
drop java class “RDC$JavaHSM”;
```

Chapter Nine - Management Reports

The Encryption Wizard for Oracle provides you with three useful management reports. The first of these reports allow you to keep track of your current encrypted columns. The two auditing reports are designed to track auditing and usage of encrypted columns that we learned to set in the Security Manager.

We recommend that the output of these reports be exported into MS Word or a similar tool for advanced formatting. If you are looking for reports that output .PDF or .HTML files you will need to use Oracle*Reports with the Oracle*Forms version of the Encryption Wizard, which is available at: ewdemo@relationalwizards.com.

Report Descriptions

All of the Encryption Wizard management reports are files within the installation directory with the extension ".rep". At the writing of this manual the current provided managements reports are:

1. **Current Encrypted Columns**

   Description:

   This report displays basic information on all of your encryptions.

   File Name:

   encrypted_columns.rep

2. **Audit Log Summary**

   Description:

   This report displays summary statistics for any audited columns that you have set for either read or write auditing.

   File Name:

   audit_log_summary.rep

3. **Audit Log Details**

   Description:

   This report displays detailed row information for any audited columns that you have set for either read or write auditing, including failed transactions due to *restricted user lists* set up in the Security Manager of the Encryption Wizard.

   File Name:

   audit_log_details.rep

Running the Management Reports

All reports are run directly from SQL*Plus. These reports can be found in the C:\Program Files\Encryption Wizard for Oracle directory or within the EncryptionWizardJava.zip download.

1. Step One

Login as RDC_Encrypt_User using SQL*Plus.

2. Step Two

Start the report from SQL*Plus as such

```
SQL>@audit_log_summary.rep
```

Or

```
SQL>start encrypted_columns.rep
```

3. Step Three

You will be prompted for four parameters that define the scope of your report

1.	Schema Owner	- Enter the Schema Owner or hit return for all schema data
2.	Table Name	- Enter the Table Name or hit return for all tables
3.	Column Name	- Enter the Column Name or hit return for all columns
4.	Output File	- Enter a valid file name for your output.

4. Step Four

View your report in the Output File you specified above. Exporting these reports to MS Word or a PDF tool allows for quick formatting.

Sample Output of Management Reports

Following we have provided three examples of the management report's output. In each case we hit <return> for the schema, table, and column specifications, so these reports cover all encrypted columns within our database.

Example One – Current Encrypted Columns

```
Time: 07/30/03 13:25:15                                    Page:          1

              The Encryption Wizard for Oracle - Management Reports

                  *** Listing of Current Encrypted Columns ***

                                            Data     Encryption  Audit
Table Owner Table Name      Column Name     Type     Type        Type  Restrict?
----------- --------------- --------------- -------- ----------- ----- ---------
SUPPS       BUGS            ASSIGNED_DATE   DATE     Obfuscation None  No
                            ASSIGNED_TO     VARCHAR2 Triple DES3 None  No
                            BUG_SEQ         NUMBER   Obfuscation None  No
                            CLOSED_BY       VARCHAR2 Triple DES3 None  No
                            CLOSED_DATE     DATE     Obfuscation None  No
                            CLOSED_TIME_SPE NUMBER   Obfuscation None  No
                            CLOSED_TIME_SPE NUMBER   Obfuscation None  No
                            CTN_SEQ         NUMBER   Obfuscation None  No
                            DUE_DATE        DATE     Obfuscation None  No
                            ENTERED_BY      VARCHAR2 Triple DES3 None  No
                            ENTERED_DATE    DATE     Obfuscation None  No
                            FIXED_BY        VARCHAR2 Triple DES3 None  No
                            FIXED_DATE      DATE     Obfuscation None  No
                            FIX_TIME_SPENT_ NUMBER   Obfuscation None  No
                            FIX_TIME_SPENT_ NUMBER   Obfuscation None  No
                            MODULE_ID       VARCHAR2 Triple DES3 None  No
                            PRIORITY_CODE   VARCHAR2 Triple DES3 None  No
                            PRIORITY_DATE   DATE     Obfuscation None  No
                            PRODUCT_ID      VARCHAR2 Triple DES3 None  No
                            PRODUCT_RELEASE VARCHAR2 Triple DES3 None  No
                            PRODUCT_VERSION VARCHAR2 Triple DES3 None  No
                            RELATED_BUG_SEQ NUMBER   Obfuscation None  No
                            SEVERITY_CODE   VARCHAR2 Triple DES3 None  No
                            SEVERITY_DATE   DATE     Obfuscation None  No
                            STATUS_CODE     VARCHAR2 Triple DES3 None  No
                            STATUS_DATE     DATE     Obfuscation None  No
                            SUMMARY         VARCHAR2 Triple DES3 None  No
                            TEST_PHASE_ID   VARCHAR2 Triple DES3 None  No
                            TEST_SCRIPT_ID  VARCHAR2 Triple DES3 None  No
                            TIMESTAMP       DATE     Obfuscation None  No
                            USER_ID         VARCHAR2 Triple DES3 None  No
                            VERIFIED_BY     VARCHAR2 Triple DES3 None  No
                            VERIFIED_DATE   DATE     Obfuscation None  No
                            VERI_TIME_SPENT NUMBER   Obfuscation None  No
                            VERI_TIME_SPENT NUMBER   Obfuscation None  No
```

```
Time: 07/30/03 13:25:15                                   Page:         2

              The Encryption Wizard for Oracle - Management Reports

                  *** Listing of Current Encrypted Columns ***

                                        Data     Encryption  Audit
Table Owner Table Name      Column Name     Type     Type        Type  Restrict?
----------- --------------- --------------- -------- ----------- ----- ---------
SALE        CUSTOMER        ADDRESS         VARCHAR2 DES         None  No
                            AREA_CODE       NUMBER   Obfuscation None  No
                            CITY            VARCHAR2 DES         None  No
                            CREDIT_LIMIT    NUMBER   Obfuscation None  No
                            NAME            VARCHAR2 DES         None  No
                            PHONE_NUMBER    NUMBER   Obfuscation None  No


            DEPARTMENT      NAME            VARCHAR2 DES         None  No


            EMPLOYEE        COMMISSION      NUMBER   Obfuscation None  No
                            FIRST_NAME      VARCHAR2 DES         None  No
                            HIRE_DATE       DATE     Obfuscation None  No
                            LAST_NAME       VARCHAR2 DES         None  No
                            MIDDLE_INITIAL  VARCHAR2 DES         None  No
                            SALARY          NUMBER   Obfuscation None  No


            ITEM            ACTUAL_PRICE    NUMBER   Obfuscation None  No
                            QUANTITY        NUMBER   Obfuscation None  No
                            TOTAL           NUMBER   Obfuscation None  No


            JOB             FUNCTION        VARCHAR2 DES         None  No


            LOCATION        REGIONAL_GROUP  VARCHAR2 DES         None  No


            PRODUCT         DESCRIPTION     VARCHAR2 DES         None  No


            SALES_ORDER     ORDER_DATE      DATE     Obfuscation None  No
                            SHIP_DATE       DATE     Obfuscation None  No
```

Example Two – Audit Log Summary

```
Time: 07/30/03 13:25:15                                          Page:         1

              The Encryption Wizard for Oracle - Management Reports

                            *** Audit Log Summary ***

                                            Data        Total    Total    Total
Table Owner Table Name      Column Name     Type        Reads   Writes    Fails
----------- --------------- --------------- -------- -------- -------- --------
SUPPS       BUGS            ASSIGNED_DATE   DATE            7        1        2
                            ASSIGNED_TO     VARCHAR2        7        1        2
                            BUG_SEQ         NUMBER          7        1        2
                            ENTERED_BY      VARCHAR2        7        1        2
                            ENTERED_DATE    DATE            7        1        2
                            MODULE_ID       VARCHAR2        7        1        2
                            PRIORITY_CODE   VARCHAR2        7        1        2
                            PRODUCT_ID      VARCHAR2        7        1        2
                            PRODUCT_RELEASE VARCHAR2        7        1        2
                            PRODUCT_VERSION VARCHAR2        7        1        2
                            SEVERITY_CODE   VARCHAR2        7        1        2
                            STATUS_CODE     VARCHAR2        7        1        2
                            SUMMARY         VARCHAR2        7        1        2
                            TEST_PHASE_ID   VARCHAR2        7        1        2
                            TEST_SCRIPT_ID  VARCHAR2        7        1        2
                            TIMESTAMP       DATE            7        1        2
                            USER_ID         VARCHAR2        7        1        2
                            VERIFIED_BY     VARCHAR2        7        1        2
                            VERIFIED_DATE   DATE            7        1        2


            BUG_ENVIRONMENT BUG_SEQ         NUMBER          4        0        1
                            ENVIRONMENT_VAL VARCHAR2        4        0        1
                            ENVIRONMENT_VAR VARCHAR2        4        0        1


            BUG_SOURCE_FILE BUG_SEQ         NUMBER          4        0        1
                            COMMENTS        VARCHAR2        4        0        1
                            MODIFIED_BY     VARCHAR2        4        0        1
                            MODIFIED_DATE   DATE            4        0        1
                            MODULE_ID       VARCHAR2        4        0        1
                            PRODUCT_ID      VARCHAR2        4        0        1
                            SOURCE_FILE     VARCHAR2        4        0        1


            BUG_TEXT        BUG_SEQ         NUMBER          4        0        1
                            LINE_NO         NUMBER          4        0        1
                            TEXT            VARCHAR2        4        0        1
                            TEXT_TYPE       VARCHAR2        4        0        1


            CONFIGURE       COMPANY_NAME    VARCHAR2        4        0        1
                            LICENSE_DATE    DATE            4        0        1
                            LICENSE_NO      VARCHAR2        4        0        1
                            OPERATING_SYSTE VARCHAR2        4        0        1
                            PRINT_COMMAND   VARCHAR2        4        0        1
```

```
Time: 07/30/03 13:25:15                                     Page:         2

              The Encryption Wizard for Oracle - Management Reports

                            *** Audit Log Summary ***

                                             Data        Total    Total    Total
Table Owner Table Name      Column Name     Type         Reads   Writes    Fails
----------- --------------- --------------- -------- -------- -------- --------


SUPPS       ENVIRONMENT     DESCRIPTION     VARCHAR2        4        0        1
                            ENVIRONMENT_VAR VARCHAR2        4        0        1
                            MANDATORY       VARCHAR2        4        0        1


            FIELD_RULES     DEFAULT_VALUE   VARCHAR2        4        0        1
                            FIELD_NAME      VARCHAR2        4        0        1
                            FORM_NAME       VARCHAR2        4        0        1


            FUNC_KEYS       BUG_USAGE       VARCHAR2        4        0        1
                            CUSTOMER_USAGE  VARCHAR2        4        0        1
                            FORM_NAME       VARCHAR2        4        0        1
                            KEY_ID          VARCHAR2        4        0        1
                            ORACLE_FUNCTION VARCHAR2        4        0        1


            MODULES         ASSIGNED_TO     VARCHAR2        4        0        1
                            MODULE_DESC     VARCHAR2        4        0        1
                            MODULE_ID       VARCHAR2        4        0        1
                            PRODUCT_ID      VARCHAR2        4        0        1


            MODULES_MAIL    MAIL_STOP       VARCHAR2        4        0        1
                            MODULE_ID       VARCHAR2        4        0        1
                            PRODUCT_ID      VARCHAR2        4        0        1
                            USER_ID         VARCHAR2        4        0        1
                            USER_NAME       VARCHAR2        4        0        1


            PRIORITY_CODES  PRIORITY_CODE   VARCHAR2        4        0        1
                            PRIORITY_DESC   VARCHAR2        4        0        1


            PRODUCTS        ASSIGNED_TO     VARCHAR2        4        0        1
                            PRODUCT_DESC    VARCHAR2        4        0        1
                            PRODUCT_ID      VARCHAR2        4        0        1


            PRODUCT_RELEASE ASSIGNED_TO     VARCHAR2        4        0        1
                            DESCRIPTION     VARCHAR2        4        0        1
                            PRODUCT_ID      VARCHAR2        4        0        1
                            RELEASE_DATE    DATE            4        0        1
                            RELEASE_ID      VARCHAR2        4        0        1
```

Example Three – Audit Log Details

```
Time: 07/30/03 13:25:15                                          Page:          1

              The Encryption Wizard for Oracle - Management Reports

                            *** Audit Log Details ***

Table Owner: SUPPS
                                Audit             User    From      Audit
Table Name      Column Name     Timestamp         Name    Program   Type
--------------- --------------- ----------------- ------- --------- ----------
BUGS            ASSIGNED_DATE   07/30/03 13:35:20 JOE     JDBC Thin Write Fail
                ASSIGNED_DATE   07/30/03 13:37:52 SUE     sqlplusw. Read Fail
                ASSIGNED_TO     07/30/03 13:35:20 BOB     JDBC Thin Write Fail
                ASSIGNED_TO     07/30/03 13:37:52 JOE     sqlplusw. Read Fail
                BUG_SEQ         07/30/03 13:35:20 SUE     JDBC Thin Write Fail
                BUG_SEQ         07/30/03 13:37:52 BOB     sqlplusw. Read Fail
                ENTERED_BY      07/30/03 13:35:20 JOE     JDBC Thin Write Fail
                ENTERED_BY      07/30/03 13:37:52 SUE     sqlplusw. Read Fail
                ENTERED_DATE    07/30/03 13:35:20 BOB     JDBC Thin Write Fail
                ENTERED_DATE    07/30/03 13:37:52 JOE     sqlplusw. Read Fail
                MODULE_ID       07/30/03 13:35:20 SUE     JDBC Thin Write Fail
                MODULE_ID       07/30/03 13:37:52 BOB     sqlplusw. Read Fail
                PRIORITY_CODE   07/30/03 13:35:20 JOE     JDBC Thin Write Fail
                PRIORITY_CODE   07/30/03 13:37:52 SUE     sqlplusw. Read Fail
                PRODUCT_ID      07/30/03 13:35:20 BOB     JDBC Thin Write Fail
                PRODUCT_ID      07/30/03 13:37:52 JOE     sqlplusw. Read Fail
                PRODUCT_RELEASE 07/30/03 13:35:20 SUE     JDBC Thin Write Fail
                PRODUCT_RELEASE 07/30/03 13:37:52 BOB     sqlplusw. Read Fail
                PRODUCT_VERSION 07/30/03 13:35:20 JOE     JDBC Thin Write Fail
                PRODUCT_VERSION 07/30/03 13:37:52 SUE     sqlplusw. Read Fail
                SEVERITY_CODE   07/30/03 13:35:20 BOB     JDBC Thin Write Fail
                SEVERITY_CODE   07/30/03 13:37:48 JOE     sqlplusw. Read Fail
                STATUS_CODE     07/30/03 13:35:20 SUE     JDBC Thin Write Fail
                STATUS_CODE     07/30/03 13:37:48 BOB     sqlplusw. Read Fail
                SUMMARY         07/30/03 13:35:21 JOE     JDBC Thin Write Fail
                SUMMARY         07/30/03 13:37:48 SUE     sqlplusw. Read Fail
                TEST_PHASE_ID   07/30/03 13:35:21 BOB     JDBC Thin Write Fail
                TEST_PHASE_ID   07/30/03 13:37:48 JOE     sqlplusw. Read Fail
                TEST_SCRIPT_ID  07/30/03 13:35:21 SUE     JDBC Thin Write Fail
                TEST_SCRIPT_ID  07/30/03 13:37:48 BOB     sqlplusw. Read Fail
                TIMESTAMP       07/30/03 13:35:21 JOE     JDBC Thin Write Fail
                TIMESTAMP       07/30/03 13:37:48 SUE     sqlplusw. Read Fail
                USER_ID         07/30/03 13:35:21 BOB     JDBC Thin Write Fail
                USER_ID         07/30/03 13:37:48 JOE     sqlplusw. Read Fail
                VERIFIED_BY     07/30/03 13:35:21 SUE     JDBC Thin Write Fail
                VERIFIED_BY     07/30/03 13:37:48 BOB     sqlplusw. Read Fail
                VERIFIED_DATE   07/30/03 13:35:21 JOE     JDBC Thin Write Fail
                VERIFIED_DATE   07/30/03 13:37:48 SUE     sqlplusw. Read Fail


BUG_ENVIRONMENT BUG_SEQ         07/30/03 13:37:49 JOE     sqlplusw. Read Fail
                ENVIRONMENT_VAL 07/30/03 13:37:49 BOB     sqlplusw. Read Fail
                ENVIRONMENT_VAR 07/30/03 13:37:49 SUE     sqlplusw. Read Fail
```

```
Time: 07/30/03 13:25:15                                          Page:         2

              The Encryption Wizard for Oracle - Management Reports

                            *** Audit Log Details ***

Table Owner: SUPPS
                                Audit             User    From      Audit
Table Name      Column Name     Timestamp         Name    Program   Type
--------------- --------------- ----------------- ------- --------- ----------
BUG_SOURCE_FILE BUG_SEQ         07/30/03 13:37:49 JOE     sqlplusw. Read Fail
                COMMENTS        07/30/03 13:37:49 BOB     sqlplusw. Read Fail
                MODIFIED_BY     07/30/03 13:37:46 SUE     sqlplusw. Read Fail
                MODIFIED_DATE   07/30/03 13:37:46 JOE     sqlplusw. Read Fail
                MODULE_ID       07/30/03 13:37:46 BOB     sqlplusw. Read Fail
                PRODUCT_ID      07/30/03 13:37:46 SUE     sqlplusw. Read Fail
                SOURCE_FILE     07/30/03 13:37:46 JOE     sqlplusw. Read Fail


BUG_TEXT        BUG_SEQ         07/30/03 13:37:46 BOB     sqlplusw. Read Fail
                LINE_NO         07/30/03 13:37:46 SUE     sqlplusw. Read Fail
                TEXT            07/30/03 13:37:46 JOE     sqlplusw. Read Fail
                TEXT_TYPE       07/30/03 13:37:46 BOB     sqlplusw. Read Fail


CONFIGURE       COMPANY_NAME    07/30/03 13:37:47 SUE     sqlplusw. Read Fail
                LICENSE_DATE    07/30/03 13:37:47 JOE     sqlplusw. Read Fail
                LICENSE_NO      07/30/03 13:37:47 BOB     sqlplusw. Read Fail
                OPERATING_SYSTE 07/30/03 13:37:47 SUE     sqlplusw. Read Fail
                PRINT_COMMAND   07/30/03 13:37:47 JOE     sqlplusw. Read Fail


ENVIRONMENT     DESCRIPTION     07/30/03 13:37:47 BOB     sqlplusw. Read Fail
                ENVIRONMENT_VAR 07/30/03 13:37:47 SUE     sqlplusw. Read Fail
                MANDATORY       07/30/03 13:37:47 JOE     sqlplusw. Read Fail


FIELD_RULES     DEFAULT_VALUE   07/30/03 13:37:48 SUE     sqlplusw. Read Fail
                FIELD_NAME      07/30/03 13:37:48 BOB     sqlplusw. Read Fail
                FORM_NAME       07/30/03 13:37:50 SUE     sqlplusw. Read Fail


FUNC_KEYS       BUG_USAGE       07/30/03 13:37:50 JOE     sqlplusw. Read Fail
                CUSTOMER_USAGE  07/30/03 13:37:50 BOB     sqlplusw. Read Fail
                FORM_NAME       07/30/03 13:37:50 SUE     sqlplusw. Read Fail
                KEY_ID          07/30/03 13:37:50 JOE     sqlplusw. Read Fail
                ORACLE_FUNCTION 07/30/03 13:37:50 BOB     sqlplusw. Read Fail


MODULES         ASSIGNED_TO     07/30/03 13:37:51 SUE     sqlplusw. Read Fail
                MODULE_DESC     07/30/03 13:37:51 JOE     sqlplusw. Read Fail
                MODULE_ID       07/30/03 13:37:51 BOB     sqlplusw. Read Fail
                PRODUCT_ID      07/30/03 13:37:51 SUE     sqlplusw. Read Fail
```

```
Time: 07/30/03 13:25:15                                          Page:         3

              The Encryption Wizard for Oracle - Management Reports

                            *** Audit Log Details ***

Table Owner: SUPPS
                                Audit             User    From      Audit
Table Name      Column Name     Timestamp         Name    Program   Type
--------------- --------------- ----------------- ------- --------- ----------
MODULES_MAIL    MAIL_STOP       07/30/03 13:37:51 JOE     sqlplusw. Read Fail
                MODULE_ID       07/30/03 13:37:51 BOB     sqlplusw. Read Fail
                PRODUCT_ID      07/30/03 13:37:51 SUE     sqlplusw. Read Fail
                USER_ID         07/30/03 13:37:51 JOE     sqlplusw. Read Fail
                USER_NAME       07/30/03 13:37:51 BOB     sqlplusw. Read Fail


PRIORITY_CODES  PRIORITY_CODE   07/30/03 13:37:52 SUE     sqlplusw. Read Fail
                PRIORITY_DESC   07/30/03 13:37:52 JOE     sqlplusw. Read Fail


PRODUCTS        ASSIGNED_TO     07/30/03 13:37:52 BOB     sqlplusw. Read Fail
                PRODUCT_DESC    07/30/03 13:37:44 JOE     sqlplusw. Read Fail
                PRODUCT_ID      07/30/03 13:37:44 BOB     sqlplusw. Read Fail


PRODUCT_RELEASE ASSIGNED_TO     07/30/03 13:37:45 SUE     sqlplusw. Read Fail
                DESCRIPTION     07/30/03 13:37:45 JOE     sqlplusw. Read Fail
                PRODUCT_ID      07/30/03 13:37:45 BOB     sqlplusw. Read Fail
                RELEASE_DATE    07/30/03 13:37:45 SUE     sqlplusw. Read Fail
                RELEASE_ID      07/30/03 13:37:45 JOE     sqlplusw. Read Fail
                VERSION_ID      07/30/03 13:37:45 BOB     sqlplusw. Read Fail


RELEASES_MAIL   MAIL_STOP       07/30/03 13:37:46 SUE     sqlplusw. Read Fail
                PRODUCT_ID      07/30/03 13:37:46 JOE     sqlplusw. Read Fail
                RELEASE_ID      07/30/03 13:37:46 BOB     sqlplusw. Read Fail
                USER_ID         07/30/03 13:37:42 SUE     sqlplusw. Write Fail
                USER_NAME       07/30/03 13:37:42 JOE     sqlplusw. Write Fail
                VERSION_ID      07/30/03 13:37:42 BOB     sqlplusw. Write Fail


REPORTS         DESCRIPTION     07/30/03 13:37:42 SUE     sqlplusw. Write Fail
                FILENAME        07/30/03 13:37:42 JOE     sqlplusw. Write Fail
                PRINT_COMMAND   07/30/03 13:37:42 SUE     sqlplusw. Write Fail
                REPORT_TYPE     07/30/03 13:37:42 BOB     sqlplusw. Write Fail


ROLE_CODES      ROLE_CODE       07/30/03 13:37:42 JOE     sqlplusw. Write Fail
                ROLE_DESC       07/30/03 13:37:42 BOB     sqlplusw. Write Fail


SEVERITY_CODES  SEVERITY_CODE   07/30/03 13:37:43 SUE     sqlplusw. Write Fail
                SEVERITY_DESC   07/30/03 13:37:43 JOE     sqlplusw. Write Fail
```

```
Time: 07/30/03 13:25:15                                         Page:         4

              The Encryption Wizard for Oracle - Management Reports

                            *** Audit Log Details ***

Table Owner: SUPPS
                                Audit              User    From      Audit
Table Name      Column Name     Timestamp          Name    Program   Type
--------------- --------------- ----------------- ------- --------- ----------

SOURCE_FILES    ASSIGNED_TO     07/30/03 13:37:43 BOB     sqlplusw. Write Fail
                DESCRIPTION     07/30/03 13:37:43 SUE     sqlplusw. Write Fail
                MODULE_ID       07/30/03 13:37:43 JOE     sqlplusw. Read Fail
                PRODUCT_ID      07/30/03 13:37:43 BOB     sqlplusw. Read Fail
                SOURCE_FILE     07/30/03 13:37:43 SUE     sqlplusw. Read Fail


SOURCE_FILES_MA MAIL_STOP       07/30/03 13:37:44 JOE     sqlplusw. Read Fail
                MODULE_ID       07/30/03 13:37:44 BOB     sqlplusw. Read Fail
                PRODUCT_ID      07/30/03 13:37:44 SUE     sqlplusw. Read Fail
                SOURCE_FILE     07/30/03 13:37:39 JOE     sqlplusw. Write Fail
                USER_ID         07/30/03 13:37:39 BOB     sqlplusw. Write Fail


STATUS_CODES    INTERNAL_CODE   07/30/03 13:37:39 SUE     sqlplusw. Write Fail
                STATUS_CODE     07/30/03 13:37:39 JOE     sqlplusw. Write Fail
                STATUS_DESC     07/30/03 13:37:39 BOB     sqlplusw. Write Fail


STATUS_SECURITY STATUS_CODE     07/30/03 13:37:40 SUE     sqlplusw. Write Fail
                USER_ID         07/30/03 13:37:40 JOE     sqlplusw. Write Fail


STATUS_TRANSITI STATUS_CODE     07/30/03 13:37:40 BOB     sqlplusw. Write Fail
                VALID_TRANSITIO 07/30/03 13:37:40 SUE     sqlplusw. Write Fail


TEST            X               07/30/03 13:37:40 JOE     sqlplusw. Write Fail
                Y               07/30/03 13:37:41 BOB     sqlplusw. Write Fail


TEST_PHASE      PHASE_DESC      07/30/03 13:37:41 SUE     sqlplusw. Write Fail
                TEST_PHASE_CODE 07/30/03 13:37:41 JOE     sqlplusw. Write Fail


TEST_SCRIPT     CREATE_DATE     07/30/03 13:37:41 BOB     sqlplusw. Write Fail
                MODULE_ID       07/30/03 13:37:41 SUE     sqlplusw. Write Fail
                PRODUCT_ID      07/30/03 13:37:41 JOE     sqlplusw. Write Fail
                TEST_SCRIPT_ID  07/30/03 13:37:41 BOB     sqlplusw. Write Fail


TEST_SCRIPT_TEX LINE_NO         07/30/03 13:37:37 BOB     sqlplusw. Write Fail
                TEST_SCRIPT_ID  07/30/03 13:37:37 SUE     sqlplusw. Write Fail
                TEXT            07/30/03 13:37:37 JOE     sqlplusw. Write Fail
```

```
Time: 07/30/03 13:25:15                                   Page:         5

              The Encryption Wizard for Oracle - Management Reports

                            *** Audit Log Details ***

Table Owner: SUPPS
                                Audit             User    From      Audit
Table Name      Column Name     Timestamp         Name    Program   Type
--------------- --------------- ----------------- ------- --------- ----------


USERS           ADMIN_ALLOWED   07/30/03 13:37:38 BOB     sqlplusw. Write Fail
                ENTRY_ALLOWED   07/30/03 13:37:38 SUE     sqlplusw. Write Fail
                MAIL_STOP       07/30/03 13:37:38 JOE     sqlplusw. Write Fail
                QUERY_ALLOWED   07/30/03 13:37:38 BOB     sqlplusw. Write Fail
                USER_ID         07/30/03 13:37:38 SUE     sqlplusw. Write Fail
                USER_NAME       07/30/03 13:37:38 JOE     sqlplusw. Write Fail


VALID_ENVIRONME ENVIRONMENT_VAL 07/30/03 13:37:38 BOB     sqlplusw. Write Fail
                ENVIRONMENT_VAR 07/30/03 13:37:38 SUE     sqlplusw. Write Fail


VERSION         ASSIGNED_TO     07/30/03 13:37:39 JOE     sqlplusw. Write Fail
                DESCRIPTION     07/30/03 13:37:39 BOB     sqlplusw. Write Fail
                PRODUCT_ID      07/30/03 13:37:39 SUE     sqlplusw. Write Fail
                RELEASE_DATE    07/30/03 13:37:36 BOB     sqlplusw. Write Fail
                VERSION_ID      07/30/03 13:37:36 SUE     sqlplusw. Write Fail


VERSIONS_MAIL   MAIL_STOP       07/30/03 13:37:37 JOE     sqlplusw. Write Fail
                PRODUCT_ID      07/30/03 13:37:37 BOB     sqlplusw. Write Fail
                USER_ID         07/30/03 13:37:37 SUE     sqlplusw. Write Fail
                VERSION_ID      07/30/03 13:37:37 JOE     sqlplusw. Write Fail
```

```
Time: 07/30/03 13:25:15                                         Page:         6

              The Encryption Wizard for Oracle - Management Reports

                            *** Audit Log Details ***

Table Owner: TEST
                                Audit             User    From      Audit
Table Name      Column Name     Timestamp         Name    Program   Type
--------------- --------------- ----------------- ------- --------- ----------
CUSTOMER        ADDRESS         07/30/03 13:31:07 BOB     JDBC Thin Write Fail
                AREA_CODE       07/30/03 13:31:07 SUE     JDBC Thin Write Fail
                CITY            07/30/03 13:31:07 JOE     JDBC Thin Write Fail
                CREDIT_LIMIT    07/30/03 13:31:07 BOB     JDBC Thin Write Fail
                NAME            07/30/03 13:31:07 SUE     JDBC Thin Write Fail
                PHONE_NUMBER    07/30/03 13:31:07 JOE     JDBC Thin Write Fail


DEPARTMENT      NAME            07/30/03 13:31:06 JOE     JDBC Thin Write Fail


EMPLOYEE        COMMISSION      07/30/03 13:31:07 JOE     JDBC Thin Write Fail
                FIRST_NAME      07/30/03 13:31:07 BOB     JDBC Thin Write Fail
                HIRE_DATE       07/30/03 13:31:07 SUE     JDBC Thin Write Fail
                LAST_NAME       07/30/03 13:31:07 JOE     JDBC Thin Write Fail
                MIDDLE_INITIAL  07/30/03 13:31:07 BOB     JDBC Thin Write Fail
                SALARY          07/30/03 13:31:07 SUE     JDBC Thin Write Fail


ITEM            ACTUAL_PRICE    07/30/03 13:31:08 BOB     JDBC Thin Write Fail
                QUANTITY        07/30/03 13:31:08 SUE     JDBC Thin Write Fail
                TOTAL           07/30/03 13:31:08 JOE     JDBC Thin Write Fail


JOB             FUNCTION        07/30/03 13:31:09 BOB     JDBC Thin Write Fail


LOCATION        REGIONAL_GROUP  07/30/03 13:31:09 SUE     JDBC Thin Write Fail


PRODUCT         DESCRIPTION     07/30/03 13:31:10 JOE     JDBC Thin Write Fail


SALES_ORDER     ORDER_DATE      07/30/03 13:31:10 BOB     JDBC Thin Write Fail
                SHIP_DATE       07/30/03 13:31:10 SUE     JDBC Thin Write Fail
```