SafeNet Database Connector User Guide for Oracle

Version 4.8.3
Documentation Version: 20090413

Copyright Information

Table of Contents

About This Guide

This introductory chapter gives a brief summary of the book's contents, identifies the audience, explains how to best use the written material, and discusses the documentation conventions used.

This chapter contains the following information:

Using This Guide

Generally speaking, SafeNet user guides are written for network administrators, security engineers, database administrators, application developers, and other technology professionals responsible for daily operations in support of data security. The written material we provide describes how to configure and operate SafeNet's products and assumes a working knowledge of networking, computer security, database management, and cryptography.

This specific book is designed for advanced database administrators familiar with Oracle databases, the DataSecure appliance, and the general system architecture of the SafeNet DataSecure Platform as documented in the DataSecure Appliance User Guide.

Organization

This user guide is divided into the chapters and appendices described below:

Chapter 1, "Overview"

Presents an overview of how a SafeNet Database Connector integrates with your current application. Topics discussed include migrating data, automating encryption and decryption, rotating crypto keys, and batch encryption and decryption.

Chapter 2, "Installing the Database Connector"

Details the installation process.

Chapter 3, "Configuring the Properties File"

Details the process of modifying the properties file.

Chapter 4, "Planning Database Migration"

Discusses what to consider before migrating your database to an encrypted solution.

Chapter 5, "Migrating Data"

Details how to migrate plaintext data from unencrypted to encrypted columns.

Chapter 6, "Managing Encrypted Data"

Details data management tasks such as key rotation, view and trigger creation, domain index management, unencryption, and modifying encrypted tables.

Chapter 7, "Using the Bulk Loader"

Provides examples of bulk encryption and decryption in an Oracle environment.

Chapter 8, "Setting up SSL"

Explains how to set up both the NAE Server and the database connector to use SSL.

Appendix A, "Database Objects"

Lists the database objects delivered with the SafeNet Database Connector for Oracle.

Appendix B, "Troubleshooting"

Provides troubleshooting tips for the SafeNet Database Connector for Oracle.

Documentation Conventions

This section describes the formatting conventions used in this manual to explain code samples, special notes, and cautions.

Code Samples

Sample code is illustrated in the format shown below. The code is provided with comments (prefaced by /*) to give you an understanding of how the functions interact.

```
DECLARE     retStat NUMBER;
         retTxt VARCHAR2(255);

BEGIN       ing_bulk_load_pkg.ing_bulk_load
         ('name_of_job', /* Job Name            */
         'N',            /* Resume flag         */
         retStat,        /* Return code         */
         retTxt          /* Return code text    */
         );

END;
```

Notes and Cautions

The following paragraph formats are used to highlight information in the text:

Note: A note conveys information that supplements the preceding text. This information may refer to certain situations or a specific technical setup.

Important! An important note is a very significant piece of information required for the completion of a task.

Tip: A tip helps you apply the information in the preceding text.

WARNING! A warning advises you to exercise care when working around specified equipment conditions. Heeding a warning can prevent personal injury, system disruption, or equipment damage.

CHAPTER 1

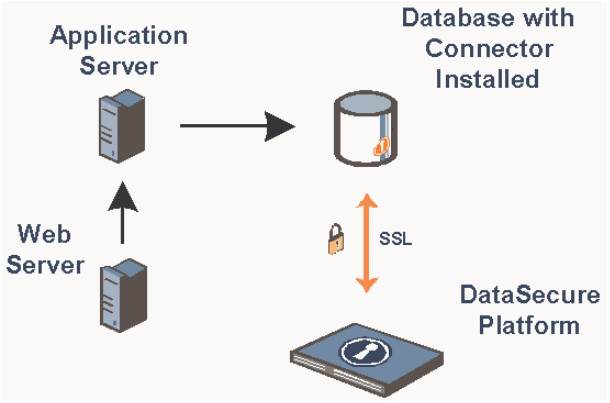
Overview

The Database Connector allows you to manage a seamless interaction between the database and the DataSecure. The DataSecure enables you to migrate tables, automatically encrypt data for insert statements and decrypt data for select statements, rotate keys, and encrypt and decrypt large batches of data.

This chapter covers the following topics:

Using the Database Connector

The Database Connector allows your database to communicate directly with an NAE Server and transparently encrypt and decrypt data. This diagram gives a high-level overview.



Here, a web server accesses an application server, which requests encrypted data from the database. The database, in turn, uses the Database Connector to authenticate users and submit cryptographic requests to the DataSecure. The DataSecure then decrypts the data and returns the plaintext values.

Integrated with this process are various levels of security that protect your data from unauthorized users. To access the features needed to view your decrypted data, users and applications must have permission to:

- Send a request from the database to the DataSecure appliance.
- Access the requested key on the DataSecure.

If either of these conditions is not met, then the sensitive data is not accessed.

Supported Platforms

The Database Connector is supported on a variety of Oracle databases (Oracle 8i, 9i, 10gR1, and 10gR2) on a number of host platforms, including AIX, HPUX, Linux, Solaris, and Windows.

For specific database and environment combinations, please see the softwar download page on our Customer Support web site.

Supported Browsers

The Database Tools can be used with the following browsers:

- Internet Explorer 5.5 and above
- Recent Mozilla-based browsers, including Netscape 7, Mozilla 1.x, and Firefox 0.9 and above

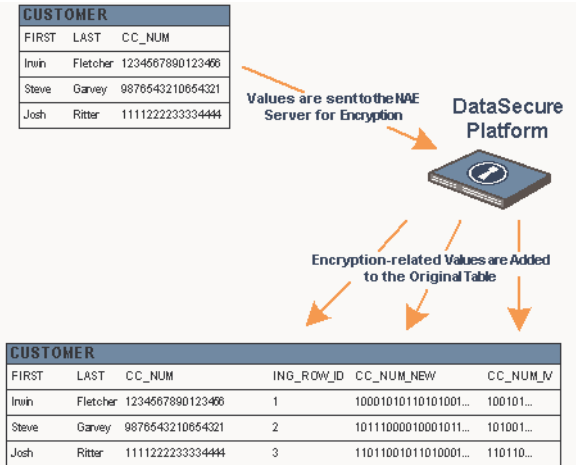
Migrating Tables

Migrating a table involves sending a column's plaintext values to the NAE Server for encryption and returning the resulting ciphertext back to the database. To set up this process you must:

- Register the database and table with the NAE Server.
- Select the column(s) to be encrypted.
- Select the encryption method.

The actual data migration is an interaction between the database table, the Database Connector, and the encryption capabilities of the NAE Server.

This diagram shows how data migration effects the base table.



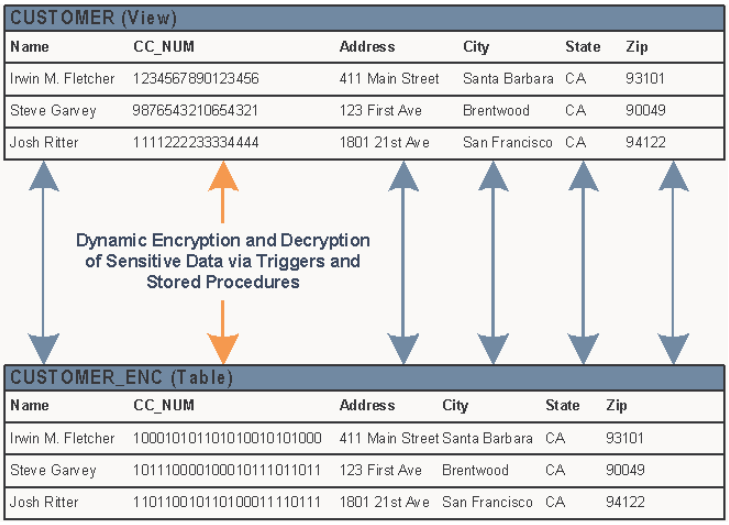
During the data migration process, the system:

- Adds an empty column to hold encrypted values (column_NEW) to the original table.
- Adds an identity column, if necessary.
- Adds an empty column to hold initialization vectors (column_IV) to the original table, if you are applying IVs at the field level.
- Sends the column values to the DataSecure appliance for encryption.
- Returns the encrypted values to the column_NEW column in the original table.
- Returns the initialization vectors to the column_IV column.

Important! We *strongly* suggest that you remove the original plaintext from your table once the data migration process is complete. Use the Management Console to remove the plaintext values.

Automating Encryption and Decryption Operations

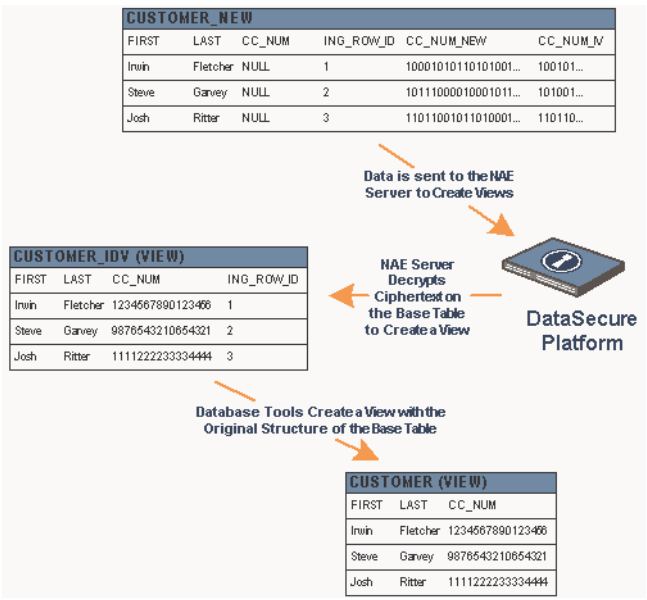
After migrating your data, you can create views and triggers to automate future encryption and decryption operations. These views and triggers use stored procedures to interact with the NAE Server behind the scenes to perform cryptographic operations on the original table without explicit instructions from the database user. Authenticated applications outside the database can query and update the tables as before, without any modification. The figure shows the relationship between the view and the base table.



Creating Views

When the system creates views for an encrypted table, referred to as the base table, it does the following:

- **Renames the base table.** The default name is `<table>_NEW`. This prevents applications from directly referencing the base table.
- **Creates a new view** that decrypts the ciphertext on the base table. The default name is `<table>_IDV`. This view calls functions to authenticate the user, access the NAE Server, and decrypt the data.
- **Creates a new view** by selecting columns from `<table>_IDV`. *The new view has the original structure of the base table* and does not have the `ING_ROW_ID` and `IV` columns.



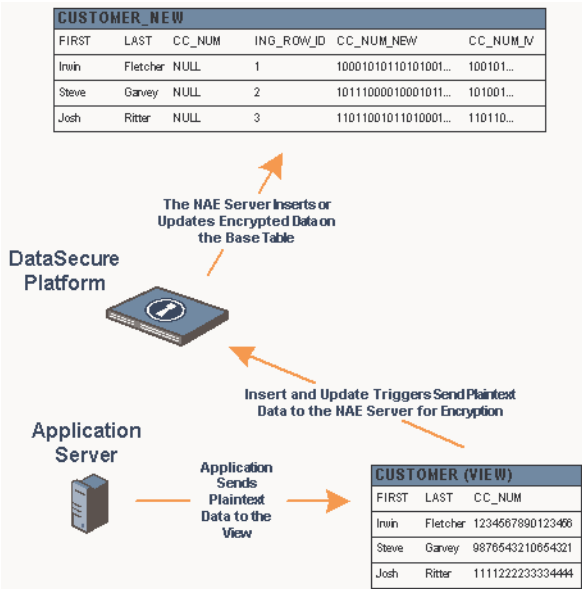
Once the views are created, all authorized applications that reference the original table are now, unknowingly, accessing a view, which in turn, uses the cryptographic properties of the NAE Server.

Creating Triggers

When the system creates triggers for an encrypted table, it does the following:

- **Creates an insert trigger** on <table>_IDV. The default name is <table>_INS_TRIG. This trigger inserts encrypted data into the base table when plaintext data is inserted into the view.
- **Creates an update trigger** on <table>_IDV. The default name is <table>_UPD_TRIG. This trigger updates the encrypted data in the base table by encrypting the plaintext data updated through the view.

Once the triggers are created, all authorized users that insert or update data are actually, interacting with the DataSecure, which is encrypting the data sent to the view and forwarding it to the base table.

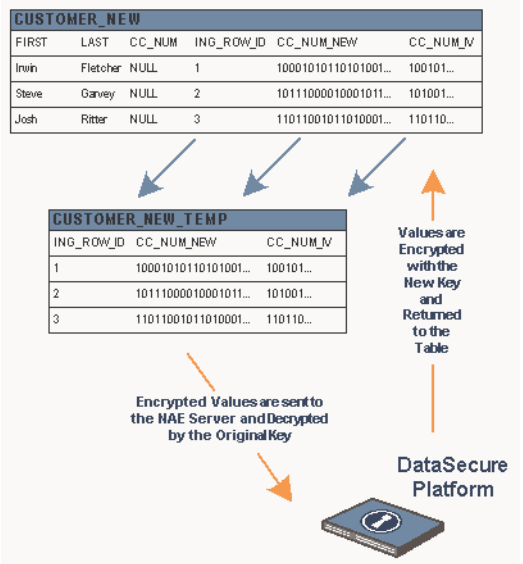


Rotating Cryptographic Keys

Although encryption provides a high level of data security, it is possible that a skilled attacker could compromise a key. The best way to limit the effect of this attack is to rotate keys. *Key rotation should be a regular part of your security maintenance plan.*

The key rotation process uses the original table to create a temporary table populated with the ciphertext, row ID, and IV (if needed). The NAE Server decrypts the ciphertext on the temporary table, re-encrypts it with a new key, and returns the new values to the base table.

The new key should have the same permissions as the old key and create encrypted data the same size as the old ciphertext.



Important! We *strongly* suggest that you remove the temporary table once the key rotation process is complete. You can use the Management Console to remove the temporary table. Please refer to “Table Operations”.

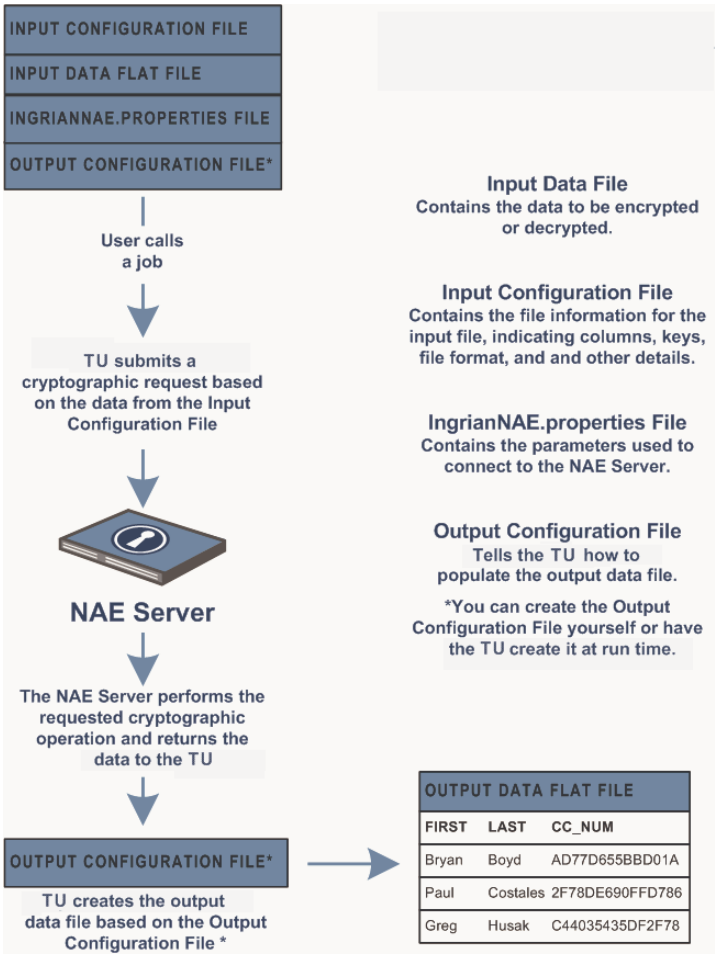
Batch Encryption and Decryption

The DataSecure Platform provides two options for doing batch processing: the Transform Utility, and the Bulk Loader. The Transform Utility extends the cryptographic features of the DataSecure Platform to flat files, which enables you to convert files between encrypted and cleartext formats at extremely fast rates, independent of the database. The Bulk Loader, on the other hand, uses a set of stored procedures to initiate cryptographic operations from within the database. Because the Transform Utility can process a significantly greater number of operations per second, it is recommended that you use the Transform Utility for your batch encryption and decryption needs.

The Transform Utility

When processing a request, the Transform Utility operates on the specified columns in an input data file and populates an output data file with the resulting data. Four files are sent to the Transform Utility at runtime: an input data file, an input configuration file, an output configuration file, and the `IngrianNAE.properties` file.

- The *input data file* contains the data to be encrypted or decrypted. The file can contain delimited or positional data. Each line of data must end with a carriage return.
- The *input configuration file* tells the Transform Utility how to use the input data file; indicating which columns to encrypt or decrypt, which keys to use, and how to read the information in the data file. This file can be in either XML or flat file format.
- The *output configuration file* tells the Transform Utility how to populate the output data file.



You need not maintain the same formatting as the input data file. For example, if your input data file stores comma-delimited values, you can specify that the output data file use positional formatting.

The file can be in either XML or flat file format. You have three options to create the output configuration:

- Create the output configuration yourself
- Have the Transform Utility create one for you to review prior to the data transformation
- Have the Transform Utility create the file and continue with the cryptographic operation

All of these options are available at runtime.

- The *IngrianNAE.properties file* contains the parameters used to connect to the NAE Server. The Transform Utility also applies the logging parameters, such as log level and log file location, from this file.

▶ For more information about the Transform Utility, see the *Transform Utility User Guide*.

The Bulk Loader

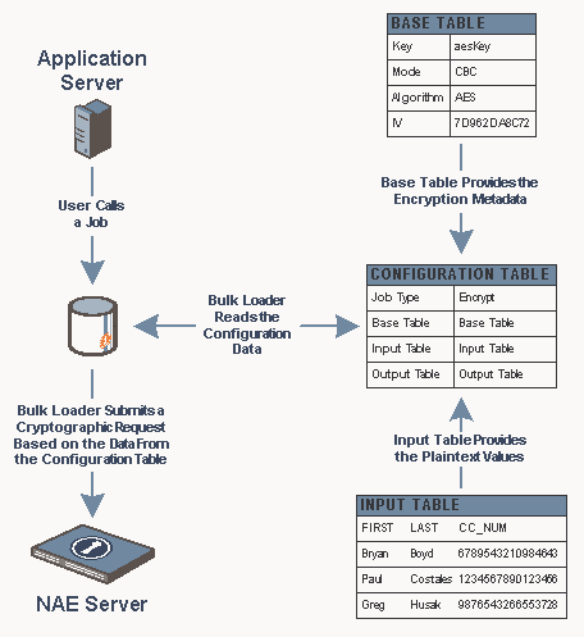
The Database Connector includes a set of stored procedures collectively referred to as the Bulk Loader. The Bulk Loader is optimized for performing cryptographic operations on large amounts of data.

The following figure illustrates how the Bulk Loader sends data to the NAE Server when a preconfigured job is called.

The Bulk Loader applies the encryption metadata from a previously–encrypted table, called the base table, to a new cryptographic operation. For each operation there is also an input table that holds the input data, and an output table that stores the results.

The combination of base table, input table, output table, and cryptographic operation are stored with a unique job name. Jobs are configured and run separately, so that configurations can be saved and re–used.

The NAE Server then returns the resulting data to the Bulk Loader which uses it to populate the output table.



CHAPTER 2

Installing the Database Connector

This chapter contains detailed instructions on how to install and set up the SafeNet Database Connector for Oracle. This chapter contains the following information:

Preparing for the Installation

Prior to installing the Database Connector, you must:

- Download and extract the database tools file.
- Configure the shared library search path.
- Modify the properties file.
- Permit the installer to create directories and symbolic links.
- Create the database user `INGRIAN` and allocate tablespace.
- Set the Oracle home environment.
- Turn on the 32-bit listener, if appropriate.
- Reconcile time zone settings, if appropriate.
- Make sure that the script can be opened by your shell.
- Configure Oracle to support our external procedures.

Once you have completed these tasks, you can run the installation program. For more information, see “Installing the Database Connector” on page 22.

Downloading and Extracting the DB Tools File

You must obtain the Database Tools file appropriate for your platform from our website.

Configuring the Shared Library Search Path

The Database Tools files include the shared libraries used by the Database Connector. You must add the location of these libraries to the correct environment variable.

In AIX environments, the environment variable is `LIBPATH`. You can set the variable using the following command:

```
export LIBPATH=$ORA_HOME/lib:$ORA_HOME/lib/ingrian:/usr/lib:$LIBPATH
```

In Linux/UNIX environments, the environment variable is `LD_LIBRARY_PATH`.

In Solaris-64 environments, if both `LD_LIBRARY_PATH` and `LD_LIBRARY_PATH_64` are defined, add the location of the Ingrian libraries to `LD_LIBRARY_PATH_64` only.

Modifying the Properties File

The `IngrianNAE.properties` file (referred to as “the properties file”) allows you to determine how NAE clients interact with NAE Servers. By modifying a set of parameters, you can specify which NAE Server a client connects to, which protocol is used to establish a connection, how frequently logs are rotated, and so on. Some basic configuration information, such as the server IP address, port, and log file are required to install the SafeNet Database Connector for Oracle.

In Linux and Unix environments, the installation script prompts you for the required information.

In Windows environments, you must, at a minimum, modify the values for the NAE_IP, NAE_Port, and Protocol in the properties file. This file is included in the file you downloaded in the step above. It gets extracted to the `/Ingrian/DBTools-oracle` directory.

If You Modify the Properties File After Installation

Important! If you modify the properties file after running the installer, you must reload the file by calling the `loadProperties` script.

- In Windows, call: `loadProperties.bat`
- In Linux and UNIX environments, call: `loadProperties.sh`

Permitting the Installer to Create Directories and Symbolic Links

The installer attempts to create a new directory called `<Oracle_Home>/lib/ingrian` and creates symbolic links in the directory `<Oracle_Home>/lib`. Please ensure that the operating system user running the installer has permission to perform these operations.

Creating the Database User INGRIAN

Before installing the Database Connector, the `INGRIAN` user must exist in your database. You must either verify that the `INGRIAN` user already exists or create it.

Important! Our code is case-sensitive; the `INGRIAN` username must be capitalized.

To create the db user `INGRIAN` and allocate the necessary tablespace, enter these commands:

```
Enter user-name: sys/<oracle-system-user-password> as sysdba
SQL> create user INGRIAN identified by INGRIAN
2 default tablespace users
3 quota unlimited on users;
User created.
SQL> @grantPermission.sql INGRIAN
```

Note: If you see an error messages, fix the problem before continuing.

The last line above grants privileges to the `INGRIAN` user. To see the list of those privileges, look at the `grantpermission.sql` file.

Setting the Oracle Home Environment

If there are multiple Oracle homes on the machine where you want to install the Database Connector, then the `ORACLE_HOME` environment variable needs to be set **before** running `install.bat` (Windows) or `install.sh` (Linux/UNIX).

Turning on the 32-bit Listener

If you are installing the SafeNet Database Connector for Oracle on Oracle 9i (64-bit) on an HP-UX 11.00 system, make sure that the 32-bit listener is on.

Note: When you install Oracle 9i (64-bit) on HP-UX 11.00, if you opt to have the external stored procedure listener on, by default only the 64-bit listener is on.

Reconciling Time Zone Settings

When deploying the Database Connector on Oracle 8i on Solaris 9, the database TIME_ZONE setting must be consistent with the time zone used by the operating system. If there is a disparity between the two, it is likely that once a table has been migrated, subsequent inserts into the encrypted DATE column are offset by the value of the database TIME_ZONE setting.

Checking the Shell

The install script assumes that the bourne shell is installed in `/bin/sh`. If it is not installed there, modify the script accordingly.

Configuring Oracle to Support Our External Procedures

For an Oracle database to support the SafeNet, Inc. external procedures, you must configure a new ExtProc listener.

Note: For more on this topic, please refer to Document 70638.1 at metalink.oracle.com.

To configure a new ExtProc listener:

1. Add the following code to the listener.ora file:

```
SID_LIST_INGRIAN =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = IngrianEXP)
      (ORACLE_HOME = <your-ORACLE-HOME>)
      (PROGRAM = extproc)
      (ENVS="LD_LIBRARY_PATH=/export/home/ingrian/4.8/DBTools-
oracle/lib"
    )
  )

INGRIAN =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = IngrianKey))
    )
#
# The following information is included to help you view
# Ingrian-related information to your Oracle logs.
LOG_DIRECTORY_INGRIAN=<your-ORACLE-HOME>
LOG_FILE_INGRIAN=Ingrian_listener.log
TRACE_LEVEL_INGRIAN=off
TRACE_DIRECTORY_INGRIAN=<your-ORACLE-HOME>
TRACE_FILE_INGRIAN=Ingrian_listener.trc
```

   Note: You can use any value for KEY but, make certain that the value specified here is the same value specified below, in the tnsnames.ora file.

2. Remove any other references to external procedures in any existing listeners.
3. Reconfigure the tnsnames.ora file.

The tnsnames.ora file must include information about the external procedure listener. *Update the file so that the external procedure section includes the key and SID referenced in the listener.ora file.*

Note: Depending on your settings, you may need to use `EXTPROC_CONNECTION_DATA.yourDomain` instead of `EXTPROC_CONNECTION_DATA`.

Add the following code to the tnsnames.ora file:

```
EXTPROC_CONNECTION_DATA =
  (DESCRIPTION =
      (ADDRESS = PROTOCOL = IPC)(KEY =IngrianKey))
    (CONNECT_DATA =
      (SID = IngrianEXP)
      (PRESENTATION =RO)
    )
  )
```

Note: If you have specified a value for the `NAMES.DEFAULT_DOMAIN` parameter in your sqlnet.ora file, then this value should be included when configuring the tnsnames.ora file.

4 Start the new listener with the following command:

```
% lsnrctl start INGRIAN
```

where `INGRIAN` is the alias used in step 1 above.

Note: If you change the `LD_LIBRARY_PATH`, you must restart the Oracle listener.

Installing the Database Connector

To install the Database Connector, you must:

- Select the installation mode.
- Run the installation script.
- Restart the Oracle listener.

Note: You cannot install the Database Connector remotely; you must install it from the machine where the database server resides.

Selecting the Installation Mode

The installation script can be run in different modes for different purposes.

The available modes are:

Mode	Description
install	The install mode assumes that you have not previously installed the Database Connector for Oracle on this machine. If you have an existing installation and you run the script with the install option, the existing installation is overwritten. In install mode, the script does the following: • Installs tables, stored procedures, and user defined functions in a schema called `INGRIAN`. • Copies required shared libraries to `<Oracle_Home>/lib/ingrian`. • Creates symbolic links to the Ingrian shared libraries and `<Oracle_Home>/lib/ingrian`. • Generates and loads an Instance ID for this installation.
reload	Reload mode does everything that install mode does, except: • the metadata tables are not regenerated, and • an Instance ID is not generated. Reload mode is quite useful to load updated files you receive from SafeNet. If you are upgrading from a previous version of the Database Connector for Oracle, you should use this option.
localonly	This option is used to install the shared libraries in clustered database environments.
uninstall	This option uninstalls the Oracle Database Provider. **WARNING!** You cannot uninstall the Database Connector until you have unencrypted all encrypted columns. For more information on unencrypting data, please refer to “Unencrypt Columns” on page 98.

Running the Installation Script

To run the installation script:

1. Open a command prompt window.
2. Navigate to the directory where the script is located.
3. Enter one of the following commands, depending on your operating system.

Operating System	Command
Windows	`install.bat <mode>`
UNIX	`install.sh <mode>`

Where `<mode>` is either install, clean, localonly, reload, or uninstall.

Note: The localonly flag requires an additional parameter to specify whether your Oracle server is 32- or 64-bit. The command might look like this:

```
install.sh localonly <32 | 64>
```

4 Enter the following information when prompted:
   - Passwords for the sys and `INGRIAN` users.
   - Enter an instance ID or accept a randomly generated default, when prompted. Databases that share our metadata must use the same instance ID.
   - NAE Server IP address and port (Linux/Unix environments only).
   - Path to the properties file. You must have read access to this file.
   - Full path and file name for the log file (Linux/Unix environments only).

Important! The path you enter for the log file must *not* include symbolic links, and it must specify a file name. On some operating systems, the use of symbolic links causes the Management Console to display the following error message:

```
The provider has not yet been installed on this database.
You must install the provider before you can map users.
```

Restarting the Oracle Listener

After you have completed the installation process, you should restart the Oracle listener. If you have another listener process for external procedures, you should restart that as well.

To restart the Oracle listener, use the commands below:

```
lsnrctl stop
lsnrctl start
```

Installing on Oracle 10gR2 RAC Environment

These installation instructions contain the following information regarding Database Connector for Oracle Version 4.3.1 on Oracle 10gR2 RAC environments:

- Prerequisites
- Installation and Configuration on Oracle 10gR2 RAC Environment

Prerequisites

Before installing the SafeNet Database Connector for Oracle on Oracle 10gR2 RAC environments, perform the following system checks to confirm that the Oracle 10gR2 RAC environment is properly configured and is operational:

- RAC Status
- Cluster Daemons Status
- Database Status
- Service Status
- Listener Status

Important! The DataSecure appliance should not become a single point of failure for the entire cluster. Therefore, it is essential in a cluster installation to have DataSecure appliances clustered as well. For information and procedures to create a NAE cluster, see “NAE Cluster Configuration” in the DataSecure Appliance User Guide.

RAC Status

Check the status of RAC applications by running the following command at the UNIX shell prompt while logged in as an oracle user on any node:

```
[oracle@rac1 ~]$ crs_stat -t
```

The following sample output is obtained from a two-node cluster with Transparent Application Failover (TAF) service configured on the first node:

Name	Type	Target	State	Host
ora....SM1.asm	application	ONLINE	ONLINE	rac1
ora....PA.lsnr	application	ONLINE	ONLINE	rac1
ora.rac1.gsd	application	ONLINE	ONLINE	rac1
ora.rac1.ons	application	ONLINE	ONLINE	rac1
ora.rac1.vip	application	ONLINE	ONLINE	rac1
ora....SM2.asm	application	ONLINE	ONLINE	rac2
ora....PS.lsnr	application	ONLINE	ONLINE	rac2
ora.rac2.gsd	application	ONLINE	ONLINE	rac2
ora.rac2.ons	application	ONLINE	ONLINE	rac2
ora.rac2.vip	application	ONLINE	ONLINE	rac2
ora....GTAF.cs	application	ONLINE	ONLINE	rac1
ora....db1.srv	application	ONLINE	ONLINE	rac1
ora.devdb.db	application	ONLINE	ONLINE	rac2
ora....b1.inst	application	ONLINE	ONLINE	rac1
ora....b2.inst	application	ONLINE	ONLINE	rac2

You can also check the status of RAC components individually.

Cluster Daemons Status

Run the following commands as an oracle user on any node:

- `$ srvctl status nodeapps -n hostname`
- `[oracle@rac1 ~]$ srvctl status nodeapps -n rac1`

  Sample output:

  ```
  VIP is running on node: rac1
  GSD is running on node: rac1
  Listener is running on node: rac1
  ONS daemon is running on node: rac1
  ```

- `[oracle@rac1 ~]$ srvctl status nodeapps -n rac2`

  Sample output:

  ```
  VIP is running on node: rac2
  GSD is running on node: rac2
  Listener is running on node: rac2
  ONS daemon is running on node: rac2
  ```

Database Status

Run the following command as an oracle user on any node:

```
$ srvctl status database -d dbname
[oracle@rac1 ~]$ srvctl status database -d devdb
```

Sample output:

```
Instance devdb1 is running on node rac1
Instance devdb2 is running on node rac2
```

Service Status

To check the status of the Transparent Application Fail Over service, if one is configured in your environment, run the following command as an oracle user on any node:

```
$ srvctl status service -d dbname -s servicename
[oracle@rac1 ~]$ srvctl status service -d devdb -s ingtaf
```

Sample output:

```
Service ingtaf is running on instance(s) devdb1
```

Listener Status

The Database Connector for Oracle requires that the external procedure listener be configured and running in order to operate.

To verify the listener configuration, run the following command as an oracle user on each node:

```
$ lsnrctl stat
[oracle@rac1 ~]$ lsnrctl stat
```

Sample output:

```
LSNRCTL for Linux: Version 10.2.0.1.0 - Production on 06-APR-2007 12:26:43
Copyright (c) 1991, 2005, Oracle.  All rights reserved.
Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
STATUS of the LISTENER
------------------------------------------
Alias                     LISTENER_RAC1
Version                   TNSLSNR for Linux: Version 10.2.0.1.0 - Production
Start Date                04-APR-2007 13:52:55
Uptime                    1 days 22 hr. 33 min. 48 sec
Trace Level               off
```

```
Security            ON: Local OS Authentication
SNMP                OFF
Listener Parameter File   /u01/app/oracle/product/10.2.0/racdb/network/
admin/listener.ora
Listener Log File         /u01/app/oracle/product/10.2.0/racdb/network/
log/listener_rac1.log
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=192.168.2.31)(PORT=1521))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=172.17.11.53)(PORT=1521))
Services Summary...
Service "+ASM" has 1 instance(s).
  Instance "+ASM1", status BLOCKED, has 1 handler(s) for this service.
Service "+ASM_XPT" has 1 instance(s).
  Instance "+ASM1", status BLOCKED, has 1 handler(s) for this service.
Service "INGTAF" has 1 instance(s).
  Instance "devdb1", status READY, has 2 handler(s) for this service..
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for this ser-
vice...
Service "devdb" has 2 instance(s).
  Instance "devdb1", status READY, has 2 handler(s) for this service..
  Instance "devdb2", status READY, has 1 handler(s) for this service..
Service "devdbXDB" has 2 instance(s).
  Instance "devdb1", status READY, has 1 handler(s) for this service..
  Instance "devdb2", status READY, has 1 handler(s) for this service..
Service "devdb_XPT" has 2 instance(s).
  Instance "devdb1", status READY, has 2 handler(s) for this service..
  Instance "devdb2", status READY, has 1 handler(s) for this service..
The command completed successfully
```

Confirm that the **PLSExtProc** external procedure listener is running on every node.

Installation and Configuration on Oracle 10gR2 RAC Environment

Installation on Oracle 10gR2 RAC Environment installation requires the following steps:

- Preparing the Installation Directory
- Installing the Database Connector for Oracle on the Primary Node
- Installing the Database Connector for Oracle on Other Nodes
- Configuring the Database Connector for Oracle
- Testing the NAE Connection
- Configuring SSL and Client Certificates

Preparing the Installation Directory

The Database Connector must be installed on every node of the cluster. The primary node requires the full installation; all other nodes can be installed with the **local_only** option.

Tip: If you have the Oracle Cluster File System (OCFS) or another shared storage, such as NFS, configured in your environment, you'll want to use a path on this storage for your temporary installation directory. This eliminates the need to transfer the installation package to every node.

Download and expand the installation package into an installation directory.

Installing the Database Connector for Oracle on the Primary Node

For detailed information and procedures required to install the Database Connector, see "Preparing for the Installation" on page 18 and "Installing the Database Connector" on page 22.

Run the installer on the primary node using the following command:

```
$ ./install.sh install
```

Respond to all installer prompts and wait for the installer to complete before performing installations on other nodes.

Installing the Database Connector for Oracle on Other Nodes

Run the following command to install the Database Connector for Oracle on all other nodes.

```
$ ./install.sh localonly 64
```

Important! Complete the installation on each node before installing the next one.

This step installs shared libraries required for the Database Connector on nodes other than the primary node.

Configuring the Database Connector for Oracle

Connect to the NAE server configured for your database.

Create a new database connection to the primary node of the cluster. You use this connection to administer user access, migrate and unmigrate tables, create views and trigger, and related tasks.

You do not need to configure database connections to any other nodes in the cluster. Only a single connection to the primary node is required.

MAP USERS

Using the newly configured connection, establish appropriate user mappings to map database/application users to NAE users.

At this point you have completed the Database Connector configuration and you should be able to encrypt and decrypt your data.

Testing the NAE Connection

Perform a connection test from the client database machine to make sure that it can communicate with the NAE server. Run the following from SQL*PLUS while connected to any node, the database itself, or a fail over service:

```
[oracle@rac1 ~]$ sqlplus qatest@ingtaf
```

Sample output:

```
SQL*Plus: Release 10.2.0.1.0 - Production on Fri Apr 6 12:48:18 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bit
Production
With the Partitioning, Real Application Clusters, OLAP and Data Mining
options
SQL> select ingrian.ing_gn_rndm(16) from dual;
INGRIAN.ING_GN_RNDM(16)
--------------------------------------------------------------------
86FA0B18B16114117094B73516679A60
SQL> Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.1.0 - 64bit Production
With the Partitioning, Real Application Clusters, OLAP and Data Mining
options
```

If random bytes were successfully generated, the database server is correctly configured to communicate with the NAE server.

If you experience errors while executing this function, it means that your ext proc listener is not configured properly, your user mapping is not valid, or there is a problem communicating with the NAE server. Resolve any problems encountered in this step before attempting to migrate tables.

Configuring SSL and Client Certificates

For detailed information and procedures required to set up SSL, see Chapter 8, “Setting up SSL”.

To set up the SSL connection and client certificates, use one of the following configuration options:

- Have the Certificate Authority File, Certificate File, and Key File reside on a file system that is shared by all cluster nodes, such as NFS or OCFS.
- Have separate copies of Certificate Authority File, Certificate File, and Key File on every node.

Important! These files must be accessible from each node using the same absolute path.

Upgrading to Future Software Versions

After you have installed the current version of the Database Connector software, you may later need to perform an upgrade. The two kinds of upgrades are:

- Version upgrades. This is an upgrade to a new version of the software.
- Patch upgrades. This is an upgrade to a new patch release.

To upgrade to either a new version or a patch release, run the installer in reload mode.

Setting Up Specific Features

Once the Database Connector software is installed, you may want to set up specific features. This section explains the following processes:

- Enabling Logging at the Database Level
- Limiting User Access to Ingrian Objects

Enabling Logging at the Database Level

To enable logging at the Oracle database level:

1. Enter the log file name in the Log_File parameter in the Logging Configuration section of the `IngrianNAE.properties.sql` file.

   Important! Remember that if your log file is in a Unix/Linux file system, you cannot include symbolic links in the path.

2. Run the loadProperties script from the Ingrian\DBTools-oracle folder. Enter one of the following commands, depending on your operating system.

Operating System	Command
Windows	`loadProperties.bat`
Linux and UNIX	`loadProperties.sh`

3. When prompted, enter the `INGRIAN` user password.

Limiting User Access to Ingrian Objects

You can run the revokePermission script to limit the users who have access to Ingrian objects. If you do not run this script, all database users can view and potentially modify the Ingrian tables.

To run the revokePermission script:

1. Open a command prompt window.
2. Navigate to the directory where the script is located.
3. Enter the following command:

```
revokePermission <sys_user> <target_user>
```

Where <target_user> is the user to be denied permission.

For example, you can enter the following command to remove public access to the Ingrian objects:

```
revokePermission sys public
```

Note: This script does not affect the public ability to configure and run the Bulk Loader.

Uninstalling the Database Provider

To uninstall the Database Connector:

1. Use the Management Console to unencrypt all of the previously-migrated tables.
2. Use the Management Console to delete all of the associated database connections.
3. Run the install script with the uninstall flag.

   Enter one of the following commands, depending on your operating system.

Operating System	Command
Windows	`install.bat uninstall`
UNIX	`install.sh uninstall`

The uninstall process then drops or deletes the following Ingrian items:

- Stored procedures.
- User-defined functions.
- Shared libraries.

CHAPTER 3

Configuring the Properties File

This chapter lists the contents of the IngrianNAE.properties file. The properties file defines, among other things, the IP address, port and protocol of the DataSecures to which your client connects. This chapter contains the following topics:

Editing the Properties File

The values in the properties file are case-sensitive. yes is not YES. tcp is not TCP. Follow the example of the default properties file.

You can comment-out values using #. You'll see that the properties file is delivered with both **EdgeSecure_Name** and **Cipher_Spec** commented-out. You may want to use comments to save settings when troubleshooting. For example, you could store commonly used NAE_IP addresses like this:

```
NAE_IP=10.0.0.2
#NAE_IP=10.0.0.3
#NAE_IP=10.0.0.4
```

When editing parameters that use time values, you can use the following abbreviations:

- ms - milliseconds. e.g. 4500ms for 4.5 seconds.
- s - seconds. e.g. 30s for 30 seconds.
- m- minutes. e.g. 5m for 5 minutes.
- h - hours. e.g. 10h for 10 hours.
- d - days. e.g. 2d for 2 days.

If you do not include an abbreviation, the default time unit is used. For most time-related values the default is milliseconds. For **Symmetric_Key_Cache_Expiry** and **Persistent_Cache_Expiry_Keys**, the default is seconds.

Reloading the Properties File

If you modify any of the values in the properties file after installing the Database Connector, you must reload the properties file by calling the `loadProperties` script.

In Windows, call:

```
loadProperties.bat
```

In Linux and UNIX environments, call:

```
loadProperties.sh
```

The Parameters

Once you install the client software, you can customize it to meet the needs of your environment by modifying the properties file. The parameters listed in the file, including the delivered settings, are shown below.

Version=2.4
NAE_IP=
NAE_Port=9000
Protocol=tcp

Use_Persistent_Connections=yes
Size_of_Connection_Pool=300
Connection_Timeout=60000
Connection_Idle_Timeout=600000
Connection_Retry_Interval=600000
Unreachable_Server_Retry_Period=60000
Maximum_Server_Retry_Period=0
Cluster_Synchronization_Delay=100
#EdgeSecure_Name=

#Cipher_Spec=HIGH:!ADH:!DH:!DSA:!EXPORT:RSA+RC4:RSA+DES:RSA+AES
CA_File=
Cert_File=
Key_File=
Passphrase=

Symmetric_Key_Cache_Enabled=no
Symmetric_Key_Cache_Expiry=43200

Persistent_Cache_Enabled=no
Persistent_Cache_Directory=
Persistent_Cache_Expiry_Keys=43200
Persistent_Cache_Max_Size=100

Log_Level=MEDIUM
Log_File=
Log_Rotation=Daily
Log_Size_Limit=100k

Version

The **Version** parameter indicates the version of the properties file and should not be modified.

NAE_IP

The **NAE_IP** parameter specifies the IP address of the DataSecure.

Port

The **NAE_Port** specifies the port of the DataSecure. The default port is 9000.

Important! Clients and servers must use the same port.

Protocol

The **Protocol** specifies the protocol used to communicate between the client and the DataSecure.

Possible settings:

- **tcp**
- **ssl** - The ssl option uses TLSv1. By default, TLSv1 is enabled on all DataSecures. *If you have disabled the use of TLSv1 on your servers, then you cannot establish SSL connections with between your* NAE clients *and servers.*

Important! Clients and servers must use the same protocol. If your DataSecures are listening for SSL requests, and your clients aren't sending SSL requests, you will run into problems.

Tip: *We recommend that you gradually increase security after confirming connectivity* between the client and the DataSecure. Once you have established a TCP connection between the client and server, it is safe to move on to SSL. Initially configuring a client under the most stringent security constraints can complicate troubleshooting.

Use_Persistent_Connections

The **Use_Persistent_Connections** parameter enables the persistent connections functionality.

Possible settings:

- **yes** - Enables the feature. The client establishes persistent connections with the NAE Servers. This is the default value.
- **no** - Disables the feature. A new connection is made for each connection request. The connection is closed as soon as the client receives the server response.

Size_of_Connection_Pool

The **Size_of_Connection_Pool** parameter is the total number of client-server connections that your configuration could possibly allow. (Not what actually exists at a given moment.)

Possible settings:

- **Any positive integer** - The default is 300.

Connections in the pool can be active or waiting, TCP or SSL. A connection is created as needed, and the pool scales as needed. The pool starts at size 0, and can grow to the value set here. Once the pool is full, new connection requests must wait for an existing connection to close.

Connection pooling is configured on a per-client basis. The size of the pool applies to each *client*, it is not a total value for a DataSecure or for a load balancing group. If there are multiple clients running on the same machine, separate connection pools are maintained for each client.

Connection_Timeout

The **Connection_Timeout** parameter specifies how long the client waits for the TCP connect function before timing out.

Possible settings:

- **0** - disables this setting. The client uses the operating system's connect timeout.
- **Any positive integer** - The default is 60000ms.

Setting this parameter a few hundred ms *less* than the operating system's connection timeout makes connection attempts to a downed server fail faster, and failover happens sooner. If a connection cannot be made before the timeout expires, the server is marked as down and taken out of the rotation.

Note: If your client is working with many versions of a key, do not set the **Connection_Timeout** parameter too low. Otherwise the client connection may close before the operation is complete.

Connection_Idle_Timeout

The **Connection_Idle_Timeout** parameter specifies the amount of time connections in the connection pool can remain idle before the client closes them.

Possible settings:

- **Any positive integer** - The default is 600000ms (10 min).

Important! There are two different connection timeout values: *one on the DataSecure*, and *one in the properties file*. The value of the timeout in the properties file must be less than what is set on the server. This lets the client control when idle connections are closed. Otherwise, the client can maintain a connection that is closed on the server side, which can lead to error.

Connection_Retry_Interval

The **Connection_Retry_Interval** parameter determines how long the client waits before trying to reconnect to a disabled server. If one of the DataSecures in a load balanced configuration is not reachable, the client assumes that the server is down, and then waits for the specified time period before trying to connect to it again.

Possible settings:

- **0** - This is the infinite retry interval. The disabled server will never be brought back into use.
- **Any positive integer** - The default value is 600000ms (10 minutes).

Unreachable_Server_Retry_Period

The **Unreachable_Server_Retry_Period** parameter specifies the amount of time the client will spend attempting to establish a connection to a load balancing group. An error is returned after the specified period if no server in the group is reachable. If logging is enabled, error messages are written to the log file.

Possible settings:

- **-1** - This is the infinite retry interval. The client keeps trying to connect to a server until a connection is established. This setting is not compatible with multi-tier load balancing because the load balancer will never switch tiers.
- **A positive integer** - If multi-tier load balancing is enabled then set this value between 1 and twice the value of the **Connection_Retry_Interval**. The default is 60000ms.

Maximum_Server_Retry_Period

The **Maximum_Server_Retry_Period** parameter specifies the total amount of time that the client will spend trying to make connections to any server on any tier. This value is only used when multi-tiered load balancing is enabled.

Possible settings:

- **-1** - The connection manager will try every server on every tier until one answers.
- **0** - The feature is disabled; there is no overall limit.
- **A positive value** - The connection manager will try to make connections for at least the duration set in **Maximum_Server_Retry_Period** and will return an error if no connection can be made on the tier in use when the try period expires.

Cluster_Synchronization_Delay

The **Cluster_Synchronization_Delay** parameter specifies how long the client will wait before assuming that key changes have been synchronized throughout a cluster. After creating, cloning, importing, or modifying a key, the client will continue to use the same DataSecure appliance until the end of this delay period.

Possible settings:

- **0** - disables the function.
- **Any positive integer.** The default is 100s. You may want a higher setting for large clusters.

For example, the client sets **Cluster_Synchronization_Delay** to 100s and sends a key creation request to appliance A, which is part of a cluster. Appliance A creates the key and automatically synchronizes with rest of the cluster. The client will use only appliance A for 100 seconds - enough time for the cluster synchronization to complete. After this time period, the client will use other cluster members as before.

EdgeSecure_Name

The client uses the **EdgeSecure_Name** parameter to communicate with an EdgeSecure.

Possible settings:

- **Name of an EdgeSecure**
- **Filename** - *the first line of the file must contain the EdgeSecure name*. Using a file enables you to use the same IngrianNAE.properties file for all of your clients and still maintain unique EdgeSecure names for each.

This parameter is tier-aware to allow for failover. Three tiers are allowed: EdgeSecure_Name.1, EdgeSecure_Name.2, and EdgeSecure_Name.3. The third tier *must* be a DataSecure.

Note: This failover feature is not associated with the Multi-Tier Load Balancing feature.

Cipherspec

The **Cipher_Spec** parameter specifies which SSL/TLS protocol and encryption algorithms to use. Multiple cipher strings can be separated by colons.

For example, the value

```
HIGH:!ADH:!DH:!DSA:!EXPORT:RSA+RC4:RSA+DES:RSA+AES
```

specifies the following:

- do NOT use anonymous Diffie-Helman (!ADH), Diffie-Helman (!DH), nor DSA (!DSA)
- use high strength ciphers, RSA+RC4, RSA+DES, and RSA+AES

Note: The default entry is commented out in the properties file; this is because this parameter is compiled into the client library. **You should only modify this parameter if you prefer to use some other combination of algorithms and protocols.** Modifying this parameter overrides the value in the library.

Important! If you specify some value other than the default, you must use RSA for key exchange.

CA_File

The **CA_File** parameter refers to the CA certificate that was used to sign the server certificate presented by the NAE Server to the client.

Possible settings:

- **The path and filename** - The path can be absolute or relative to your application. Don't use quotes, even if the path contains spaces.

Because all DataSecures in a clustered environment must have an identical configuration, all servers in the cluster use the same server certificate. As such, you only need to point to one CA certificate in the **CA_File** system parameter. If you do not supply the CA certificate that was used to sign the server certificate used by the DataSecures, your client applications cannot establish SSL connections with any of the servers in the cluster.

If a local CA on the DataSecure was used to sign the NAE Server certificate, you can download the certificate for the local CA, and put that certificate on the client.

Cert_File

The **Cert_File** parameter stores the path and filename of the client certificate. This is only used when your SSL configuration requires clients to provider a client certificate to authenticate to the DataSecures.

Possible settings:

- **The path and filename** - The path can be absolute or relative to your application. Don't use quotes, even if the path contains spaces. Client certificates *must* be PEM encoded.

Important! If this value is set, the certificate and private key must be present, even if the DataSecure is not configured to request a client certificate.

Key_File

The **Key_File** parameter refers to the private key associated with the client certificate specified in the **Cert_File** parameter.

Possible settings:

- **The path and filename** - The path can be absolute or relative to your application. Don't use quotes, even if the path contains spaces. The client private key must be in PEM-encoded PKCS#12 format.

Because this key is encrypted, you must use the **Passphrase** parameter so the DataSecure can decrypt it.

Important! If this value is set, the certificate and private key must be present, even if the DataSecure is not configured to request a client certificate.

Passphrase

The **Passphrase** parameter refers to the passphrase associated with the private key.

Possible settings:

- **The passphrase associated with the private key named in Key_File**

If you do NOT provide this passphrase, the client attempts to read the passphrase from standard input; this causes the application to hang.

Important! Remember that the properties file is NOT encrypted. Make sure that this file resides in a secure directory and has appropriate permissions so that it is readable only by the appropriate application or user.

Symmetric_Key_Cache_Enabled

The **Symmetric_Key_Cache_Enabled** parameter determines if the symmetric key caching feature is enabled. *Only symmetric keys can be cached.*

Possible settings:

- **no** - Key caching is disabled. Remote encryption (encryption performed on the DataSecure) is available as normal.
- **yes** - Key caching is enabled. **Protocol** must be set to ssl. (And ssl must be configured.)
- **tcp_ok** - Key caching is enabled over both tcp and ssl connections.

Symmetric_Key_Cache_Expiry

The **Symmetric_Key_Cache_Expiry** parameter determines the *minimum* amount of time that a key will remain in the client key cache.

Possible settings:

- **0** - This is the infinite timeout setting. Keys are never purged from the client cache.
- **A positive integer** - At the end of this interval, the key will be purged from the cache the next time the library is called. The default value is 43200 **seconds** (12 hours).

Persistent_Cache_Enabled

The **Persistent_Cache_Enabled** parameter determines if the persistent key caching feature is enabled.

Possible settings:

- **yes** - The feature is enabled. To enable this feature, you must also enable symmetric key caching: **Symmetric_Key_Cache_Enabled** must be set to yes or tcp_ok.
- **no** - The feature is disabled. This is the default setting.

Persistent_Cache_Directory

The **Persistent_Cache_Directory** parameter determines where the provider will create the persistent cache file.

Possible settings:

- **The path to the directory that will contain the keys** - The directory must already exist. The path can be absolute or relative to your application. Don't use quotes, even if the path contains spaces.

Persistent_Cache_Expiry_Keys

The **Persistent_Cache_Expiry_Keys** parameter determines the number of seconds after which a key may be removed from the cache. To enable the persistent cache, this value must be greater than zero.

Possible settings:

- **0** - This is the infinite timeout setting. Keys are never purged from the cache.
- **Any positive integer** - At the end of this interval, the key will be purged from the cache the next time the library is called. The default value is 43200 **seconds** (12 hours).

Persistent_Cache_Max_Size

The **Persistent_Cache_Max_Size** parameter determines the maximum number of keys that can be stored in the persistent cache.

Possible settings:

- **0** - disables the feature.
- **Any positive integer** - The default value is 100 keys.

Log_Level

The **Log_Level** parameter determines the level of logging performed by the client.

Possible settings:

- **NONE** – disables client logging. We recommend that you not disable logging.
- **LOW** – only error messages are logged.
- **MEDIUM** – the client logs error messages and warnings.
- **HIGH** – the client logs error messages, warnings and informational messages. This level generates a very large number of entries and is usually reserved for debugging.

Important! The user running your client application must have permission to write to the log file, and to create new files in the directory where the log files are created.

Log_File

The **Log_File** parameter specifies a name (and possibly a path) for the log file.

Possible settings:

- **a filename** - The log will be created in the same directory as the client. The default value is Logfile.txt.
- **a path and filename** - The path can be absolute or relative to your application. Don't use quotes, even if the path contains spaces.

Use the **INGRAN_LOGFILE_SUFFIX** environment variable to create individual log files for each client application. When the variable is set, its value is appended to the value set in the **Log_File** parameter. When a unique value is set for each client application, each client gets its own logfile.

For example, if your **Log_File** parameter is set to /foo/Logfile and your application's **INGRIAN_LOGFILE_SUFFIX** is set to app1, then that log file will be written to /foo/Logfile.app1. Then, if you set **INGRIAN_LOGFILE_SUFFIX** to app2 in a second application, that log file will appear in /foo/bar/Logfile.app2.

Similarly, the **INGRIAN_LOGFILE_PREFIX** environment variable enables you to prepend the value set in the **Log_File** parameter. To create a log file in /tmp/application1/Logfile.txt, you would set **INGRIAN_LOGFILE_PREFIX** to /tmp/application1/ and accept the **Log_File** default.

Important! Do not include soft links (or symbolic links) in the path you specify for the **Log_File** parameter. The presence of soft links in this path will invariably lead to problems when trying to create user mappings.

Log_Rotation

The **Log_Rotation** parameter specifies whether logs are rotated daily or once they reach a certain size.

Possible settings:

- **Daily** - Rotates logs daily. This is the default.
- **Size** - Rotates logs when the reach the size specified in **Log_Size_Limit**.

Log_Size_Limit

The **Log_Size_Limit** parameter specifies how large log files can be before they are rotated. This parameter is used only when **Log_Rotation** is set to Size.

Possible settings:

- **Any positive integer** - The default unit is bytes. You can use the suffix k (or K) for kilobytes and m (or M) for megabytes. The default value is 100k.

Reading System Properties From the Windows Registry

By default, the client reads the system properties from the properties file; however, you can configure the provider to read the system properties from the system registry instead. This option is only available in Microsoft Windows environments.

When the provider is launched, it searches the registry for the ConfigFilename key, which contains the location of the properties file. If the file is found, the provider stores the data and uses it while the client application is running. If the properties file is not found, the provider searches the registry for the individual parameters. (If no keys are found, the client application cannot run.)

For the provider to read values from the registry rather than the properties file, you must rename the IngrianNAE.properties file (to hide it) and you must create the appropriate registry keys.

As part of the Database Connector, we distribute a sample registry configuration. You can use this sample, or you can manually create string values for the system values you want to set. Be aware that using the sample registry file overwrites any existing values in the registry.

Important! You do not have to create a string value for each property in the properties file; however, you must set the **Version** parameter correctly. If you are not sure what value to provide, check the properties file that shipped with the provider.

Setting Properties in the Registry via the Sample Configuration

1. Navigate to the directory into which you installed the SafeNet Cryptographic Provider. The default directory is C:\Program Files\Ingrian.
2. Open the `SampleRegistryConfig.reg` file in a text editor.

   The sample configuration file includes only a subset of the available system properties. The following system properties are set with the values listed below:

```
"Version"="2.0"
"NAE_IP"="192.168.200.223"
"NAE_Port"="9000"
"Protocol"="tcp"
"Use_Persistent_Connections"="yes"
"Size_of_Connection_Pool"="300"
"Log_Level"="HIGH"
```

3. Modify the parameters in the file according to the needs of your deployment. Add appropriate system properties if necessary.
4. Save your changes and close the file.
5. Double-click on `SampleRegistryConfig.reg` to create the necessary registry keys.

Manually Setting Properties in the Registry

To set the system properties in the registry manually, create a new string value for the system properties relevant to your deployment in the following location:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Ingrian\
```

For the IP address of the DataSecure, for example, you would create a new string value called NAE_IP and you would assign it a value equal to the DataSecure's IP address. Remember, if you are specifying multiple DataSecures in a load balancing group, you only need to create one registry key. Simply separate IP addresses with a colon.

CHAPTER 4

Planning Database Migration

This chapter contains information that is critical to a successful data migration operation. It is strongly recommended that you read all the material in this chapter and consider the trade–offs when presented. This chapter contains the following topics:

Planning Your Database Migration

Migrating from columns with plaintext data to columns with encrypted data is an important task that requires planning and collaboration between the database administrator and the security engineer. If you are doing complete database integration, this process should be transparent to all applications that interact with the database, meaning that you will not have to make changes to applications that access your sensitive information. If you are integrating at the application level, then you might have to make some changes to the application.

Whichever method of integration you implement, you must consider what data to encrypt, how to encrypt it, and how to manage the encrypted data.

Selecting the Data to Encrypt

When selecting the data to be encrypted, you must consider the character type, data type, column properties, and the length of the plaintext column.

Supported Character Types

The DataSecure Appliance supports the encryption of single-byte, multi-byte, and Unicode data.

Supported Data Types

The following table lists the data types supported by the data migration process:

- CHAR
- DATE
- DECIMAL
- NCHAR
- NUMBER (supported with specified precision and scale, up to 31 digits)
- NUMERIC
- NVARCHAR2
- VARCHAR
- VARCHAR2
- LONG (you cannot create views and triggers in tables that include LONG columns)
- LONG RAW (you cannot create views and triggers in tables that include LONG RAW columns)

Any data types that do not appear in this list cannot be encrypted.

Column Encryption Guidelines

The ability to encrypt a column depends on the relationship between the column and its table. Below is a list of roles that columns can play and their effect on encryption.

- **Identity column** – Cannot be encrypted.
- **Primary key** – Primary keys are dropped during migration. You must manually recreate primary keys if you want to preserve the conditions established by the primary keys. If the primary key is not referenced in a foreign key constraint, verify that the key is not referenced implicitly as a foreign key before encrypting.
- **Foreign key** – To encrypt a foreign key you must drop the constraints prior to data migration. After migration, you can re-establish them.
- **Indexed columns** – Indexed columns can be encrypted, however, the sort order of the encrypted data will not be consistent with the sort order of the plaintext data.

You should also evaluate the constraints placed on your columns, as these values may affect the data migration process. Below is a list of constraints and their effect on encryption.

- **Join constraints** – Confirm that the columns you are encrypting are not part of a join constraint. If you are encrypting a column that is part of a join constraint, you should encrypt both columns.

- **Unique constraints** – When encrypting a column with a unique constraint, that constraint is dropped during the data migration process. If you want to retain the unique constraint after encryption, you should manually recreate the unique constraint. You cannot use field-level IV on a column with unique constraints. Instead, you should choose one IV for the entire column.
- **Check constraint** – To encrypt a column, you must drop any check constraints.

Additional rules apply to the following topics:

- **Default values** – Columns with a default value assigned to them cannot be encrypted. This is because the default constraint adds plaintext data to the column. Applications accessing that data then try to decrypt plaintext data, yielding unexpected results.
- **`NULL` values** – `NULL` values are not encrypted by the Database Connector. If a migrated column contains `NULL` values, those values remain unencrypted in the resulting encrypted column. When a database query yields a `NULL` value, no cryptographic process is required, so the Database Connector does not interact with the NAE Server for that query.
- **Columns referenced in triggers on the table** – These columns can be encrypted; however, the DBA should review all triggers that reference the column to ensure that the trigger functions as expected. No warning is given by the system when you migrate a column with a trigger on it.
- **Encrypted columns** – The columns that are currently encrypted cannot be encrypted.
- **Tables containing `LONG` or `LONG RAW` columns** – If a table in an Oracle database contains a column of type `LONG` or `LONG RAW`, you can migrate data in that table; however, you cannot create views and triggers against this table, due to a limitation in Oracle. This is an important consideration if you want to automate subsequent insert, update, and select calls on the encrypted data.

Length of the Plaintext Column

The SafeNet Database Connector for Oracle uses the `RAW` data type to store ciphertext. `RAW` columns cannot exceed 2000 bytes. If the padding process would create a ciphertext larger than 2000 bytes, encryption is not allowed.

Using AES, the Database Connector does not allow the encryption of plaintext values larger than 1999 bytes.

The following table illustrates how padding affects column length when migration is performed with AES keys.

Plaintext Column Length (bytes)	Resulting Ciphertext Column Length (bytes)
15	16
16	32
17	32

Plaintext Column Length (bytes)	Resulting Ciphertext Column Length (bytes)
1983	1984
1984	2000
1985	2000
1999	2000
2000	2016 (encryption not allowed)
2001	2016 (encryption not allowed)

Using DES/DESede, the Database Connector does not allow the encryption of plaintext values larger than 1999 bytes.

The following table illustrates how padding affects column length when migration is performed with DES/DESede keys.

Plaintext Column Length (bytes)	Resulting Ciphertext Column Length (bytes)
7	8
8	16
9	16
1991	1992
1992	2000
1993	2000
1999	2000
2000	2008 (encryption not allowed)
2001	2008 (encryption not allowed)

Deciding How to Encrypt Your Data

When deciding how to encrypt your data, you must consider which algorithm to use, what mode to use, how to apply initialization vectors, whether to use padding, select replacement values, and determine how much space the resulting data requires.

Choosing an Encryption Algorithm

You can use any of the following encryption algorithms:

- AES (key sizes of 128, 192, and 256 bits).
- DESede (key sizes of 112 and 168 bits).
- SEED (key sizes of 128 bits)
- DES (key size of 56 bits).

We recommend that you use an AES key (any size) or a 168 bit DESede key, as these ciphers are stronger than the others. It is recommended that you *not* use DES keys because DES is considered to be a weak cipher.

The encryption algorithm you select limits the size of the plaintext data that you can encrypt. For more information, see “Length of the Plaintext Column” on page 46.

Applying Initialization Vectors

When using an algorithm in CBC mode, you may apply an Initialization Vector (IV) at the field-level or at the column-level. When applying IVs at the field-level, a unique IV is used for the encryption of each field. In this case, a new column is added to your table. When you apply IVs at the column-level, there is only one IV per column, and that IV is stored in a separate metadata table.

You might prefer to apply IVs at the field–level if you are encrypting values that might be identical. For example, if you are encrypting names, and two people in the table have the same name, those names encrypt to the same value if you supply the same IV for both encrypt operations. If, however, you supply a different IV for the two encrypt operations, then the encrypt operations yield different results. Doing field-level encryption allows you to achieve an even higher level of security because the IVs are different for every value being encrypted.

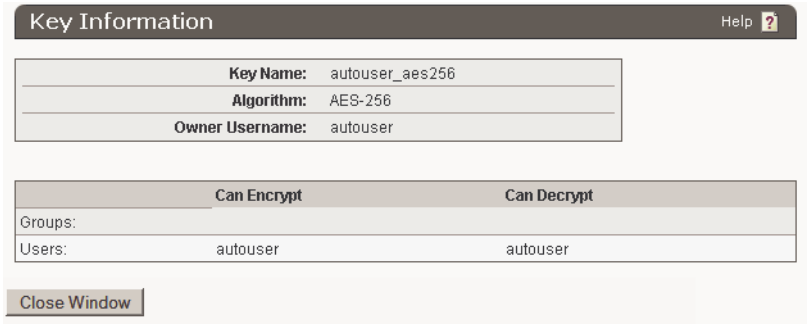
However, you should be aware that applying IVs at the field-level might cause a significant performance disadvantage when doing an exact search. When you apply IVs at the field-level, you cannot encrypt the search value, which means that you must decrypt all rows you are searching through. If you are encrypting values that you know are unique, like credit card numbers or social security numbers, it is recommended that you apply IVs at the column-level.

Your options for the IV field of the Column Properties section are as follows:

- user–specified IV for column – You supply a single IV for an entire column.
- random IV for column – The NAE Server provides one randomly generated IV that is used to encrypt all values in a column.
- random IV for each field – The NAE Server provides a randomly generated IV for each value in a column. With this option, the NAE Server adds a column for the IV to the base table.

Key Information

The Key Information section displays the name of the key, key algorithm, and the key owner. The window also warns against using global keys for data migration because global keys are available to all NAE users.



The following table describes the components of the Key Information section.

Component	Description
Name	Displays the name of the key used to encrypt and decrypt data in this column.
Algorithm	Displays the name of the algorithm used with this key.
Owner	Displays the key owner. If the key is global, there is no associated owner. **Tip:** As a security precaution, we recommend that you *do not* use global keys for database encryption.
Can Encrypt	Lists the groups and users that can use this key for encryption.
Can Decrypt	Lists the groups and users that can use this key for decryption.
Close Window	Click **Close Window** to close the Key Information page and return to the Column Properties section.

Padding

For symmetric keys, the NAE Server automatically selects a padding scheme during the encryption process. One of two methods are used:

- PKCS#5 Padding – This scheme is almost always used when encrypting with symmetric keys (AES, DESede, and DES).
- No Padding – This scheme is used when the column's data type is `CHAR` or `CHARACTER` and the column's original width is an exact multiple of the encryption block size.

Replacement Values

If a database user attempts to access encrypted data to which they do not have decryption permission, the system returns an error message. You can specify the content of those permission–related errors using the replacement values feature.

Note: The replacement value is only used if the user has *no* decryption permissions. If an error results from the user's authorization policy (i.e., the decryption request occurs outside of the usage period, or exceeds the maximum operations per hour) the actual error is returned. This is done for two reasons. First, returning a policy violation error helps alert you of a possible attack. Second, returning this error ensures that the same query always yields the same results. If a query could return either the replacement value *or* legitimate plaintext, without any way of indicating which was returned, data in the client application could be corrupted.

You can use the **Decryption Behavior for Users with Insufficient Permissions** field on the Column Properties section to specify that the system return a specific replacement value, a `NULL` value, or the original standard error.

By using the replacement value together with the default user mapping, you can ensure that all unauthorized database users receive the same message when attempting to access encrypt

Important! A user with select, update, and delete privileges can delete data from a migrated table based on the error replacement values.

Note: Replacement values are not returned if a query yields a `NULL` value. When a query results in a `NULL` value, no cryptographic process is required, so the database tools do not interact with the NAE Server and the replacement values feature is not activated.

For more information about Default User Mapping, see "Default User Mapping" on page 52.

Space Requirements Considerations

Some of the things you must consider when alloting space for an encrypted solution are: the algorithm used to encrypt data, whether you use field-level or column-level IVs, and the addition of an identity column.

Managing the Encrypted Data

Prior to migrating your database you must consider how you will manage the encrypted data. The sections below discuss ciphertext data types, user mapping, identity columns, space requirements, and changes to the original column.

Ciphertext Data Types

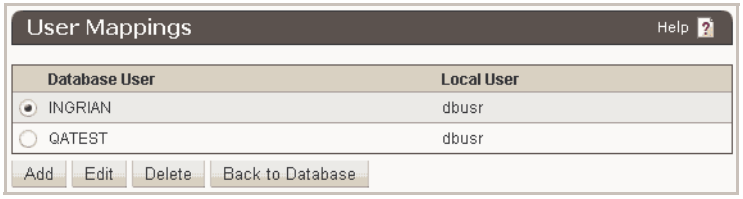
The data type used to store your ciphertext varies by database. The SafeNet Database Connector for Oracle stores ciphertext as `RAW`.

Managing User Mappings

A user mapping is an association between a database user and an NAE user. The Ingrian Database Tools rely on these user mappings when accessing the NAE Server.

In addition to managing your encrypted data, you must also determine who can access that information. The user mapping feature enables you to associate a database user with an NAE user account, or with a default user account. The Database Connector uses this user mapping to authenticate the database user with the NAE Server before submitting cryptographic requests.

Important! User names for user mappings are case-sensitive.



The Database Connector authenticates to the NAE Server as the NAE user mapped to the database user who is performing the operation. (If the database user is valid, but is not specifically listed, the default mapping value is used, when enabled.) If the user mapping contains a valid NAE user and password, and that NAE user has permission to access the key required for the operation, then the request is honored. However, if either of the conditions above is not met, then the operation is not performed.

You must have a valid user mapping to create triggers and views through the Management Console, regardless of database environment.

In Oracle environments, data migration, unencryption, and key rotation do not require a user mapping.

You may want to enable some features for all database users not otherwise listed on the User Mappings section. To do this you would associate the Default Mapping value with a specific NAE user. For example, you could create an NAE user with access to global keys, or you could create an NAE user with access to no specific permissions in order to enable the replacement value feature.

For more information about Replacement Values, see “Replacement Values” on page 49.

Important! If you change user mappings, Oracle databases require that you establish a new connection to the database for the changes to take effect.

Default User Mapping

The default NAE user is a catch-all NAE user that is used to connect to the NAE Server when no user mapping exists for a database user that is trying to access encrypted data. When there is no default user mapping, and an unmapped database user attempts to access sensitive data, the database connector returns an error message and does not send the request to the NAE Server.

It is useful to create a default user mapping to prevent the database connector from automatically returning this error. When an unmapped database user attempts to access sensitive data, instead of returning an error message, the Database Connector connects to the NAE Server as the default NAE user. How the NAE Server responds to request at that point depends on how the NAE Server is configured. The NAE Server might return an “insufficient permissions” error, it might return NULL, or it might return a pre–configured replacement value.

When the Default Mapping database user is assigned, the system creates an entry on the `ING_AUTHORIZED_USER` table where the database user is * (an asterisk). For this reason, you should avoid using an asterisk to represent a specific database user.

WARNING! Although the default NAE user can be used for both encryption and decryption operations, it is strongly recommended that the default NAE user have no key or group permissions. The point of creating a default user mapping is to gracefully handle requests for encrypted data from database users who are not authorized to view that data. If the default NAE user has key or group permissions, you are potentially allowing unauthorized database users to view sensitive data.

Identity Columns

In Oracle databases, the `ING_ROW_ID` column is added to the base table. This column enables the various batch operations that can be performed in the Ingrian DataSecure environment.

The Size of Encrypted Data

Encrypted data is predictably larger than plaintext data, so it is important that you make sure you have enough table space for the operation you are performing (data migration, unencryption, or key rotation). During the data migration process, a new column is created. The original column might have a length of 50 bytes; the new column, however will have a different length. The length of the new column depends on which algorithm you use to encrypt your data, and is displayed on the Column Properties page.

For columns encrypted with symmetric keys (AES, DESede, and DES), the new column size increases to the next multiple of the block size. Remember that DES and DESede use block sizes of 8 bytes, and AES uses a block size of 16 bytes. So, if the original column was 50 bytes, then the new column is 56 bytes for DES and DESede; for AES, the new column is 64 bytes.

If the length of the original column is a multiple of the block size, then the new column is the original column size plus the block size. For example, if a column contained a 56 byte string, and that string was encrypted with a DESede key, then the ciphertext is 64 bytes long.

If that same 50 byte column were encrypted with an AES key, the new column would require 64 bytes. In addition, if you are using a random IV for each field, then your table has another column of either 8 bytes (DES and DESede) or 16 bytes (AES).

Consider a scenario where you have a column that is 50 bytes in length and you want to encrypt that column with a DESede key. There are 10,000 rows in the column, and you've decided to use a unique IV for each value.

The original column is set to `NULL` and a new column is created. The new column is 56 bytes in length. Before encryption, the column required 500,000 bytes. After encryption, the column requires 560,000 bytes. Additionally, a new column is created to hold the IVs. The IVs are 8 bytes each, so an additional 80,000 bytes of space is required for the IVs.

Note: When you add a domain index to a CHAR or VARCHAR column, the DESCRIBE function will return the maximum size of a VARCHAR2 column (4000) regardless of the actual data size. Column size does not change when a domain index is added.

CHAPTER 5

Migrating Data

This chapter contains step–by–step instructions on how to migrate plaintext data from unencrypted to encrypted columns. This chapter contains the following information:

Database Configuration Page

On the Database Configuration page, you can view the databases that connect to the DataSecure appliance and add a database. Before any data can be migrated, you must first add the database through the Management Console. Once added, the new entry shows up in the list of databases.

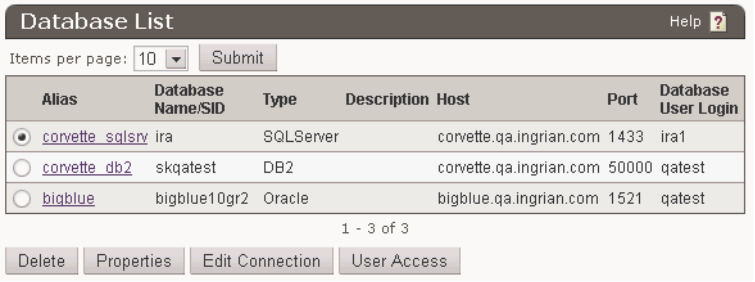
Note: When you are making configuration changes on the Database Tools page, database information is being accessed over your network connection. It might take a few seconds for data or schema to be returned to the NAE Server; this is normal. Please be patient if it seems like these pages load noticeably slower than other pages on the Management Console. You should avoid clicking on items when the UI seems unresponsive, as this might log you out of the system.

The Database Configuration page contains the following sections:

- Database List
- Add a Database
- Confirm Database Login
- Connection Information
- Database Connector Version Info
- User Mappings
- Tables with Encrypted Columns
- Add Table
- Column Encryption for Table
- Column Properties
- Database Configuration Procedures

Database List

Use the Database List section to view the databases (or database servers) that have been added through the Management Console. Once added, you can configure the database's cryptographic settings through the Management Console.



The following table describes the components of the Database List section.

Component	Description
Alias	Displays the alias used to refer to the combination of hostname or IP, port, database login and password, and database name used to connect to this database.
Database Name/SID	Displays the name or SID of the database that stores the columns that you want to encrypt.
Type	Type of database.
Description	Displays an optional description of the database.
Host	Displays the hostname or IP address of the database server. If you enter a hostname (instead of an IP address), it is important to note that the NAE Server is only able to find the database if you have configured a valid DNS Server on the Network Configuration page.
Port	Displays the port on which the database server is listening for connections. On Oracle, the default is 1521.
Database User Login	Displays the login name that is used to connect to the database.
Delete	Click **Delete** to remove a database from this section. Administrators must have the Database Tools ACL to perform this function.
Properties	Click **Properties** to access the connection information, encrypted tables, and table operations for the selected database. Administrators must have the Database Tools ACL to perform this function.
Edit Connection	Click **Edit Connection** to modify the database connection data listed in this section. Administrators must have the Database Tools ACL to perform this function.
User Access	Click **User Access** to access the User Mappings section and manage user mapping information. Administrators must have the Database User Access ACL to perform this function.

Related Overview Material

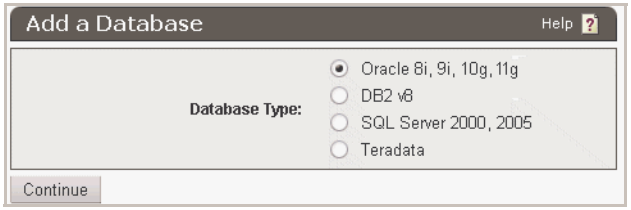
- “Managing User Mappings” on page 51

Related Procedures

- “Adding a Database to the Database List” on page 67
- “Removing a Database From the Database List” on page 68
- “Updating Database Connection Information” on page 69
- “Setting Up User Mappings” on page 69
- “Setting Up Default User Mapping and Error Replacement Values” on page 70

Add a Database

Use the Add a Database section to select a database type and begin the process of adding a database.



The following table describes the components of the Database Connector Version Info section.

Component	Description
Database Type	Select the type of database to which you will connect. Available options are *Oracle 8i, 9i, 10g*, *DB2 v8*, *SQL Server 2000, 2005*, and *Teradata*.
Continue	Click Continue to access the Connection Information page.

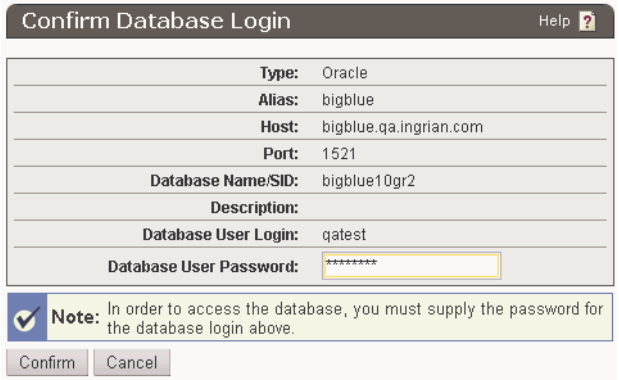
Related Procedures

- “Adding a Database to the Database List” on page 67

Confirm Database Login

Use the Confirm Database Login section to provide the password for the database user login used to access the database.

When you click Properties, you are presented with an intermediate screen requiring you to provide the password for the database login that is being used to login to the database.



The following table describes the components of the Confirm Database Login section.

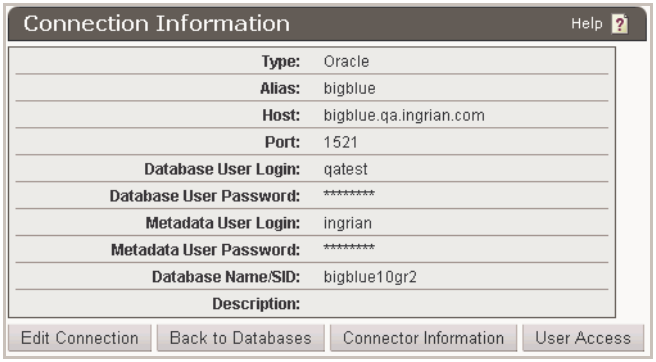
Component	Description
Type	Displays the type of database.
Alias	The alias used to refer to the combination of hostname or IP, port, database login and password, and database name used to connect to this database.
Host	The hostname or IP address of the database server.
Port	The port on which the database server is listening.
Database Name/SID	The database name or SID.
Description	Optional description of the database.
Database User Login	Login name used to connect to the database.
Database User Password	Enter the password to log in to the database.
Confirm	Click **Confirm** to submit the login password and access the connection information, encrypted tables, user mappings, and table operations for the selected database
Cancel	Click **Cancel** to abort the login process.

Related Procedures

- “Adding a Database to the Database List” on page 67

Connection Information

Use this section to enter the connection information for your database.



The following table describes the components of the Connection Information section.

Component	Description
Type	Displays the type of database you are connecting to.
Alias	Enter a name to be used to refer to this connection information. This field uniquely identifies the database connection. The system uses this term to access the hostname or IP, port, database login and password, and database name on this page.
Host	Enter the hostname or IP address of the database server. If you enter a hostname (instead of an IP address), it is important to note that the NAE Server is only able to find the database if you have configured a valid DNS Server on the Network Configuration page.
Port	Enter the port on which the database server is listening for connections. For Oracle, the default port is 1521.
Database User Login	Enter the login name that has permission to modify the tables to be migrated.
Database User Password	Enter the password associated with the login.
Metadata User Login	Enter the database login used to connect to the database. For Oracle, the default login is ingrian. The metadata user modifies Ingrian metadata tables. For Oracle databases, the INGRIAN user must have permissions to create views and triggers, drop views and triggers, and create tables.
Metadata User Password	Enter the password associated with the database login.
Database Name/SID	Enter the name of the database that stores the columns you want to encrypt.
Description	Enter a description for the database. This field is optional.

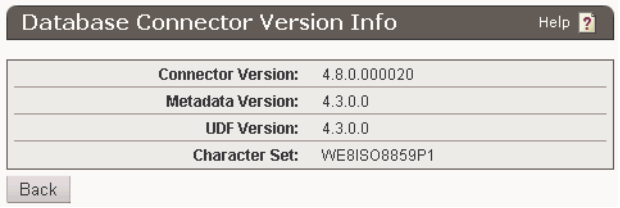
Component	Description
Add Database	Click **Add Database** to add the database to the Database List. **Note:** When adding databases, or modifying the Database Tools pages, database information is being accessed over a network connection. It might take a few seconds for data or schema to be returned to the NAE Server; this is normal.
Back to Databases	Click **Back to Databases** to return to the Database Configuration page.

Related Procedures

- “Adding a Database to the Database List” on page 67

Database Connector Version Info

The Database Connector Version Info section displays the version of the connector, metadata, and UDF. You will experience limited functionality when these values are not in agreement.



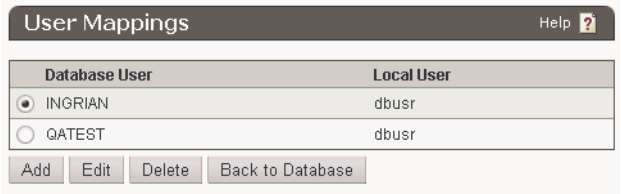
The following table describes the components of the Database Connector Version Info section.

Component	Description
Connector Version	Displays the database connector version.
Metadata Version	Displays the metadata version.
UDF Version	Displays the UDF version.
Character Set	Displays the character set used by the database.
Back	Click **Back** to return to the Database Configuration page.

User Mappings

The User Mappings section shows the association between database users and NAE users.

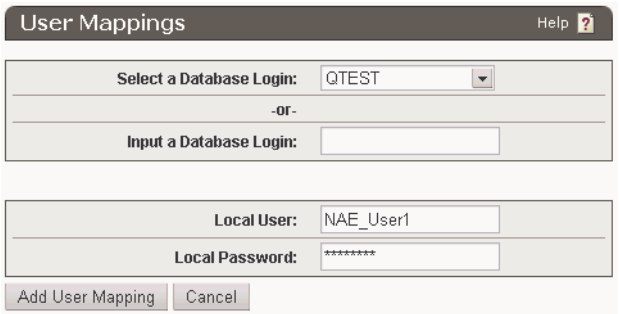
When there are user mappings, the User Mapping looks like this:



The following table describes the components of the User Mappings section.

Component	Description
Database User	This field is displayed when at least one database user has been mapped to an NAE user. It shows the database login used to connect to the database. The Default Mapping value applies to all database users not otherwise listed in this section.
NAE User	When creating a user mapping, this field is used to specify the NAE user account available to the corresponding database login. When at least one user mapping exists, this field displays which NAE user a particular database user is mapped to.
Add	Click **Add** to add a user mapping.
Edit	Click **Edit** to update the selected user mapping. Note: If you change user mappings, Oracle databases require that you establish a new connection to the database for the changes to take effect.
Delete	Click **Delete** to remove the selected user mapping.
Back to Database	Click **Back to Database** to return to the Connection Information section. This button is only available if you accessed the User Mappings section from the Connection Information section.

After clicking the **Add** button, the Management Console displays the following:



Related Overview Material

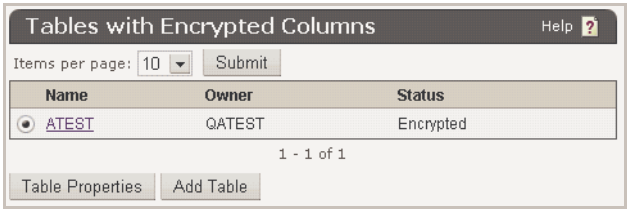
- “Managing User Mappings” on page 51
- “Default User Mapping” on page 52

Related Procedures

- “Setting Up User Mappings” on page 69
- “Setting Up Default User Mapping and Error Replacement Values” on page 70

Tables with Encrypted Columns

Use this section to view all of the tables in the database that have been configured for encryption.



The following table describes the components of the Tables with Encrypted Columns section.

Component	Description
Name	Displays the name of the table.
Owner	Displays the table’s owner/schema.
Status	Displays the table’s encryption status. Possible values are *Encrypted, Deployed, Migration Pending*, *Requires Upgrade*, *Failed Operation*, *Cancelled Operation, Processing,* and *Has Old Data.*
Table Properties	Click **Table Properties** to access the Column Encryption for Table and Table Operations sections.
Add Table	Click **Add Table** to add a new table and configure the migration settings.

Related Procedures

- “Adding a Table” on page 73

Add Table

Use this section to select the table to migrate.

The following table describes the components of the Add Table section.

Component	Description
Owner/ Schema	Displays the owner/schema associated with the login. To encrypt tables owned by other users, click the **Change Owner** button and select a new owner/schema.
Select a Table	Select a table from the list. The system displays only the tables accessible by the chosen owner/schema.
Input a Table Name	Enter a table accessible by the chosen owner/schema. If the database contains more than 50 tables, you may need to enter the table name in the text box.
Add Table	Click **Add Table** to add this table to the Tables with Encrypted Columns section. You can then select the columns to encrypt.
Back	Click **Back** to return to the Database Configuration page without adding a table.

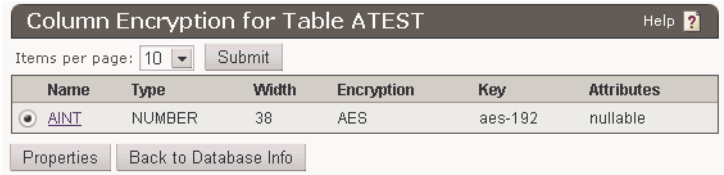
Important! When encrypting tables in Oracle databases, the NAE Server checks to make sure the user/schema has the appropriate permissions to do the migration. If the necessary permissions have not been granted, the **Data Migration** button in the Table Operations section is disabled and a message explains why.

Related Procedures

- “Adding a Table” on page 73

Column Encryption for Table

Use this section to view the column-level encryption settings for each column in a table.



After importing a table, the Column Encryption section lists the columns. Once columns are configured for encryption, they are displayed in orange. After migration, the highlight is removed.

Before a column is configured for encryption, there is no key or encryption algorithm associated with that column. Therefore, there is no value for the Key field and the value for the Encryption field is *None* for the first two columns listed.

The following table describes the components of the Column Encryption for Table section.

Component	Description
Name	Lists names of the column in a table.
Type	Lists the column type. Column types vary by database.
Width	Displays the width of the column.
Encryption	If the column is configured for migration, the column is highlighted in orange, and the type of algorithm that will be used to encrypt and decrypt data in that column is listed.
Key	Lists the name of the key that will be used to encrypt and decrypt data in that column.
Attributes	Lists notable attributes that you should take note of before encrypting a column. Some attributes, such as nullable, will not preclude the encryption of a column; other attributes, such as foreign key, will preclude encryption. Some attributes, such as primary key should be considered very carefully because it is possible that the key is referenced implicitly as a foreign key.
Properties	Click **Properties** to access the Database Column Properties section for the selected column and view its properties. If the column is currently in an encrypted state, you can not modify its properties. If the column is not encrypted, you will be able to modify the properties.
Back to Database Info	Click **Back to Database Info** to return to the Database Configuration page.

Related Overview Material

- “Selecting the Data to Encrypt” on page 44

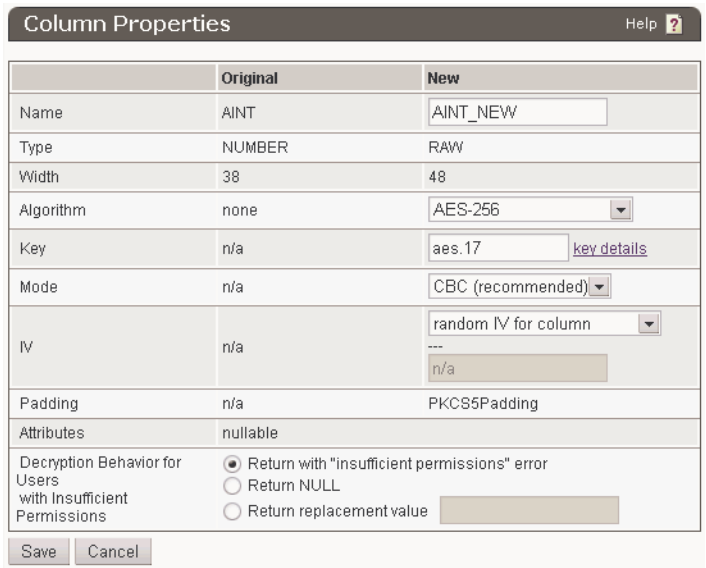
Related Procedures

- “Assigning the Column-Level Encryption Details” on page 74

Column Properties

Use the Column Properties section to view and modify the encryption properties of a column. These properties include the name of the new column to be created during the encryption process, the encryption algorithm, the key, mode, and IV. With the exception of the error replacement values, the table encryption settings cannot be changed after the table is migrated.

Important! Once you save the column properties for a particular column, it is important that you not significantly modify that column or table (e.g. rename the column, drop and recreate the table, etc.) before doing a data migration. You might encounter a broad range of problems if you do.



The following table describes the components of the Column Properties section.

Component	Description
Name	The new default name is <column>_NEW. As part of the data migration process, the system creates a new column to hold the encrypted data. This name is also used to create any IV columns. You can accept the default column name or enter a different name. Once you have saved the column properties here in the Management Console, do not modify the column name in the database. Discrepancies between the column name in the database and the encryption instructions cause an error. If such an error occurs during the migration, restore the original column name shown in the error message.
Type	Displays the datatype of the selected column.
Width	Displays the width of the selected column, before and after encryption. **Note:** The algorithm you select impacts the column width for the encrypted data.

Component	Description
Algorithm	Select an encryption algorithm. The algorithm can be any of the following: • AES • DESede • DES (We discourage using DES.) • SEED (If the SEED algorithm has been feature activated on the DataSecure appliance.) The NAE Server supports PKCS#5Padding and NoPadding options. However, because the NoPadding option is not practical for encrypting data in a database, PKCS#5Padding is used for all block ciphers (DES, DESede, and AES).
Key	Select an encryption and decryption key. Only keys that apply the selected algorithm and to which you have access are available. The Key Details window is described in “Key Information” on page 48. The key you select must be available to the database users that perform the migration.
Mode	Select a mode for AES, DESede, SEED, and DES algorithms. We recommend that you use block ciphers (DES, DESede, SEED, and AES), in CBC mode unless you have a compelling reason to use ECB mode. CBC is considered to be a more secure mode for a variety of reasons: ECB’s largest disadvantage is that for a given key, two identical plaintexts will correspond to an identical ciphertext; whereas CBC uses the ciphertext of the previous block of plaintext as the initialization vector for the encryption of the next block of plaintext. This succeeds in guaranteeing that two identical plaintext blocks will not result in the same ciphertext. CBC detects if blocks arrive out of order, which prevents a block switching attack.
IV	Select the initialization vector (IV) method. An IV is a sequence of random bytes appended to the front of the plaintext before encryption by a block cipher. The IV method is only available when using AES, DES, or DESede algorithms. The IV you specify for operations that use DES or DESede keys must be eight bytes; for AES keys, the IV must be sixteen bytes. If you specify an IV here, the IV must be specified in hexadecimal (base 16 encoded) characters. An eight byte IV requires sixteen characters; a sixteen byte IV requires thirty-two characters. The IV is required for CBC mode and not allowed for ECB mode. Using an IV eliminates the possibility that the initial ciphertext block will be the same for any two encryption operations that use the same key. You can apply IVs at the field-level or column-level. **Note:** After a column has been encrypted, this field displays the IV that was used during the encryption (where appropriate). This is especially useful when using the ETL Utility to perform bulk encrypt or decrypt operations. For more information about the benefits of either field–level and column–level IVs, see “Applying Initialization Vectors”.
Padding	Displays the padding mode to be used when encrypting the column. The default value is PKCS5Padding.
Attributes	Displays the attributes of the column.

Component	Description
Decryption Behavior for Users with Insufficient Permissions	Select the value returned by the system when a user with insufficient access attempts to query the database. Valid values are: • Return “insufficient permissions” error – unauthorized requests return an error. • Return NULL – returns a NULL value in for when unauthorized requests are made in this column. The query will execute without generating an error, and the value returned for the decrypted column is NULL. • Return replacement value – Selecting this option allows you to specify the value that is returned when a user makes an unauthorized attempt to decrypt the data in this column. The value that you specify must be a valid value for the data type and length of the column being configured. A query by an unauthorized user will return successfully, and the specified value will be returned in the place of the actual decrypted value. Replacement values are not returned if a query yields a NULL value. When a query results in a NULL value, no cryptographic process is required, so the database tools do not interact with the NAE Server and the replacement values feature is not activated. **Note:** A user with select, update, and delete privileges can delete data from a migrated table based on the error replacement values. For more information, see “Replacement Values” on page 49. The error replacement values feature requires that the Database Tools settings are identical on all cluster members. See “Cluster Settings” in the DataSecure Appliance User Guide for more information.
Save	Click **Save** to save the Column Properties values. The Column Encryption for Table section will then display this column in orange. The Table Operations section will indicate that the **Job History** and **Data Migration** buttons are available.
Cancel	Click **Cancel** to return to the Column Encryption for Table section without saving the values in this section.

Related Overview Material

- “Selecting the Data to Encrypt” on page 44
- “Replacement Values” on page 49

Related Procedures

- “Assigning the Column-Level Encryption Details” on page 74
- “Setting Up Default User Mapping and Error Replacement Values” on page 70

Database Configuration Procedures

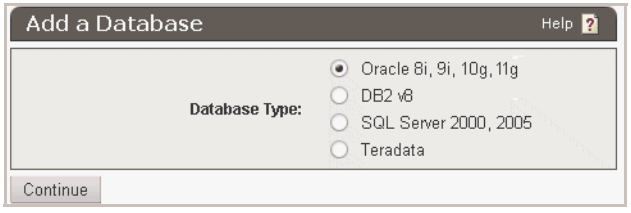
This section describes the procedures you will follow when configuring a database connection and preparing to migrate data. It explains the following processes:

- Adding a Database to the Database List
- Removing a Database From the Database List
- Updating Database Connection Information
- Setting Up User Mappings
- Setting Up Default User Mapping and Error Replacement Values
- Adding a Table
- Preparing for Data Migration
- Selecting the Data to Migrate
- Assigning the Column-Level Encryption Details

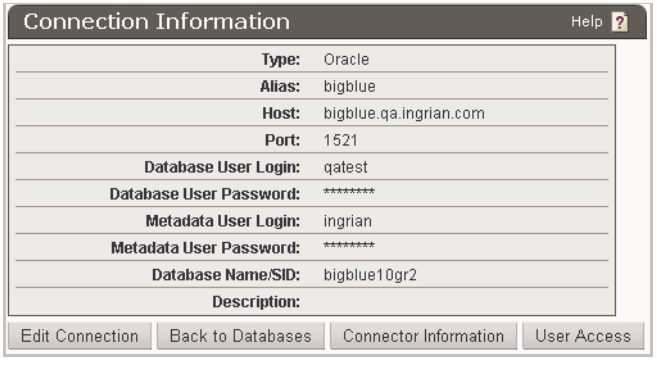
Adding a Database to the Database List

To add a database to the Database List:

1. Log in to the Management Console as an administrator with Database Tools access control.
2. Navigate to the Add a Database section of the Database Configuration page.



3. Select *Oracle 8i, 9i, 10g, 11g* and click the **Continue** button.



4 Enter the following information on the Connection Information page:
   - **Alias:** Enter a name to be used to refer to this connection information. This field uniquely defines the database connection. The system will use this term to access the hostname or IP, port, database login and password, metadata login and password, and database name on this page.
   - **Host:** Enter the hostname or IP address of the database server. If you enter a hostname, the NAE Server is only able to find the database if you have configured a valid DNS Server on the Network Configuration page.
   - **Port:** Enter the port on which the database server is listening for connections. For Oracle, the default port is 1521.
   - **Database User Login:** Enter the login name that has permission to modify the tables to be migrated.
   - **Database User Password:** Enter the password associated with the login.
   - **Metadata User Password:** Enter the password associated with the *ingrian* database login. *ingrian* is the database login used to connect to the Ingrian database.
   - **Database Name/SID:** Enter the name of the database that stores the columns you want to encrypt.
   - **Description:** Enter a description for the database. This field is optional.

5 Click the **Add Database** button.

Note: When adding databases, or modifying the Database Tools pages, database information is being accessed over your network connection. It might take a few seconds for data or schema to be returned to the NAE Server; this is normal.

Related Sections

- “Add a Database” on page 56
- “Connection Information” on page 58

Removing a Database From the Database List

To remove a database from the Database List:

1 Log in to the Management Console as an administrator with Database Tools access control.

2 Navigate to the Database List section of the Database Configuration page.

3 Select the database from the list.

4 Select **Delete**.

Related Sections

- “Database List” on page 55

Updating Database Connection Information

To update the connection information for a connected database:

1. Log in to the Management Console as an administrator with Database Tools access control.
2. Navigate to the Database List section of the Database Configuration page.
3. Select the database and click **Edit Connection** to access the Connection Information section.
4. Update the fields as needed.
5. Click **Save**.

Related Sections

- “Connection Information” on page 58

Setting Up User Mappings

To create a user mapping:

1. Log in to the Management Console as an administrator with Database User Access access control.

   Note: Administrators do *not* need access to the Database Tools to create a user mapping, but they must have the Database User Access ACL.

2. Navigate to the User Mappings section by clicking the **User Access** button on the Connection Information or the Database List section.
3. Click the **Add** button to access the User Mappings section
4. Enter the following information in the User Mappings section:
   - Database Login: Use either the **Select a Database Login** or the **Input a Database Login** field to enter the database login.
   - NAE User: Enter the NAE user to which you want to map the database user.
   - NAE Password: Enter the password for the NAE user.
5. Click the **Add User Mapping** button to add the mapping.

   The new information appears in the User Mappings section and in the `ING_AUTHORIZED_USER` table in the database. NAE usernames and passwords are encrypted before they are inserted into the `ING_AUTHORIZED_USER` table.

Related Overview Material

- “Managing User Mappings” on page 51
- “Default User Mapping” on page 52

Related Sections

- “User Mappings” on page 60

Setting Up Default User Mapping and Error Replacement Values

To create the default user mapping:

1. Log in to the Management Console as an administrator with Database User Access access control.

   Note: Administrators do *not* need access to the Database Tools to create a user mapping, but they must have the Database User Access ACL.

2. Navigate to the Local NAE Users section on the NAE User & Group Configuration page on the Key Management tab.
3. Click the **Add** button to create a new NAE user.
4. Enter the following information in the Local NAE Users section:
   - Username: Use an easily identifiable name, such as “defaultmapping”.
   - Password: Enter the password for the local NAE user. The requirements for the local NAE user password depend on your Password Management Settings.

   Note: *Do not* select **NAE User Administration Permission**, **Change Password Permission**, or **Distinct Passwords per EdgeSecure**.

5. Navigate to the User Mappings section by clicking the User Access button on the Connection Information or the Database List section.

   Note: Administrators do not need access to the Database Tools to create a user mapping, but they must have the Database User Access ACL.

6. Click the **Add** button to access the User Mappings section.
7. Enter the following information in the User Mappings section:

- **Select a Database Login**: Select [Default Mapping].
- **NAE User**: Enter the NAE user that you created above.
- **NAE Password**: Enter the password for the NAE user.

8 Click the **Add User Mapping** button to add the mapping.

The new information appears in the User Mappings section and in the `ING_AUTHORIZED_USER` table in the database. The default database user is listed as * (an asterisk). For this reason, you should avoid using an asterisk to represent a specific database user. NAE usernames and passwords are encrypted before they are inserted into the `ING_AUTHORIZED_USER` table.

Note: You only need to create the default user mapping once for a database. You do not need to follow this procedure for each table.

9 Navigate to the Column Properties section for a column to which you would like to add a replacement value.

10 Select *Return replacement value* in the **Decryption Behavior for Users with Insufficient Permissions** field.

Note: You can alter the **Decryption Behavior for Users with Insufficient Permissions** field even after migrating the table.

11 Enter a value that meets the column's datatype specification. For example, if the column is defined as `VARCHAR(20)`, the replacement value must be a valid `VARCHAR(20)` value.

To verify that the default user mapping is set up correctly:

1 Log in to the database as a user that does not have decryption permission.

2 Run the following query:

```
select * from ingrian.ing_auth_user;
```

If no rows are returned, the default user is not configured correctly.

Related Overview Material

- "Default User Mapping" on page 52
- "Replacement Values" on page 49

Related Sections

- "User Mappings" on page 60

Preparing for Data Migration

Prior to migrating your data, you should:

1. Back up the database. We *strongly* recommend that you back up your database before migrating data.
2. Configure the NAE Server.

   For more on configuring the NAE Server, consult the DataSecure Appliance User Guide.
3. Generate keys on the NAE Server.

   Make sure that you assign encryption and decryption privileges appropriately. For information on how to generate keys, consult the DataSecure Appliance User Guide.
4. Grant permissions to the user migrating the data.

   The NAE Server can prevent database users with insufficient permissions from initiating a data migration.

   To migrate data, database users must have the following permissions within the database:
   - Connect.
   - Create any table.
   - Create any index.
   - Drop any table.
   - Select any table.
   - Alter table – required for the table being encrypted.
   - Update table – required for the table being encrypted.

   Note: Additional permissions are required to create views and triggers. For a list of those permissions, see “Creating Views and Triggers” on page 92.

Selecting the Data to Migrate

To select the data to migrate you must complete the following steps:

- Add your database to the database list. This procedure establishes a connection between your database and the NAE Server.
- Set up user mappings. This procedure associates database users with NAE users.
- Select the table to migrate.
- Assign the column-level encryption details.

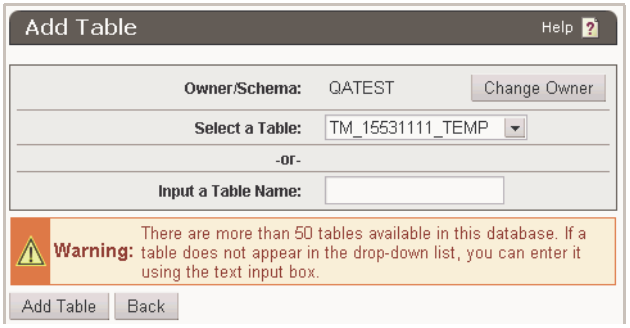
Once these setup steps are completed, you can run a migration process.

Adding a Table

To select a table to migrate:

1. Log in to the Management Console as an administrator with Database Tools access control.
2. Navigate to the Tables with Encrypted Columns section of the Database Tools page.

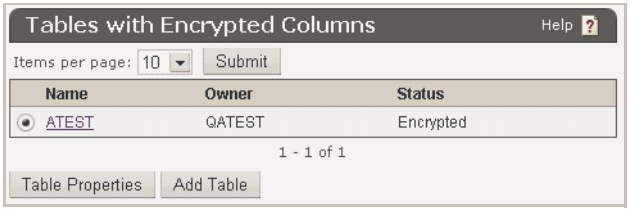
3. Click the **Add Table** button to access the Add Table section.



4. The page will display the owner/schema associated with the login. Click **Change Owner** to encrypt a table owned by another user.
5. Use either the **Select a Table** or the **Input a Table Name** field to enter a table. The system only accepts a table accessible by the chosen owner/schema. The **Select a Table** field can display only 50 tables; for a large database, you may need to use the **Input a Table Name** field.

   Important! When encrypting tables in Oracle databases, the NAE Server checks to make sure the user/schema has the appropriate permissions to do the migration. If the necessary permissions have not been granted, the **Data Migration** button in the Table Operations section is disabled and a message explains why.

6. Click the **Add Table** button. The table is added to the Tables with Encrypted Columns section and you can select a column to encrypt.



Related Sections

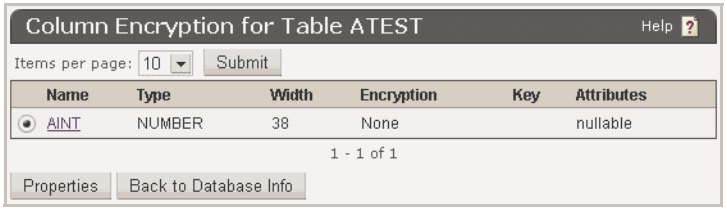
- “Add Table” on page 62
- “Tables with Encrypted Columns” on page 61

Assigning the Column-Level Encryption Details

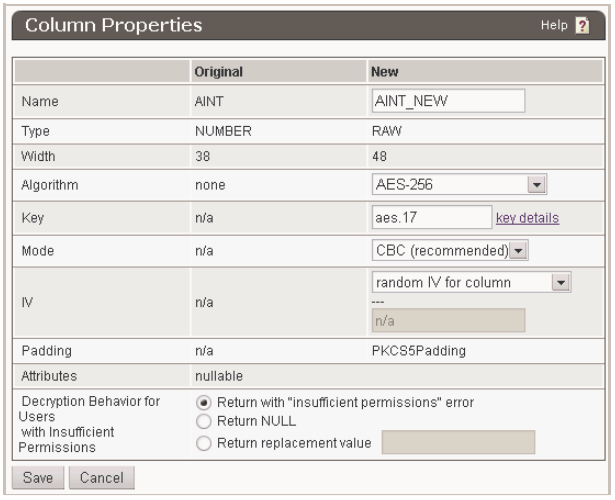
The column-level encryption details determine how and if a column is encrypted and decrypted.

To assign the column-level encryption details:

1. Log in to the Management Console as an administrator with Database Tools access control.
2. Navigate to the Tables with Encrypted Columns section of the Database Tools page.
3. Click a table name or select the table and click the **Table Properties** button to access the Column Encryption for Table section.



4. Click a column name or select the column and click the **Properties** button to access the Column Properties section.



5 Modify the fields below as shown:

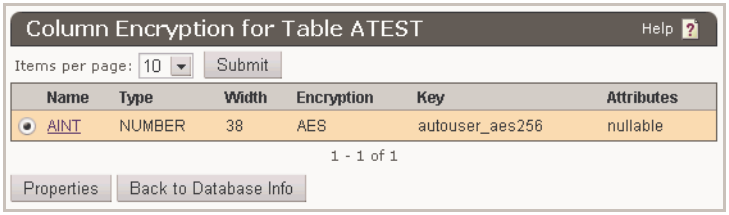
Component	Description
Name	The new default name is <column>_NEW. As part of the data migration process, the system creates a new column to hold the encrypted data. This name is also used to create any IV columns. You can accept the default column name or enter a different name. Once you have saved the column properties here in the Management Console, do not modify the column name in the database. Discrepancies between the column name in the database and the encryption instructions cause an error. If such an error occurs during the migration, restore the original column name shown in the error message.
Type	Displays the datatype of the selected column.
Width	Displays the width of the selected column, before and after encryption. **Note:** The algorithm you select impacts the column width for the encrypted data.
Algorithm	Select an encryption algorithm. The algorithm can be any of the following: • AES • DESede • DES (We discourage using DES.) • SEED (If the SEED algorithm has been feature activated on the DataSecure appliance.) The NAE Server supports PKCS#5Padding and NoPadding options. However, because the NoPadding option is not practical for encrypting data in a database, PKCS#5Padding is used for all block ciphers (DES, DESede, and AES).
Key	Select an encryption and decryption key. Only keys that apply the selected algorithm and to which you have access are available. The Key Details window is described in “Key Information” on page 48. The key you select must be available to the database users that perform the migration.
Mode	Select a mode for AES, DESede, SEED, and DES algorithms. We recommend that you use block ciphers (DES, DESede, SEED, and AES), in CBC mode unless you have a compelling reason to use ECB mode. CBC is considered to be a more secure mode for a variety of reasons: ECB’s largest disadvantage is that for a given key, two identical plaintexts will correspond to an identical ciphertext; whereas CBC uses the ciphertext of the previous block of plaintext as the initialization vector for the encryption of the next block of plaintext. This succeeds in guaranteeing that two identical plaintext blocks will not result in the same ciphertext. CBC detects if blocks arrive out of order, which prevents a block switching attack.

Component	Description
IV	Select the initialization vector (IV) method. An IV is a sequence of random bytes appended to the front of the plaintext before encryption by a block cipher. The IV method is only available when using AES, DES, or DESede algorithms. The IV you specify for operations that use DES or DESede keys must be eight bytes; for AES keys, the IV must be sixteen bytes. If you specify an IV here, the IV must be specified in hexadecimal (base 16 encoded) characters. An eight byte IV requires sixteen characters; a sixteen byte IV requires thirty-two characters. The IV is required for CBC mode and not allowed for ECB mode. Using an IV eliminates the possibility that the initial ciphertext block will be the same for any two encryption operations that use the same key. You can apply IVs at the field-level or column-level. For more information about the benefits of either field–level and column–level IVs, see "Applying Initialization Vectors".
Padding	Displays the padding mode to be used when encrypting the column. The default value is PKCS5Padding.
Attributes	Displays the attributes of the column.
Decryption Behavior for Users with Insufficient Permissions	Select the value returned by the system when a user with insufficient access attempts to query the database. Valid values are: • Return "insufficient permissions" error – unauthorized requests return an error. • Return NULL – returns a NULL value in for when unauthorized requests are made in this column. The query will execute without generating an error, and the value returned for the decrypted column is NULL. • Return replacement value – Selecting this option allows you to specify the value that is returned when a user makes an unauthorized attempt to decrypt the data in this column. The value that you specify must be a valid value for the data type and length of the column being configured. A query by an unauthorized user will return successfully, and the specified value will be returned in the place of the actual decrypted value. Replacement values are not returned if a query yields a NULL value. When a query results in a NULL value, no cryptographic process is required, so the database tools do not interact with the NAE Server and the replacement values feature is not activated. **Note:** A user with select, update, and delete privileges can delete data from a migrated table based on the error replacement values. For more information, see "Replacement Values" on page 49. The error replacement values feature requires that the Database Tools settings are identical on all cluster members. See "Cluster Settings" in the DataSecure Appliance User Guide for more information.

6 Click the **Save** button to save the Column Properties values.

The Column Encryption for Table section displays this column in orange. The Table Operations section also indicates that the **Job History** and **Data Migration** buttons are available.

Click the **Cancel** button to return to the Column Encryption for Table section without saving the values in this section.



Data Migration Sections

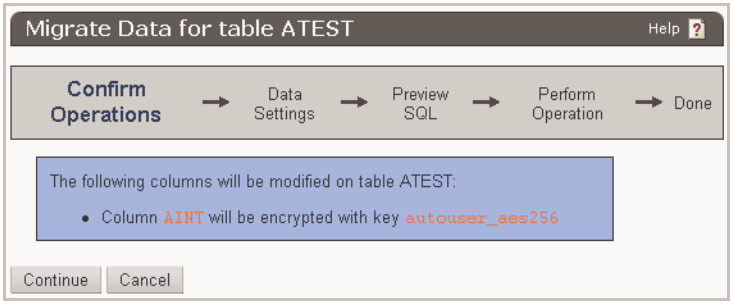
Data migration is the process of migrating a column or columns in a table from a cleartext state to an encrypted state.

You launch the data migration process from the Table Operations section. The migration process involves the following sections:

- Confirm Operations
- Data Settings
- Preview SQL
- Perform Operation
- Done

Confirm Operations

The Migrate Data for table: Confirm Operations section displays the columns and keys involved in the data migration.



The following table describes the components of the Migrate Data for table: Confirm Operations section. This section lists the columns to be encrypted and the corresponding encryption keys.

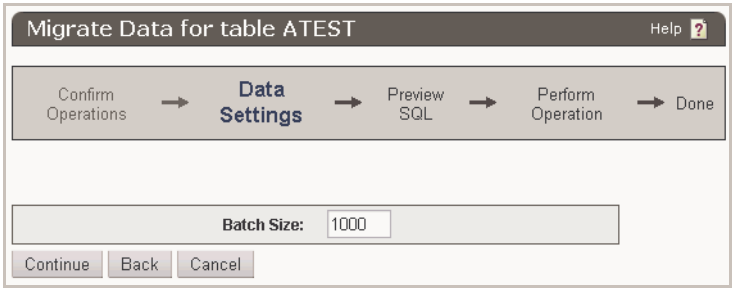
Component	Description
Continue	Click **Continue** to proceed to the Data Settings section.
Cancel	Click **Cancel** to stop the migration process.

Related Procedures

- “Migrating Data” on page 81

Data Settings

Use the Migrate Data for table: Data Settings section to enter the batch size.



The following table describes the components of the Migrate Data for table: Data Settings section. This section lists the columns to be encrypted and the corresponding encryption keys.

Component	Description
Batch Size	Specify the batch size of the migration job.
Continue	Click **Continue** to preview the SQL.
Back	Click **Back** to return to the Confirm Operations section.
Cancel	Click **Cancel** to stop the migration process.

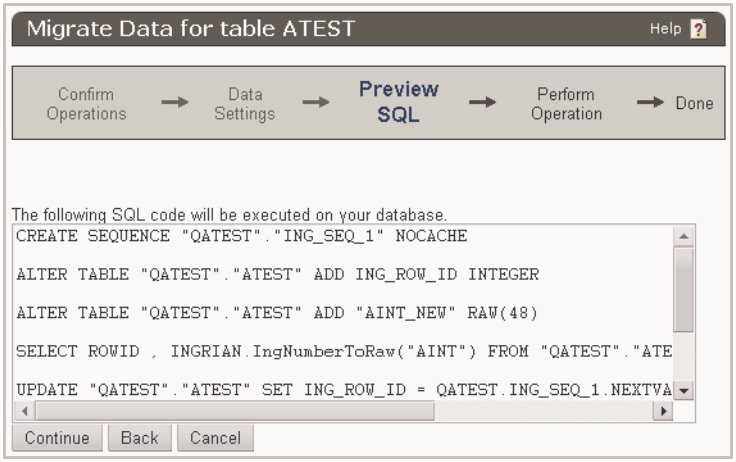
Related Procedures

- “Migrating Data” on page 81

Preview SQL

The Migrate Data for table: Preview SQL section displays the SQL code that will be executed on your database.

Note: The SQL code displayed is a model of a SQL statement that will be run for each row. The model substitutes "?" for the actual data values.



The following table describes the components of the Migrate Data for table: Preview SQL section.

Component	Description
SQL code	Displays the SQL code that will be executed on your database. You cannot edit this code.
Continue	Click **Continue** to perform the operation.
Back	Click **Back** to return to the Data Settings section.
Cancel	Click **Cancel** to stop the migration process.

Related Procedures

- "Migrating Data" on page 81

Perform Operation

The Migrate Data for table: Perform Operation section displays the current status of the process.

The following table describes the components of the Migrate Data for table: Perform Operation section.

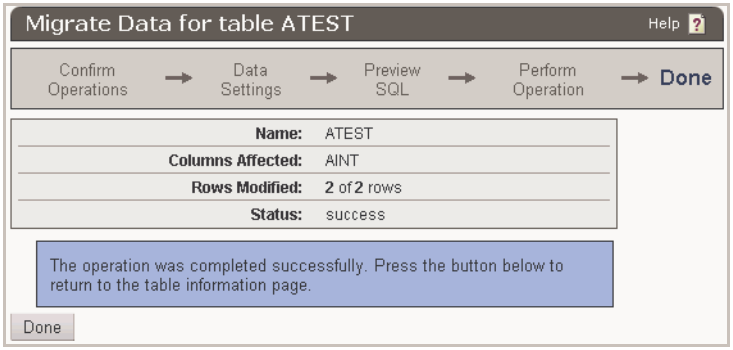
Component	Description
Name	Displays the name of the table.
Columns Affected	Displays the names of the columns being encrypted.
Rows Modified	Displays the number of rows that have been modified. This number will grow as the page refreshes, until the process is complete.
Status	Displays the process status.
Refresh	Click **Refresh** to refresh the page and display the latest number of rows modified and the current status.
Cancel	Click **Cancel** to stop the process. The process will stop after completing the current batch.

Related Procedures

- “Migrating Data” on page 81

Done

The Migrate Data for table: Done section appears when the process is complete.



The following table describes the components of the Migrate Data for table: Done section.

Component	Description
Name	Displays the name of the table.
Columns Affected	Displays the names of the columns being encrypted.
Rows Modified	Displays the number of rows that have been modified.

Component	Description *(continued)*
Status	Displays the process status.
Done	Click **Done** to return to the Database Configuration page.

Related Procedures

- “Migrating Data” on page 81

Data Migration Procedure

This section describes the procedure you will use when migrating data. You must have already connected a database to the DataSecure appliance and prepared a table for migration. This section explains the following process:

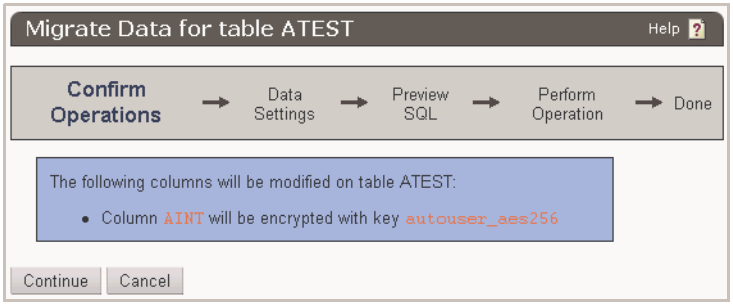
- Migrating Data

Migrating Data

You can migrate your data only after connecting the database and the NAE Server, establishing user mappings that link database users and NAE users, selecting a table, and associating columns with cryptographic parameters.

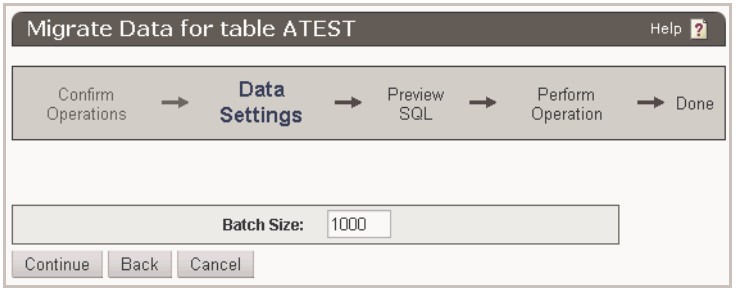
To migrate your data:

1. Navigate to the Database List section of the Database Tools page.
2. Click the alias of the desired database, or select the alias and click the **Properties** button.
3. Navigate to the Tables with Encrypted Columns section. Click the name of the table you want to migrate, or select the name and click the **Table Properties** button.
4. Navigate to the Table Operations section and click the **Data Migration** button.
5. Review the information on the Confirm Operations page. Click the **Continue** button if this information is correct.

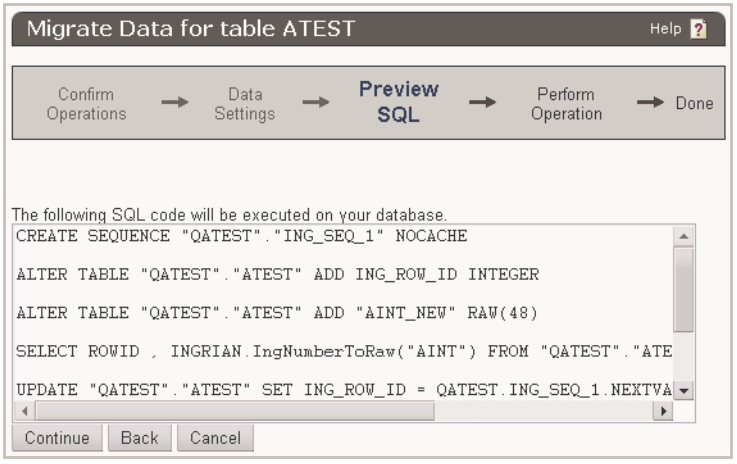


6 Review the information on the Data Settings page. Change the batch size if necessary.

Click the **Continue** button if this information is correct.



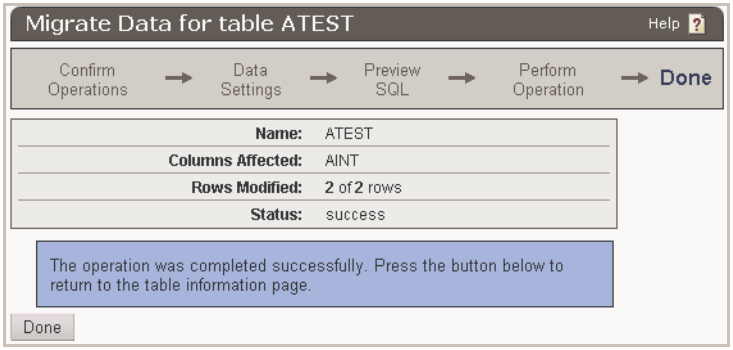
7 The operations about to be performed is displayed for your reference. Click **Continue** once you have reviewed the information below. This starts the data migration process.



Note: After you commit to migrate the data, the user interface might seem unresponsive for a brief period of time. This is normal. Please be patient while the NAE Server interacts with the database server. Do not click the **Migrate Data** button again.

8 View the progress of the data migration on the Migrate Data page. You can click the **Cancel** button to stop the migration.

You can navigate away from and return to the Migrate Data page. Once the migration is complete, the Management Console displays the Done page.

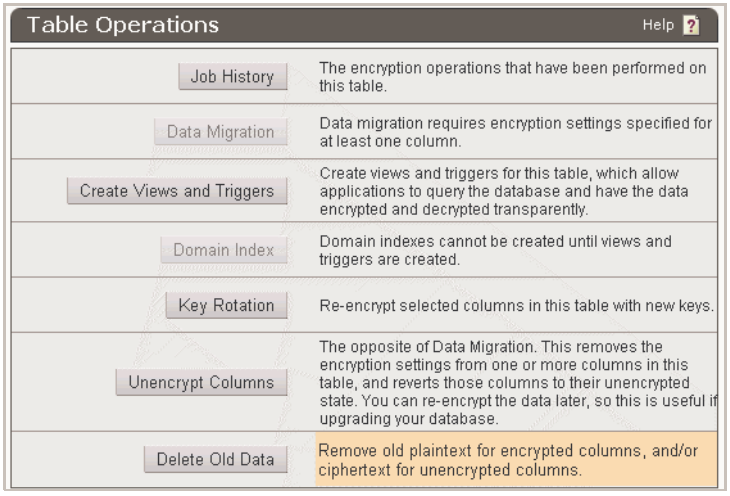


9 The data migration process ends normally, is cancelled, or ends in error.

a If the process ends successfully, click the **Done** button. The data migration is complete.

The **Data Migration** button is unavailable in the Table Section. The following buttons are available: **Job History**, **Create Views and Triggers**, **Key Rotation**, **Unencrypt Columns**, and **Delete Old Data**. The **Domain Index** button is unavailable until you create views and triggers. For information about these operations, see “Managing Encrypted Data” on page 85.

Important! We *strongly* recommend that you use the **Delete Old Data** button to remove the plaintext stored on your table. For instructions on deleting old data, see “Deleting Old Data” on page 89.



b If the data migration process ends in error:

- Click the **Return to Table** button to return to the Table Operations section.
- Click the **Display Failed Operation** button to view the Job Information of the failed process.
- View the details on the Job Information section. You can choose one of the following options:

  Click the **Restore Data** button to drop the columns created during the migration. This returns the table to its pre-migration state.

  Click the **Resume Operation** button to continue the process from where it ended. You may want to select this option after you have remedied the source of the error.

c If you cancelled the data migration process, navigate to the Job Information page by clicking the **Return to Table** button.

- View the details on the Job Information section. You can choose one of the following options:

  Click the **Resume Operation** button to continue the process from where it was cancelled.

  Click the **Restore Data** button to drop the columns created during the migration. This returns the table to its pre-migration state.

Related Sections

- “Table Operations” on page 85
- “Confirm Operations” on page 77
- “Data Settings” on page 78
- “Preview SQL” on page 79
- “Perform Operation” on page 79
- “Done” on page 80

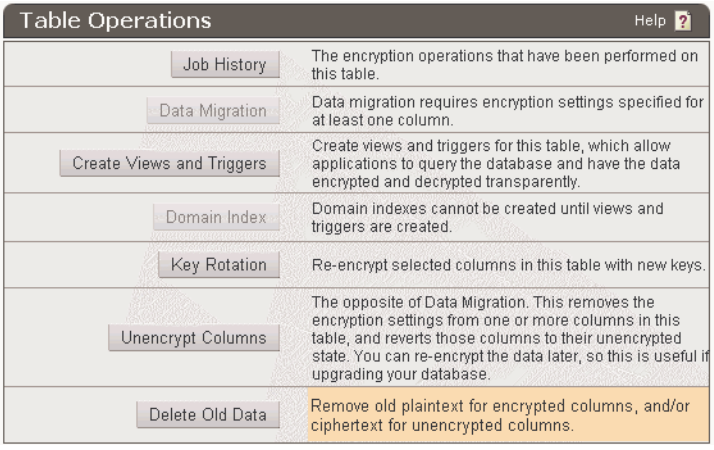
CHAPTER 6

Managing Encrypted Data

This chapter contains step–by–step instructions on how to perform operations on your tables once they have been migrated. Operations include unencrypting tables, creating and deleting views and triggers, rotating keys, and modifying tables. This chapter contains the following information:

Table Operations

From the Table Operations section of the Management Console, you can initiate cryptographic operations such as data migration, key rotation, and unencryption.



The following table describes the components of the Table Operations section.

Component	Description
Job History	Click **Job History** to view the cryptographic operations (data migration, key rotation) that have been performed on this table.
Data Migration	Click **Data Migration** to begin the process of data migration, which modified the database and encrypts the specified columns.
Delete Views and Triggers	Click **Delete Views and Triggers** to create or delete views and triggers associated with a table. Triggers and views are used to facilitate seamless integration with existing applications.
Domain Index	Click **Domain Index** to create a domain index on your encrypted columns. This button is only available for tables within an Oracle database for which views and triggers have been created.
Key Rotation	Click **Key Rotation** to re-encrypt selected columns with new keys.
Unencrypt Columns	Click **Unencrypt Columns** to remove the encryption settings from one or more columns and return those columns to their unencrypted state. You can re-encrypt the data later, so this is useful when upgrading your database.
Delete Old Data	Click **Delete Old Data** to remove the plaintext values on which the encrypted data is based. The plaintext is replaced with null values.
Display Failed Operation	Click **Display Failed Operation** to access the Job Information page to view the details of the operation, resume the operation or restore the table to its previous state. This button only appears if the previous job failed.
Upgrade Table	Click **Upgrade Table** to upgrade the table to a newer version. When this button appears, you will not be able to use any of the other functionality normally available on this screen until the table is upgraded. This button only appears if the table version is older than the DB Connector.

Depending on the state of the table, the operations available from the Table Operations section of the Management Console vary. For example, when no columns are configured for encryption, only the **Job History** button is available. Once a column has been configured for encryption, the **Job History** button stays enabled, and the **Data Migration** button is now enabled. Once a table has been migrated, the **Data Migration** button is disabled, and the remaining buttons are enabled. The **Domain Index** button is enabled only for tables within an Oracle database.

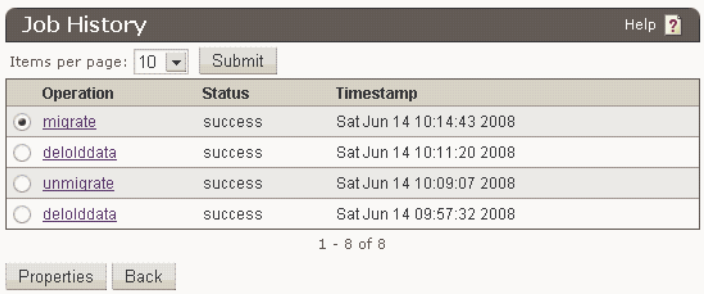
Job History

The Job History section allows you to see what operations have been performed against a table, whether the operation was successfully completed, and when the operation was initiated.

One of the more important functions of the Job History section is that it provides a way to handle operations that do not complete (either because the operation failed or was cancelled). From the

Job History page, you can restore the table to the state it was in before an operation failed or was cancelled, or you can continue the operation that did not complete.

Important! You can only restore or resume the most recent operation.



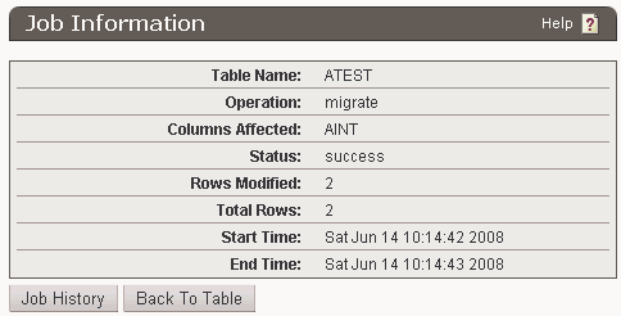
The following table describes the components of the Job History section.

Component	Description
Operation	Displays the type of operation that occurred. Possible values are: • migrate: Data migration. • rotatekey: Key rotation. • unmigrate: Unencrypt columns. • upgrade: Upgrade tables. Click this link to access the Job Information section and view more detailed information about the job.
Status	Displays the status of the job. Possible values are: • success: The job run successfully. • restored: The data was restored to its unencrypted state. • cancelled: The job was cancelled by a user. • failure: The job did not complete successfully. • processing: The job is currently processing.
Timestamp	Displays the timestamp for the process.
Properties	Click **Properties** to access the Database Job Information page and view the details of this operation.
Back	Click **Back** to return to the Database Table Properties page.

Job Information

When an operation is completed successfully, the Job Information page provides some basic information about the operation, including the table name, the column name(s), the operation (migrate, key rotation, etc.), the number of rows affected, and the start and end times.

An example of the Job Information page for a successful migration operation is shown here.



The following table describes the components of the Job Information section.

Component	Description
Table Name	Displays the name of the table.
Operation	Displays the type of operation that was attempted. Possible values are: • migrate: Data migration. • rotatekey: Key rotation. • unmigrate: Unencrypt columns. • upgrade: Upgrade tables.
Columns Affected	Displays the name of the column or columns involved in the job.
Status	Displays the status of the job. Possible values are: • success: The job run successfully. • restored: The data was restored to its unencrypted state. • cancelled: The job was cancelled by a user. • failure: The job did not complete successfully. • processing: The job is currently processing.
Rows Migrated / Rows Rotated	Displays the number of rows that were successfully operated upon before the process ended.
Total Rows	Displays the total number of rows that were processed.
Error Message	Displays an error message if the job failed. This message can indicate where to find additional information or where to begin troubleshooting.
Start Time	Displays the time when the operation began.
End Time	Displays the time when the operation ended.
Resume Operation	Click **Resume Operation** to resume the operation. This button is only available in the event of a job failure or cancellation.
Restore Data	Click **Restore Data** to restore the table to the state it was in before the operation was initiated. This button is only available in the event of a job failure or cancellation.

Component	Description *(continued)*
Job History	Click **Job History** to display the job history.
Back to Table	Click **Return to Table** to return to the Database Table Properties page.

Viewing Job History

To view the history of jobs performed on your table:

1. Navigate to the Database List section of the Database Tools page.
2. Select an alias.
3. Navigate to the Tables with Encrypted Columns section and select a table.
4. Select the **Job History** button on the Table Operations section to view the operations that have been performed on the table.
5. Click the operation link, or select an operation and click the **Properties** button to access the Job Information page and see the details of the job.

   You can use the Job Information page to view the details of the job, resume an interrupted operation, or restore a table to its previous state.

Deleting Old Data

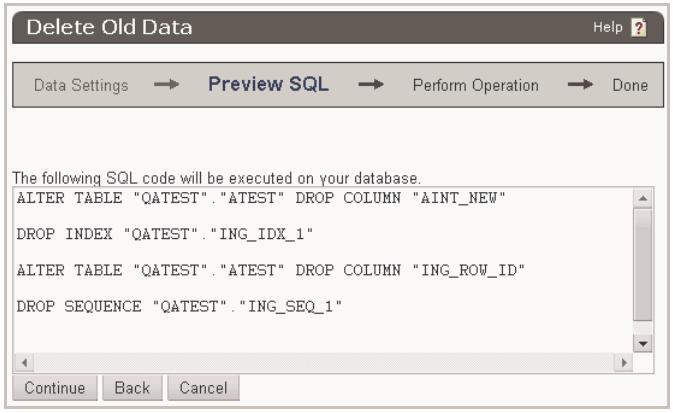
Once you successfully migrate data, we *strongly* suggest that you remove the old plaintext stored on your table.

The delete old data process removes the plaintext of the columns you have encrypted. The columns are set to `NULL`. If the column is not nullable, the values in the column are assigned a different value depending on the database and data type. The following table shows what values are assigned to non–nullable columns during data migration.

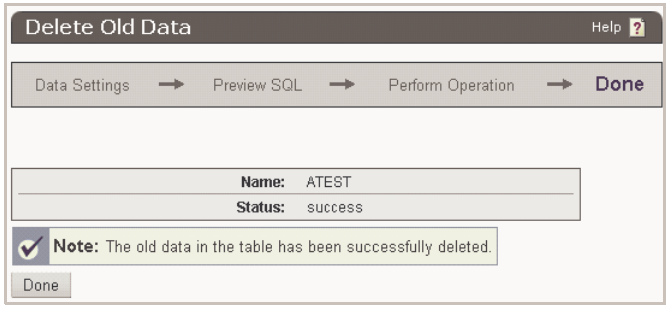
Data Type	Value
CHAR	‘ ’ (single space)
VARCHAR	‘ ’ (single space)
VARCHAR2	‘ ’ (single space)
DATE	01-JAN-1900
All others	0

To delete old plaintext data:

1. Navigate to the Database List section of the Database Tools page.
2. Select an alias.
3. Navigate to the Tables with Encrypted Columns section.
4. Select the **Delete Old Data** button on the Table Operations section to set the batch size for the deletion process.
5. Click **Continue** to preview the SQL code to be executed on your database. You cannot edit this code.



6. Click the **Continue** button to execute the SQL code and delete the plaintext data.



Unencrypting Tables

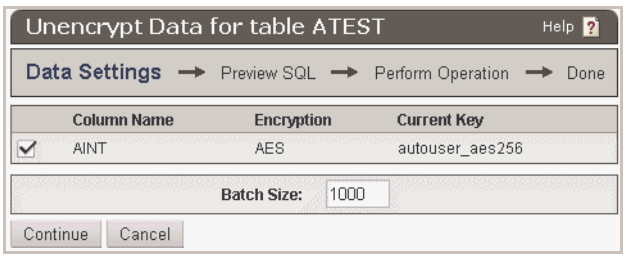
The following steps describe what happens when you remove encryption from a table. For the sake of simplicity, it is assumed that you are unencrypting one column in the table.

1. The column containing the ciphertext is copied to a temporary table along with the identity column and the IV column if there is one.
2. The ciphertext is sent to the NAE Server along with the IV for decryption.
3. The NAE Server returns the plaintext data to the database, and the plaintext is inserted into the original column. Remember that when the data was originally migrated, another column was created and the original column was set to `NULL`.
4. After the plaintext data is inserted into the column, any new columns that were created during the migration are dropped. These might include the column that held the ciphertext, the IV column, and the identity column.

   Tip: Although unencryption restores the plaintext data and drop the columns created during the data migration, it does not restore any constraints lost during encryption. If you want to return a database to its true pre-migration state, restore a backup.

To unencrypt a column:

1. Navigate to the Database Tools page.
2. Click on the database that contains the table with the encrypted column.
3. At the Database Properties page, click on the table that contains the data you want to unencrypt.
4. At the Database Table Properties page, scroll down to the section entitled Table Operations, and click **Unencrypt Columns**.
5. At the Database Unencryption page, shown below, select the column you want to unencrypt, and click **Continue**.



6. Preview the SQL statements, and click **Unencrypt Columns**. At this point, the unencrypt process begins. The user interface indicates when the process is complete.

Creating Views and Triggers

Views and triggers allow your applications to interact with the encrypted data in a seamless fashion. When you create a view on a table, the base table is renamed and a view is created with the original name of the table. Additionally, unique triggers are created for inserts and updates.

View/Trigger Settings

Use the Views and Triggers Settings: View/Trigger Settings section to view and modify the names of the new table, the view, the insert trigger, and the update trigger.

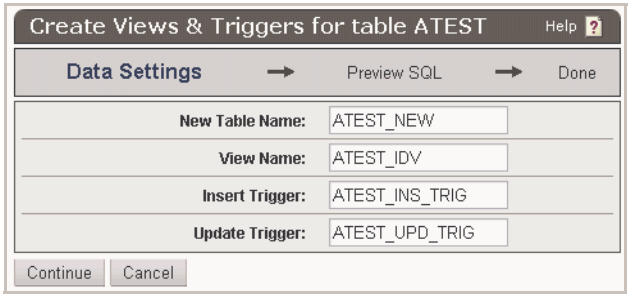
To create views and triggers, a user must have the following permissions within the schema in which the table resides:

- Connect.
- Create any view.
- Drop any view.
- Create any trigger.
- Drop any trigger.

Note: If a table contains a column of type `LONG` or `LONG RAW`, you cannot create views and triggers against that table in Oracle environments.

To create views and triggers:

1. Navigate to the Database Configuration page.
2. Click on the database that contains the table with the encrypted column.
3. At the Database Properties page, click on the table for which you want to create triggers and views.
4. At the Database Table Properties page, scroll down to the section entitled Table Operations, and click **Create Views and Triggers**.



5 Confirm the settings for the table and click **Continue**.

If the table contains a column or columns for which default values are configured, you must decide how to apply `NULL` values passed to the insert trigger. Use the Data Settings section to instruct the trigger to insert the default value or to insert a `NULL` value.

Click Cancel to abort the process and return to the Database Table Properties page.

6 Preview the SQL and click **Create View/Triggers**.

You cannot edit this code.

If you look in the database at this point, you should notice that the base table was renamed. If you go to the Views section of the database you're working in, you should be able to select the view with the original name of the base table and return all rows. Your data is displayed in plaintext.

Note: If you maintained any SQL*Plus sessions while creating the views and triggers, you need to close those sessions and reconnect to repopulate the meta-data cache. Otherwise SQL*Plus sessions continue to use the obsolete meta-data.

Deleting Views and Triggers

When you delete the views and triggers associated with a table, the base table is renamed back to the original name, and applications are no longer able to seamlessly interact with any encrypted data in the column.

To delete a view and triggers associated with a table:

1 Navigate to the Database Tools page.

2 Click on the database that contains the table with the encrypted column(s).

3 At the Database Properties page, click on the table for which you want to delete triggers and views.

4 Scroll down to the section of the page entitled Table Operations, and click **Delete Views and Triggers**.

5 Preview the SQL code that will be executed on your database. You cannot edit this code.

6 Click **Continue** to execute the SQL code.

If you look in the database at this point, you should notice that the base table was renamed back to its original name. If you go to the Views section of the database you're working in, you notice that the view has been deleted. If you navigate to the tables section of your database, and examine your data, it displays in ciphertext.

Click **Back** to return to the Database Table Properties page.

Click **Cancel** to stop the migration process.

Creating Domain Indexes

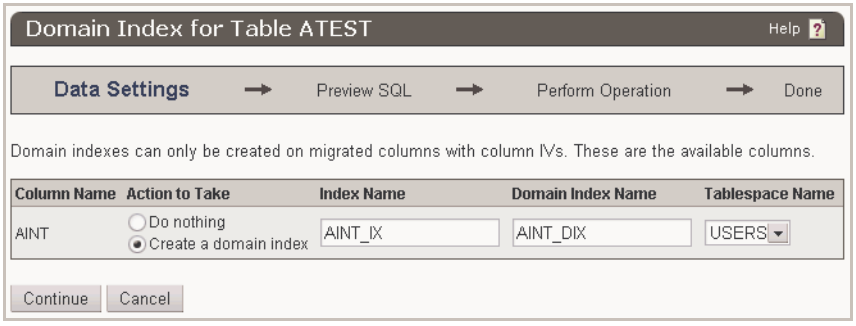
After you have migrated a table with column–level IVs in Oracle 9i or greater, you can create domain indexes on the encrypted columns. A domain index increases the speed of equality queries on encrypted columns.

Views and triggers are recreated when domain indexes are added or removed from a table.

When views and triggers are deleted, or when a column is unencrypted, the domain indexes are automatically removed.

To create domain indexes, follow the steps below:

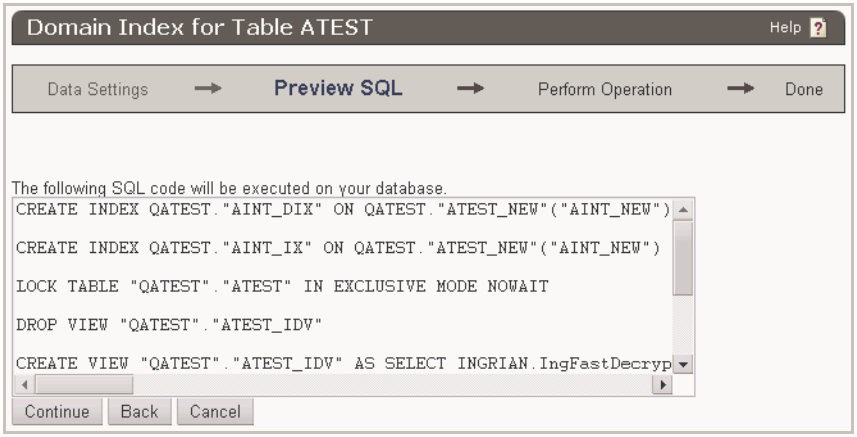
1. Navigate to the Database Tools page.
2. Click on the database that contains the table with the encrypted column.
3. At the Database Properties page, click on the table for which you want to create domain indexes.
4. At the Database Table Properties page, scroll down to the section entitled Table Operations, and click **Domain Index**.



5. Select the **Create a domain index** button for the encrypted column(s) for which you would like to create a domain index. To perform no action on a column, be sure that the **Do Nothing** button is selected. Enter values in the **Index Name**, **Domain Index Name** or accept the default values. Select the tablespace from the pulldown menu. Click **Continue**.

   Note: Domain indexes cannot be created on columns encrypted with row–level IVs.

6. Preview the SQL and click **Continue**.



Once a domain index is created, you can again select the **Domain Index** button to remove the domain index or to create domain indexes on additional columns.

Note: Once a domain index is created, the describe view does not show the scale or precision for the column.

Key Rotation

The Key Rotation feature provides an added level of security for your sensitive data, limiting the amount of time that a malicious user has to try and decrypt your data. Regular rotation of keys should be included as a regular part of a security maintenance plan.

Before you can rotate keys, you must create keys of the same type and size as the old keys. In addition, it is important that the group permissions are exactly the same in the new key as they were in the old key.

Understanding Temporary Tables

Temporary tables perform a very important function in key rotation and upgrade operations. The following information is copied from the base table to the temporary table before the data is processed:

- The ciphertext columns you defined for the operation.
- IV columns.
- Identity columns.

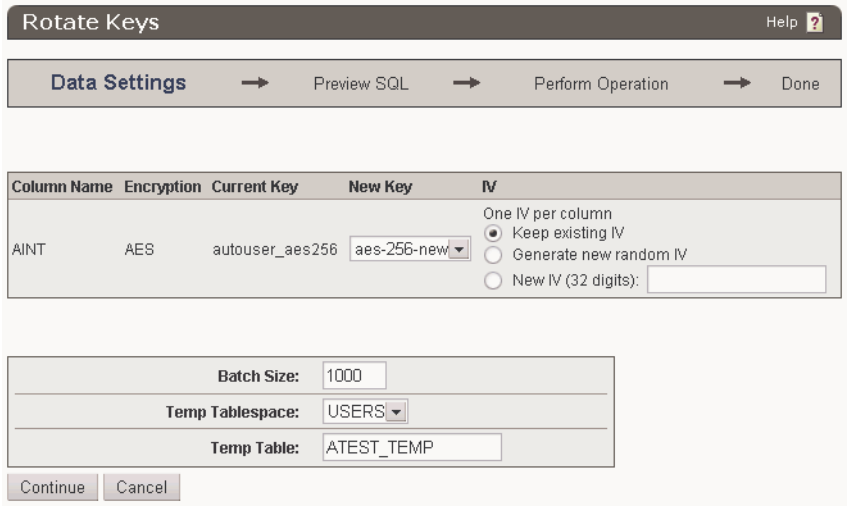
WARNING! You should delete the temporary table when the operation is complete and you have verified your data. You can delete the temporary tables from the Management Console. For more information, please see “Removing the Temporary Table” on page 100.

The following steps describe the sequence of events for rotating keys:

1. If a temporary table exists, the table is dropped. Then the column containing the ciphertext is copied to a temporary table along with the identity column and the IV column if there is one.
2. The ciphertext is sent to the NAE Server along with the IV for decryption.
3. The data is decrypted, a new IV is generated (if necessary), and the data is encrypted with the new key.
4. The NAE Server returns the ciphertext data to the database, and the ciphertext is inserted into the original column.

Key Rotation Settings

Use the Key Rotation Settings to set the parameters for the new encryption.



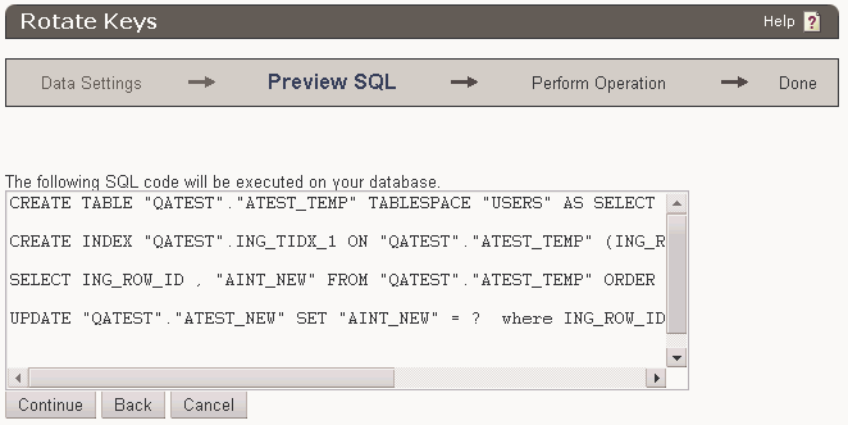
The following table describes the components of the Confirm Key Rotation section.

Component	Description
Column Name	The encrypted columns available for key rotation.
Encryption	Displays the algorithm used to encrypt the column.
Current Key	Displays the key that was used to encrypt the column.
New Key	Select the new key to encrypt the column.

Component	Description *(continued)*
IV	Select the method used to generate the IV. You can keep the existing IV, use a new random IV, or enter a specific IV.
Batch Size	Enter the batch size for the key rotation process.
Temp Tablespace	Select the temp tablespace on which to create the temp table.
Temp Table	Displays the temp table that will be used during the key rotation.
Continue	Click **Continue** to access the Preview SQL section and view the SQL code.
Cancel	Click **Cancel** to abort this process and return to the Database Table Properties page.

Preview SQL

Use the Preview SQL section to view the SQL that will be executed.



The following table describes the components of the Preview SQL Code section.

Component	Description
SQL Code	Displays the SQL code that will be executed on your database. You cannot edit this code.
Continue	Click **Continue** to launch the key rotation process.
Back	Click **Back** to return to the Key Rotation Settings section.
Cancel	Click **Cancel** to abort this process and return to the Database Table Properties page.

Rotating Keys

Follow the steps below to rotate keys:

1. Navigate to the Database Tools page.
2. Click on the database that contains the table with the encrypted column.
3. At the Database Properties page, click on the table for which you want to rotate keys.
4. Scroll down to the section of the page entitled Table Operations, and click **Key Rotation**.
5. Select the new key, enter an IV if necessary, and specify the batch size. Click **Continue**.

   The IV you specify for operations that use DES or DESede keys must be eight bytes; for AES keys, the IV must be sixteen bytes. If you are going to specify an IV here, the IV must be base 16 encoded.

   Though the maximum batch size is 100,000, the batch size you specify here should not exceed 10,000.
6. Preview the SQL and click **Rotate Keys**. Because this process requires that all the ciphertext is decrypted, new IVs generated, and data encrypted again, the key rotation process takes more time than any of the operations described above.

Note: If you maintained any SQL*Plus sessions during the key rotation, you need to close those sessions and reconnect to repopulate the meta-data cache. Otherwise the SQL*Plus session continues to use the obsolete meta-data.

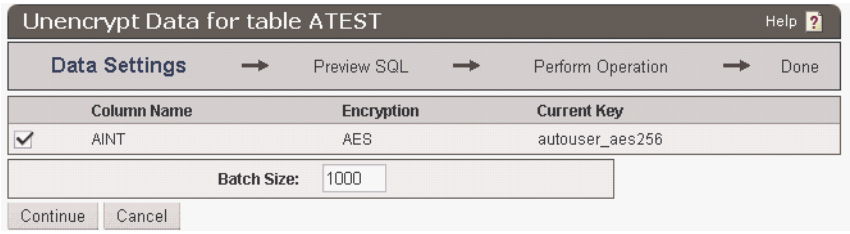
Unencrypt Columns

The following process describes what happens when you unencrypt a column. For the sake of simplicity, it is assumed that you will be unencrypting one column in the table.

1. The column containing the ciphertext is copied to a temporary table along with the identity column and the IV column if there is one.
2. The ciphertext is sent to the NAE Server along with the IV for decryption.
3. The NAE Server returns the plaintext data to the database, and the plaintext is inserted into the original column. Remember that when the data was originally migrated, another column was created and the original column was set to NULL.
4. After the plaintext data is inserted into the column, any new columns that were created during the migration are dropped. These might include the column that held the ciphertext, the IV column, and the identity column.

Confirm Operations

Use the Confirm Unencryption: Confirm Operations section to set the parameters for the unencryption.

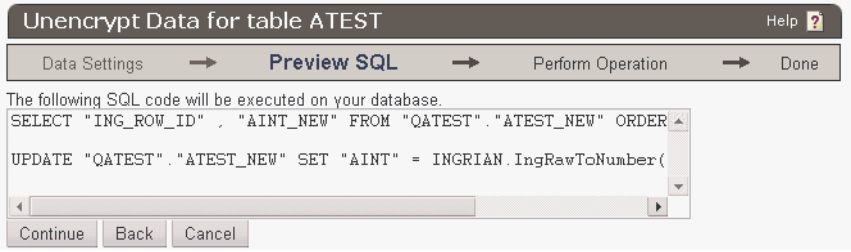


The following table describes the components of the Confirm Unencryption: Confirm Operations section.

Component	Description
Column Name	The encrypted columns available for unencryption.
Encryption	Displays the algorithm used to encrypt the column.
Current Key	Displays the key that was used to encrypt the column.
Batch Size	Enter the batch size for the key rotation process.
Continue	Click **Continue** to access the Preview SQL section and view the SQL code.
Cancel	Click **Cancel** to abort the process and return to the Database Table Properties page.

Preview SQL

Use the Preview SQL section to view the SQL that will be executed.



The following table describes the components of the Confirm Unencryption: Preview SQL section.

Component	Description
SQL Code	Displays the SQL code that will be executed on your database. You cannot edit this code.
Continue	Click **Continue** to launch the unencryption process.
Back	Click **Back** to return to the Confirm Operations section.
Cancel	Click **Cancel** to abort the process and return to the Database Table Properties page.

Removing the Temporary Table

As mentioned above, the temporary tables created during key rotation, and upgrades are very important. You should only delete a temporary table when you know that you have no need for the data in the table. For example, the temporary table is used for resume and restore operations. If you might be performing a resume or restore operation, then you need the temporary table. On the other hand, you do not want to leave the temporary table in the database any longer than necessary.

To remove a temporary table:

1. In the Management Console, navigate to the Database Tools page.
2. Click on the database that contains the temporary table.
3. Click on the Table that contains the sensitive data you just encrypted.
4. Scroll down to the section of the page entitled Table Operations, and click Remove Temp Table.
5. At the Remove Temp Tables page, the Management Console prompts you to confirm the removal of the temporary table. Click Remove Table to proceed.

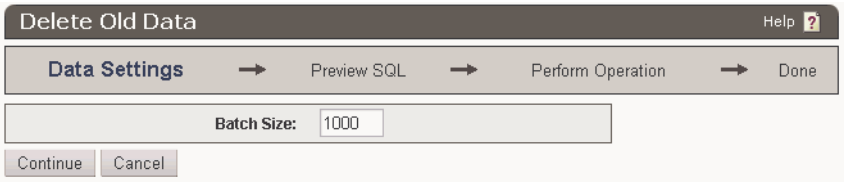
Note: When using Oracle 10gR2, the Remove Temp Table operation (available after a key rotation) renames the temp table and places it in the recycle bin. The temp table remains there until the recycle bin reaches its memory capacity or is purged.

To remove the temp table from the recycle bin, you should either purge the entire recycle bin by executing a PURGE RECYCLEBIN statement, or purge the temp table, as shown below.

```
SQL> SHOW RECYCLEBIN;
ORIGINAL NAME    RECYCLEBIN NAME    OBJECT TYPE  DROP TIME
-------------    ---------------    ------------ ---------
TSAN_TEMP        BIN$GfucBoABS      TABLE        2006-08-01:14
SQL> PURGE TABLE "BIN$GfucBoABS";
Table purged.
```

Delete Old Data

After you migrate your table, a new row of encrypted data is added, but the original data still exists in plaintext. We strongly recommend that you remove this data.



The following table describes the components of the Delete Old Data: Preview SQL section.

Component	Description
Batch Size	Enter the batch size for the deletion process.
Continue	Click **Continue** to execute the SQL code and remove the temporary table.
Cancel	Click **Cancel** to abort the process and return to the Database Configuration page.

Modifying Encrypted Tables

We strongly discourage you from manually modifying any of the metadata tables associated with the DataSecure Appliance. Doing so could result in data loss. Likewise, you should take great caution when modifying the base table. For example, if you use the Database Tools to create a view and triggers for a particular table, and you manually delete that view, you cannot restore the table or unencrypt the columns in the table.

Likewise, once you have migrated a table, it is strongly recommended that you not do any of the following:

- Drop or rename the encrypted table.
- Drop or rename the encrypted column.
- Change the data type or extend the length of an encrypted column.
- Modify the data type of the encrypted column.
- Rename the IV column.
- Rename a view created by the NAE Server.
- Rename a trigger created by the NAE Server.

Please note that this is not an exhaustive list of the things you can do to put your encrypted database into a dysfunctional state.

Altering an encrypted table can involve deleting and recreating views and triggers as well as decrypting and encrypting data. The sections below illustrate how to add, delete, and resize columns on an encrypted table.

Adding an Unencrypted Column

1. Delete the triggers and views using the Management Console.
2. Add the column to the base table.
3. Recreate the triggers and views using the Management Console.

Adding a Column to Encrypt

1. Add the column to the base table.
2. Configure the column for encryption using the Management Console.
3. Migrate the data. The migration process deletes the old views and triggers and create new ones using the new column.

Deleting an Unencrypted Column

1. Delete the triggers and views using the Management Console.
2. Delete the column from the base table.
3. Recreate the triggers and views using the Management Console.

Deleting an Encrypted Column

1. Unencrypt the column using the Management Console.
2. Delete the triggers and views using the Management Console.
3. Delete the column from the base table
4. Recreate the triggers and views using the Management Console.

Resizing an Unencrypted Column

Modify the column size in the base table.

Resizing an Encrypted Column

1. Unencrypt the column using the Management Console.
2. Modify the column size in the base table.
3. Configure the column for encryption using the Management Console.
4. Migrate the data. The migration process deletes the old views and triggers and create new ones using the new column.

Granting or Revoking Table and Column Permissions

1. Delete the triggers and views using the Management Console.
2. Execute the necessary grant or revoke SQL statements.
3. Recreate the triggers and views using the Management Console.

CHAPTER 7

Using the Bulk Loader

Important! **This feature has been deprecated.**

This chapter discusses how to set up and run the Bulk Loader, and provides some examples of common usage. These examples guide you through the entire process, from creating tables to encrypting and decrypting data to inserting processed data back into the base table. This chapter contains the following topics:

Supported Data Types

The following table shows the data types supported by the Bulk Loader.

- CHAR
- DATE
- DECIMAL
- NCHAR
- NUMBER
- NUMERIC
- NVARCHAR
- VARCHAR
- VARCHAR2

Configuring Oracle to Use the Bulk Loader

Before you can use the Bulk Loader, you must:

- Configure Oracle to support our external procedures.
- Configure memory allocation, when using AIX.
- Grant permissions to the `INGRIAN` user.
- Configure SSL, if desired.

  For information on how to configure SSL, see “Setting up SSL” on page 49.

Configuring Memory Allocation on the AIX Operating System

When using the Bulk Loader on an AIX operating system you might have to reset the MAXDATA loader parameter, depending on your batch size. After resetting this parameter to the new memory allocation, you must also:

1. Update the Oracle user’s UNIX shell profile to include:

   ```
   export LDR_CNTRL_MAXDATA="<new_memory_allocation>"
   ```

   WARNING! You must carefully consider the new memory allocation as it affects other applications. Setting the parameter to 0x60000000 should be sufficient for most systems.

2. Restart the database.
3. Restart the listener.

Granting Permissions to the INGRIAN User

For users to run the Bulk Loader on tables in their own schema, they must create their own output table and then grant the following permissions to the `INGRIAN` user:

- Select on <base table>.
- Select on <input table>.
- Insert on <output table>.

Running the Bulk Loader

To use the Bulk Loader to encrypt and decrypt data, you must:

- Configure a job.

- Run the job.

Configuring a Job

A job is a set of parameters that specify what data the Bulk Loader processes, how it processes the data, and where the results are stored. These parameters are saved on the `ING_BULK_LOAD_CONFIG` table and retrieved whenever the job name is passed to the `ING_BULK_LOAD` procedure.

Once defined, you can reuse a job configuration repeatedly.

When configuring a job, you must pass the following parameters:

- **Job Name** – A string between 1 and 30 characters long that uniquely defines the job. Reusing a previously-defined job name overrides the old values.
- **Cryptographic Process** – The process to be performed. Valid values are **ENCRYPT** or **DECRYPT**.
- **Owner Name** – The owner of the base, input, and output tables. This value is required to access these tables.
- **Base Table** – Name of the table that was previously encrypted using the Data Migration utility in the Management Console. The Bulk Loader bases new encryption or decryption requests on the process used to encrypt this table.

  For more information on base tables, see “Selecting a Base Table” on page 106.
- **Input Table** – The table specified here must contain the data to be processed.

  For more information on input tables, see “Selecting an Input Table” on page 107.
- **Output Table** – Name of the table that receives the result of the bulk load process. If the output table does not exist, the Bulk Loader creates the table and assign ownership to the owner specified above.

  For more information on output tables, see “Preparing an Output Table” on page 108.
- **Tablespace** – Tablespace where the output table is created. This parameter is ignored if the table already exists.
- **Batch Size** – The number of rows processed and stored in memory before the results are committed to the output table.

  Jobs that specify batch sizes larger than 10,000 are rejected. For the Bulk Loader to accept larger batch sizes, you must increase the `MAX_BATCH_SIZE` value listed on the `ING_PROPERTY` table.

Increasing batch size can increase your performance speed, but also requires more system resources. While the Bulk Loader can accept 100,000 rows per batch, batch sizes over 50,000 does not improve performance.

Tip: If system resources are running low while a Bulk Loader job is running, try using a smaller batch size.

- **Return Code Variable** – The Bulk Loader uses this variable to pass the return code generated by the process. A value of **0** indicates a successful job run. A value less than zero indicates a failure.
- **Return Code Text Variable** – The Bulk Loader uses this variable to pass the return code text generated by the process.

Prior to running the job, you should define variables for the return codes.

The SQL statement used to define a job takes the following form:

```
DECLARE     retStat NUMBER; retTxt VARCHAR2(255);

BEGIN       ing_bulk_load_pkg.ing_bulk_load_setup
         ('name_of_job', /* Job Name                    */
         'encrypt',     /* Job Type (encrypt or decrypt)  */
         'TABLE_OWNER', /* Base table owner            */
         'BASE_TABLE',  /* Base table name             */
         'INPUT_TABLE', /* Input table name            */
         'OUTPUT_TABLE', /* Output table name          */
         'TABLESPACE',  /* Output tablespace           */
         1000,          /* Batch size                  */
         retStat,       /* Return code                 */
         retTxt         /* Return code text            */
         );

END;
/
```

Note: The values used to configure a job are case-sensitive.

Selecting a Base Table

The base table is a previously-migrated table that serves as a model for your new cryptographic job. The Bulk Loader accesses the encryption metadata associated with the base table and uses it to process the input table. The same key, mode, algorithm, and IV used to encrypt the base table columns are used to encrypt and decrypt the corresponding columns in the input table. These corresponding columns must share the same column name, data type, and width.

If you have created triggers and views on the base table, then you should specify the original name of the base table.

Selecting an Input Table

The input table contains the values that you want to encrypt or decrypt. These columns must have the same structure as their counterparts on the base table: the same column names, data types, and widths.

Tip: We recommend that you do not generate the input table from a view, if one exists. There can be structural differences between the view and the original table that inhibit your operation.

The Bulk Loader does not perform a cryptographic operation on any column in the input table that was not encrypted in the base table.

If you use the same table as both the base and the input table, all columns that have been encrypted with the Data Migration utility are processed by the Bulk Loader. You might use this setup when decrypting data.

If there are multiple encrypted columns in your base table, and you only want to use the Bulk Loader to process one of those encrypted columns, or if you only want to process a subset of data in one column, then you should create a separate table or view, and use that as your input table.

The input table *must* have the following structure:

- An `ING_ROW_ID` column of data type `NUMBER`.

  This column must be populated with values from the base table sequence.
- A unique index on the `ING_ROW_ID` column.

If the input table does not have an indexed `ING_ROW_ID` column of data type `NUMBER`, the Bulk Loader job fails. If the input table has all of these requisite parts, then the Bulk Loader uses the `ING_ROW_ID` column to connect the input and output tables.

You can run the bulkLoadPrep.sql script to check if the `ING_ROW_ID` column, sequence and index exist, and add them if needed. This script requires that you own the job to which the input table belongs, or have the correct permissions in the input table.

Alternatively, you can add the `ING_ROW_ID` column and follow the steps outlined below.

To populate the `ING_ROW_ID` column with values from the base table sequence:

1. Find the name of the desired sequence by calling the `ing_get_sequence_for_table` procedure using the following SQL statement:

   ```
   SELECT ingrian.ing_get_sequence_for_table('<BASETABLE>') FROM DUAL;
   ```

2. Apply the sequence using the following SQL statement:

   ```
   UPDATE my_input_table SET ING_ROW_ID = <basetable_seq>.nextval;
   ```

3. Create an index on the `ING_ROW_ID` column using the following SQL statement:

   ```
   CREATE UNIQUE INDEX my_input_table_idx ON my_input_table(ING_ROW_ID);
   ```

Preparing an Output Table

The output table stores the data created by the Bulk Loader job. You specify the name of the output table and can use either an existing table, or name a nonexistent one.

To prepare an output table:

- Select an existing table or specify a name for a new table.
- Set the output table structure.
- Set output table permissions.

SELECTING AN EXISTING TABLE

By specifying an existing table, you have the option of creating a table with any set of creation parameters that you want.

If the output table already exists when the process starts, it is validated to ensure it has the expected columns.

The Bulk Loader never drops an existing output table and recreates it. If the output table does not contain all of the encrypted columns present in the input table, the load process exits without processing any data. To avoid this, you either have to manually update the output table before processing a job, or enter a nonexistent table that the Bulk Loader creates.

Tip: If you use an existing output table, we recommend that you drop any existing indexes on this table to improve the performance of the Bulk Loader. You can recreate the indexes when the job is done. For more on recreating indexes, see “Creating Indexes on the Output Table” on page 112.

SPECIFYING A NAME FOR A NEW TABLE

If you enter the name of a table that does not exist in the specified tablespace, the Bulk Loader creates it while running the job. This table inherits the database owner and tablespace parameters associated with the job name.

SETTING THE OUTPUT TABLE STRUCTURE

The output table *must* contain the columns listed below. If the Bulk Loader creates the output table, it automatically includes these columns.

- `ING_ROW_ID` of column type `NUMBER`. This identity column stores the values from the identity column in the input table.
- The encrypted or decrypted value column(s). Each column in the input table that holds an encrypted or decrypted value must have a counterpart of the *same name* on the output table. The output table column must have the data type and width needed to receive the encrypted or decrypted values; these values differ from those of the input table column.
- The IV column(s), if field-level IVs are used. Each IV column on the base table must have a counterpart on the output table. The column on the output table must have the same name, data type, and width.

SETTING OUTPUT TABLE PERMISSIONS

If you are encrypting tables that are not in the `INGRIAN` schema, you should manually create the output table and grant permission to the `INGRIAN` user to insert into that table. Otherwise, the process fails. This is because the Bulk Loader runs as the `INGRIAN` user, which does not automatically have permission to insert into an output table created in your schema.

Running a Job

To run a job, you must call the bulk load procedure and pass the configuration values from the `ING_BULK_LOAD_CONFIG` table.

Important! If the output table already contains data, remove that data before running the Bulk Loader.

When running a job, you must pass the following parameters:

- **Job Name** – The job name references the parameters listed on the `ING_BULK_LOAD_CONFIG` table.
- **Resume Flag** – This field determines whether a job that ended before completion resumes from the point at which it stopped. A **Y** indicates that the process resumes where it left off. An **N** indicates that the process starts from the beginning.
- **Return Code Variable** – The Bulk Loader uses this variable to pass the return code generated by the process. A value of **0** indicates a successful job run. A value less than zero indicates a failure.
- **Return Code Text Variable** – The Bulk Loader uses this variable to pass the return code text generated by the process.

Prior to running the job, you should define variables for the return codes.

The SQL statement used to run a job takes the following form:

```
DECLARE     retStat NUMBER; retTxt VARCHAR2(255);

BEGIN       ing_bulk_load_pkg.ing_bulk_load
         ('name_of_job', /* Job Name            */
         'N',           /* Resume flag         */
         retStat,       /* Return code         */
         retTxt         /* Return code text    */
         );

END;
/
```

Note: The values used to configure a job are case-sensitive.

Managing Space to Improve Performance

The Bulk Loader's performance is significantly affected by the speed with which it can insert values into the output table. We suggest that you:

- Allocate enough space on the output table to minimize the number of new extents added during the insert process.
- Set the tablespace rollback size for a large insert.
- Size the redo logs to minimize the amount of log switching that occurs during the insert process.

Truncating the Output Table

You can truncate the output table *before* adding new rows by including the truncate output table flag when calling ing_bulk_load.

Important! The `ingrian` user must have `drop any table` system privileges to use this feature. This privilege is not granted during the installation process; you must do it manually.

The SQL statement used to run a job - and truncate the output table - takes the following form:

```
DECLARE     retStat NUMBER; retTxt VARCHAR2(255);

BEGIN       ing_bulk_load_pkg.ing_bulk_load
         ('name_of_job', /* Job Name                    */
         'N',           /* Resume flag                 */
         'Y',           /* Truncate Output Table flag */
         retStat,       /* Return code                 */
         retTxt         /* Return code text            */
         );

END;
/
```

Monitoring the Job from the Database

The recommended way to monitor Bulk Loader jobs is to view the contents of the `ING_JOB_LOG` table. This table contains the complete job log information for every run of every Bulk Loader job. The data is committed as it is logged, so you can view a job while it is in progress.

The SQL statement used to monitor a job takes the following form:

```
col descr format a80
set pagesize 100
set linesize 100

SELECT      log_seq

AS          id, to_char(create_dt, 'HH24:MI:SS')

AS          time, log_txt

AS          descr

FROM        ing_job_log

WHERE       job_id =
         (SELECT MAX(job_id)
         FROM ing_bulk_load_run);
```

You will most likely run the Bulk Loader from a script, as opposed to running it from the command line.

When running a job from the command line, you may preface the SQL statement with a “set serveroutput on size 1000000” command to view the output logging from your terminal.

The maximum amount of data that Oracle can hold in the serveroutput buffer is 1,000,000 bytes. For large jobs (those in excess of 20,000 batches), you should not set serveroutput on, as this causes the server output buffer to overflow.

Tip: All data displayed on the screen during the job run is logged to the table `ING_JOB_LOG`. The data displayed through the serveroutput feature is not available until the job is completed. To monitor data as it is being logged, query the table `ING_JOB_LOG`, as described above.

Monitor the Job from the Outside Database

You can monitor a job from outside of the database by accessing the log file generated on the database server. The name and location of this file is determined by the database log values specified in the `IngrianNAE.properties.sql` file.

Canceling a Job

To cancel a job while it is running, you can issue a SQL statement from the command line. When the statement is issued, the job stops after the current batch is completed.

For example, the Bulk Loader is processing a table of 10,000 rows with a batch size of 1000. A cancel command is issued when the Bulk Loader is processing row 5020. The process stops after the Bulk Loader has processed row 6000 and then commits values 5001 - 6000 to the output table.

The SQL statement used to cancel a job takes the following form:

```
BEGIN    ing_bulk_load_pkg.ing_bulk_load_cancel ('name_of_job');

END;
/
```

Working with Output Data

When the Bulk Load job is completed, you can:

- Create indexes.
- Join and insert data.
- Truncate the output table.

Creating Indexes on the Output Table

Create any indexes on the output table required to optimize performance for your specific process. If the output table will be joined with the input or base table, we recommend that you also create a unique index on the `ING_ROW_ID` column.

To create an index on the `ING_ROW_ID` column, use the following SQL statement:

```
CREATE UNIQUE INDEX my_output_table_idx on my_output_table(ING_ROW_ID);
```

Joining and Inserting Data

Once the output table contains the results of the Bulk Loader job, you may want to transfer the data to the base table.

You need to transfer the encrypted data, as well as any IVs. If the encrypted column and its corresponding IV are not kept in sync, it will be impossible to decrypt the data.

The SQL statement used to insert the data takes the following form:

```
INSERT INTO     base_table
            (FIRST, LAST, ING_ROW_ID, CC_NUM_NEW, CC_NUM_IV)
VALUES         (SELECT FIRST,
                    LAST,
                    it.ING_ROW_ID,
                    CC_NUM_NEW,
                    CC_NUM_IV

            FROM    input_table it,
                    output_table ot

            WHERE   it.ING_ROW_ID = ot.ING_ROW_ID
            );
```

Truncate the Output Table

We recommend that you truncate the output table as soon as processing is complete. When encrypting data, this avoids the possibility of leaving plaintext values in the database.

Managing the Bulk Loader Tables

During the configuration and running of Bulk Loader jobs, several tables are populated with data. An overview of these tables, as well as advice on maintaining them, is provided below.

- `ING_BULK_LOAD_CONFIG` – This table stores one row for each Bulk Loader job you define. This row contains the parameters required to run and resume the job. Deleting an entry from this table removes the job configuration and prevents you from launching or resuming a job.
- `ING_BULK_LOAD_RUN` – This table stores the parameters supplied for a specific Bulk Loader job run. A new row is created each time the Bulk Loader is run. This row is referenced when a job is launched or resumed. Rows related to current jobs, or jobs that may need to be resumed should not be removed. When deleting entries, you should use Job ID or date as your selection criteria.
- `ING_JOB` – This table stores the details of a data migration operation. A new row is created each time the Bulk Loader is run. This row is used to determine where cancelled jobs should resume. When deleting entries, you should use Job ID or date as your selection criteria.
- `ING_JOB_BATCH` – This table stores the status of a migration job. A new row s create for each processed batch and for pre-processing information such as input and output table validation. This table is used to determine where cancelled jobs should resume. When deleting entries, you should use Job ID or date as your selection criteria.

  For example, when processing 100 rows with a batch size of 10, there are 11 entries: 1 entry for each batch, and 1 for the pre-processing information.
- `ING_JOB_LOG` – This table stores the complete job log information for every run of every Bulk Loader job. This table can grow quickly; for example, one run of one job consisting of 10 batches could yield 300 rows of data or more in this table. When deleting entries, you should use Job ID or date as your selection criteria.

Examples

This section contains two examples of using the Bulk Loader:

1. Encrypting data and inserting it into the base table.
2. Decrypting data and selecting the new values.

Both examples involve an online retailer that uses an Oracle database to store customer names and credit card numbers. The `CUSTOMER` table contains information about existing customers and the `CUSTOMER_DAILY` table contains information about new customers.

To complete both examples, you must set up the `CUSTOMER` and `CUSTOMER_DAILY` tables.

Setting Up the CUSTOMER Table

To set up the CUSTOMER table, enter the following SQL statement:

```
CREATE TABLE    CUSTOMER (
            FIRST VARCHAR2(30),
            LAST VARCHAR2(30),
            CC_NUM VARCHAR2(16)
            );

INSERT INTO     CUSTOMER
VALUES          ('Irwin', 'Fletcher', '1234567890123456');

INSERT INTO     CUSTOMER
VALUES          ('Steve', 'Garvey', '9876543210654321');

INSERT INTO     CUSTOMER
VALUES          ('Josh', 'Ritter', '1111222233334444');

COMMIT;
```

After you create the CUSTOMER table, use the Management Console to set the column properties of CC_NUM so that the column is encrypted with a random IV for each field. Migrate the table, delete the old data, and then create views and triggers. Then use the ing_get_sequence_for_table procedure to find the sequence used to populate the ING_ROW_ID column.

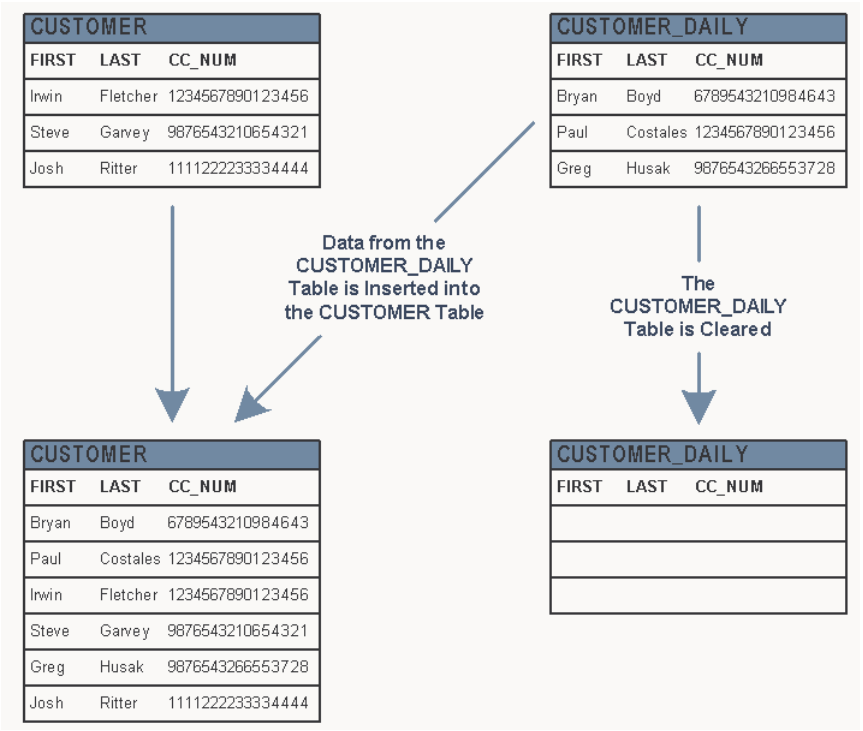
Setting Up the CUSTOMER_DAILY Table

To set up the CUSTOMER_DAILY table, enter the following SQL statement:

```
CREATE TABLE    CUSTOMER_DAILY
AS              SELECT *
FROM            CUSTOMER
WHERE           0=1
INSERT INTO      CUSTOMER_DAILY (FIRST, LAST, CC_NUM)
VALUES          ('Bryan', 'Boyd', '6789543210984643');
INSERT INTO     CUSTOMER_DAILY (FIRST, LAST, CC_NUM)
VALUES          ('Paul', 'Costales', '1234567890123456');
INSERT INTO     CUSTOMER_DAILY (FIRST, LAST, CC_NUM)
VALUES          ('Greg', 'Husak', '9876543266553728');
ALTER TABLE CUSTOMER_DAILY add ING_ROW_ID NUMBER;
COMMIT;
UPDATE CUSTOMER_DAILY SET ING_ROW_ID = <seq>.nextval;
--where <seq> is the sequence from the CUSTOMER table.
CREATE UNIQUE INDEX CUST_DAILY_idx on CUSTOMER_DAILY (ING_ROW_ID);
```

Example 1: Encrypt Data and Insert Into Base Table

The values on the CUSTOMER_DAILY table are inserted into the CUSTOMER table and then cleared at the end of each day. Before introducing the SafeNet Data Privacy Solution, the process worked as shown here.



The online retailer now wants to encrypt the customer credit card numbers stored in the CUSTOMER table. (You've already done this after creating the CUSTOMER table, above.)

During the data migration process, the CUSTOMER table is renamed CUSTOMER_NEW, and gains the following columns:

- ING_ROW_ID. Holds the identity column for the row.
- CC_NUM_NEW. Holds the encrypted credit card value.
- CC_NUM_IV. Holds the IVs for each encrypted credit card.

The new table is shown below.

CUSTOMER_NEW

FIRST	LAST	CC_NUM	ING_ROW_ID	CC_NUM_NEW	CC_NUM_IV
Irwin	Fletcher	NULL	1	10001010110101001...	100101...
Steve	Garvey	NULL	2	10111000010001011...	101001...
Josh	Ritter	NULL	3	11011001011010001...	110110...

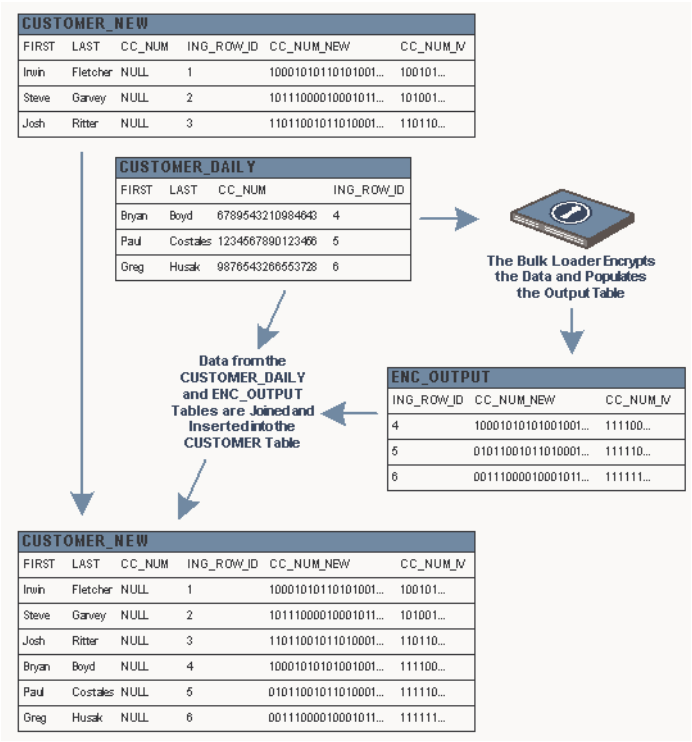
The daily process of inserting the CUSTOMER_DAILY table into the CUSTOMER table must change to include the ING_ROW_ID, CC_NUM_NEW, and CC_NUM_IV columns. These values must be calculated using the same encryption process that was used to migrate the CUSTOMER table.

The solution is to use the Bulk Loader to:

- Recreate the process used to migrate the CUSTOMER table.
- Encrypt the values on the CUSTOMER_DAILY table.
- Populate the output table with the encryption information.

Once the output table contains the encrypted data, it can be joined with the CUSTOMER_DAILY table, and inserted in the CUSTOMER table.

The new process is shown below.



To do this, you would:

1. Configure the encryption job.
2. Run the encryption job.
3. Join and insert the data.
4. Truncate the output table.

Configuring the Encryption Job

To configure this job, you would enter the following SQL statement:

```
set serveroutput on size 1000000

DECLARE     retStat NUMBER; retTxt VARCHAR2(255);

BEGIN       ing_bulk_load_pkg.ing_bulk_load_setup
         ('encrypt_customer_daily',  /* Job Name       */
         'encrypt',                  /* Job Type       */
         'INGRIAN',                  /* Base table owner  */
         'CUSTOMER_NEW',             /* Base table name   */
         'CUSTOMER_DAILY',           /* Input table name   */
         'ENC_OUTPUT',               /* Output table name   */
         'USERS',                    /* Output tablespace   */
         1000,                       /* Batch size     */
         retStat,                    /* Return code    */
         retTxt                      /* Return code text  */
         );

END;
/
```

Note: The Bulk Loader creates the `ENC_OUTPUT` table if one does not exist. If the `ENC_OUTPUT` table does exist, it must have the proper structure, otherwise the job ends without processing any data.

Running the Encryption Job

To run the job, you would enter the following SQL statement:

```
DECLARE     retStat NUMBER;
         retTxt VARCHAR2(255);

BEGIN       ing_bulk_load_pkg.ing_bulk_load
         ('encrypt_customer_daily',  /* Job Name
         */
         'N',                        /* Resume flag   */
         retStat,                    /* Return code   */
         retTxt                      /* Return code text*/
         );

END;
/
```

Joining and Inserting the Data

To join the CUSTOMER_DAILY and ENC_OUTPUT tables and insert the first, last, cc_num_enc, and cc_num_iv columns into the CUSTOMER_NEW table, you would enter the following SQL statement:

```
INSERT INTO     CUSTOMER_NEW
            (FIRST, LAST, ING_ROW_ID, CC_NUM_NEW, CC_NUM_IV)
VALUES         (SELECT FIRST,
                    LAST,
                    cd.ING_ROW_ID,
                    CC_NUM_NEW,
                    CC_NUM_IV

            FROM    CUSTOMER_DAILY cd,
                    ENC_OUTPUT eo

            WHERE   cd.ING_ROW_ID = eo.ING_ROW_ID
            );
```

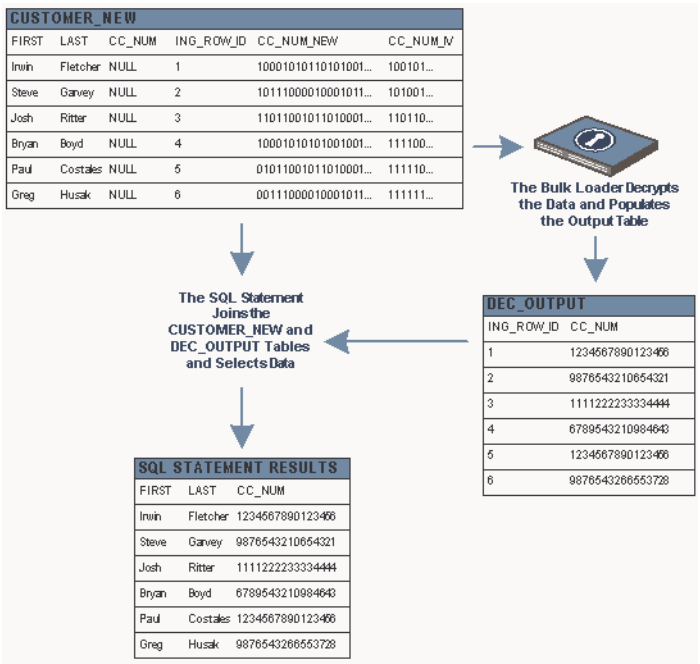
Truncating the Output Table

The output now contains row IDs, encrypted credit card numbers, and IVs. You should remove this data as soon as your processing is complete.

Example 2: Decrypt and Select

The online retailer now wants to search the CUSTOMER table for specific credit cards and their corresponding users. To do this, you would:

- Configure a decryption job.
- Run the decryption job.
- Join and select the data.
- Truncate the output table.



Configuring the Decryption Job

To configure this job, you would enter the following SQL statement:

```
set serveroutput on size 1000000
DECLARE     retStat NUMBER; retTxt VARCHAR2(255);
BEGIN       ing_bulk_load_pkg.ing_bulk_load_setup
        ('decrypt_customer_ccnum',    /* Job Name       */
        'decrypt',                  /* Job Type       */
        'INGRIAN',                  /* Base table owner  */
        'CUSTOMER_NEW',             /* Base table name   */
        'CUSTOMER_NEW',             /* Input table name  */
        'DEC_OUTPUT',               /* Output table name */
        'USERS',                    /* Output tablespace */
        1000,                       /* Batch size     */
        retStat,                    /* Return code    */
        retTxt                      /* Return code text  */
        );
END;
/
```

Running the Decryption Job

To run the job, you would enter the following SQL statement:

```
set serveroutput on size 1000000

DECLARE     retStat NUMBER; retTxt VARCHAR2(255);

BEGIN       ing_bulk_load_pkg.ing_bulk_load
         ('decrypt_customer_ccnum',  /* Job Name       */
         'N',                       /* Resume flag     */
         retStat,                   /* Return code   */
         retTxt                     /* Return code text*/
         );

END;
/
```

Joining and Selecting the Data

To select the first and last columns from the `CUSTOMER_NEW` table and the corresponding decrypted credit card number from the `DEC_OUTPUT` table, you would enter the following SQL statement:

```
SELECT      c.FIRST, c.LAST, do.CC_NUM

FROM        CUSTOMER_NEW c, DEC_OUTPUT do

WHERE       c.ING_ROW_ID = do.ING_ROW_ID
```

Truncate the Output Table

The output now contains row IDs and decrypted credit card numbers. You should remove this data as soon as your processing is complete.

CHAPTER 8

Setting up SSL

This chapter provides an overview of SafeNet, Inc.' SSL and SSL with Client Certificate Authentication features, and provides a walkthrough of both configuration procedures. This chapter contains the following topics:

SSL Overview

Standard SSL communication requires a certificate that identifies the server. This certificate is signed by a certificate authority (CA) known to both the server and the client. During the SSL handshake, the server certificate is passed to the client. The client uses a copy of the CA certificate to validate the server certificate, thus authenticating the server.

While the CA can be a third-party CA or your corporate CA, you will most likely use a local CA on the DataSecure appliance. If you are not using a local CA, consult your CA documentation for instructions on signing requests and exporting certificates.

Tip: SafeNet, Inc. recommends that you increase security *only after* confirming network connectivity. You should establish a TCP connection before enabling SSL. Otherwise, an unrelated network connection mistake could interfere with your SSL setup and complicate the troubleshooting process.

To use an SSL connection when communicating with the DataSecure appliance, you must configure both the server and the client.

To configure the server, you must:

- Create a server certificate. (If you're using a cluster, each member must have its own, unique certificate.)

  This may involve the following steps:
  - Creating a Local CA.
  - Creating a Server Certificate Request on the Management Console.
  - Signing a Server Certificate Request with a Local CA.
- Update the NAE Server settings on the Management Console (Device, NAE Server, NAE Server).

  You'll need to check **Use SSL** and select your server certificate in the **Server Certificate** field.

To configure the client, you must:

- Place a copy of the CA certificate on your client.

  This may involve the following step:
  - Downloading the Local CA Certificate.
- Update `IngrianNAE.properties` file as follows:
  - `Protocol=ssl`
  - `CA_File=<location and name of the CA certificate file>`

  Note: Whenever you update the properties file, you must restart the database for the changes to take effect.

SSL Configuration Procedures

This section describes the procedures you will follow when configuring SSL. It explains the following processes:

- Creating a Local CA
- Creating a Server Certificate Request on the Management Console
- Signing a Server Certificate Request with a Local CA
- Importing a Server Certificate to the DataSecure Appliance
- Downloading the Local CA Certificate

Creating a Local CA

To create a local CA:

1. Log on to the Management Console as an administrator with Certificate Authorities access control.
2. Navigate to the Create Local Certificate Authority section on the Certificate and CA Configuration page (Security, Certificates & CAs, Local CAs).
3. Modify the fields as needed.
4. Select either *Self-signed Root CA* or *Intermediate CA Request* as the **Certificate Authority Type**.
5. Click **Create**.

Note: Only a local CA can sign certificate requests on the DataSecure appliance. If you are using a CA that does not reside on the DataSecure appliance you cannot use the Management Console to sign certificate requests.

Creating a Server Certificate Request on the Management Console

To create a server certificate request on the Management Console:

1. Log on to the Management Console as an administrator with Certificates access control.
2. Navigate to the Create Certificate Request section of the Certificate Configuration page (Security, Certificates & CAs, Certificates) and modify the fields as needed.
3. Click **Create Certificate Request**. This creates the certificate request and places it in the Certificate List section of the Certificate and CA Configuration page. The new entry shows that the **Certificate Purpose** is *Certificate Request* and that the **Certificate Status** is *Request Pending*.

Signing a Server Certificate Request with a Local CA

To sign a server certificate request with a local CA:

1. Log in to the Management Console as an administrator with Certificates and Certificate Authorities access controls.
2. Navigate to the Certificate List section on the Certificate and CA Configuration page (Security, Certificates & CAs, Certificates).

3. Select the certificate request and click **Properties**.
4. Copy the text of the certificate request. The copied text must include the header (-----BEGIN CERTIFICATE REQUEST-----) and footer (-----END CERTIFICATE REQUEST-----).
5. Navigate to the Local Certificate Authority List (Security, Certificates & CAs, Local CAs). Select the local CA and click **Sign Request** to access the Sign Certificate Request section.
6. Modify the fields as shown:
   - **Sign with Certificate Authority** - Select the CA that signs the request.
   - **Certificate Purpose** - Select *Server*.
   - **Certificate Duration (days)** - Enter the life span of the certificate.
   - **Certificate Request** - Paste all text from the certificate request, including the header and footer.
7. Click **Sign Request**. This will take you to the CA Certificate Information section.
8. Copy the actual certificate. The copied text must include the header (-----BEGIN CERTIFICATE-----) and footer (-----END CERTIFICATE-----).
9. Navigate back to the Certificate List section (Security, Certificates & CAs, Certificates). Select your certificate request and click **Properties**.
10. Click **Install Certificate**.
11. Paste the actual certificate in the Certificate Response text box. Click **Save**. The Management Console returns you to the Certificate List section. The section will now show that the **Certificate Purpose** is *Server* and that the **Certificate Status** is *Active*.

The certificate can now be used as the server certificate for the NAE Server.

Importing a Server Certificate to the DataSecure Appliance

As an alternative to the certificate creation procedure outlined above, you can import a certificate to the DataSecure appliance.

To import a certificate to the DataSecure appliance:

1. Log in to the Management Console as an administrator with Certificates access control.
2. Navigate to the Import Certificate section of the Certificate and CA Configuration page (Security, Certificates & CAs, Certificates).
3. Select the method used to import the certificate file.
4. Enter the name of the file and the private key password.
5. Click the **Import Certificate** button.

The certificate can now be used as the server certificate for the NAE Server.

Downloading the Local CA Certificate

To download a local CA certificate from the DataSecure appliance:

1. Log in to the Management Console as an administrator with Certificate Authorities access control.
2. Navigate to the Local Certificate Authority List section of the Certificates and CA Configuration page (Security, Certificates & CAs, Local CAs).
3. Select the Local CA and click the **Download** button to download the file to your client. You should place the CA certificate in a secure location and modify access appropriately.

Note: Use the `CA_File` parameter in the IngrianNAE.properties file to indicate the name and location of the CA certificate.

SSL Walkthrough for SafeNet Clients

This walkthrough assumes the following:

- You have read the SSL overview section.
- You have configured a TCP connection between your client and the DataSecure appliance.

There are a few different ways that you could configure SSL. For example, you can use a CA that does not reside on the DataSecure appliance or you can create a new one. This walkthrough makes such decisions for you. By following these instructions, you will:

- Create a Local CA.
- Create a Certificate Request.
- Create a Server Certificate by signing the Certificate Request with the Local CA.
- Download the Local CA to the client.

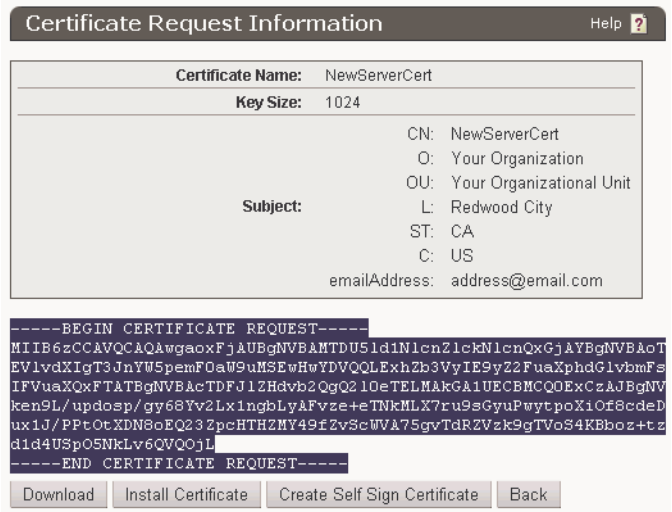
Once you have completed and understood this walkthrough, you might decide to alter some of the steps to better fit your organization's policies.

To configure SSL:

1. Log in to the Management Console as an administrator with Certificates, Certificate Authorities, and NAE Server access controls.
2. Navigate to the Create Local Certificate Authority section (Security, Certificates & CAs, Local CAs). Enter the values shown below to create a new local CA. Click **Create**.

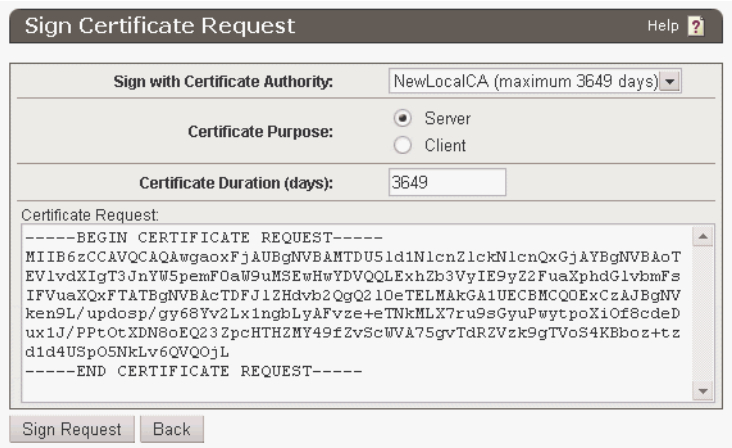
3 Navigate to the Create Certificate Request section (Security, Certificates & CAs, Certificates). Enter the values shown below to create a request. Click **Create Certificate Request**.

4 Select your new certificate request from the Certificate List section (right above the Create Certificate Request section). Click **Properties**. Copy the actual request (highlighted below). Include the header and footer.



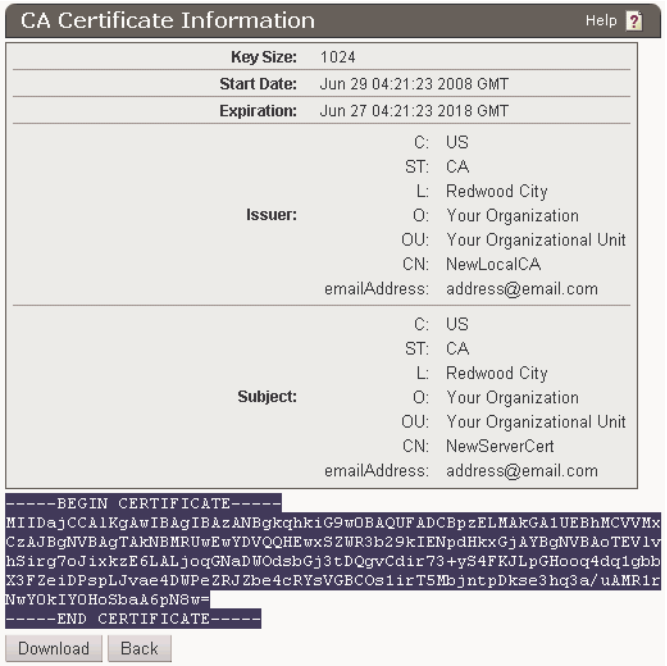
5 Navigate back to the Local Certificate Authority List section (Security, Certificates & CAs, Local CAs). Select your new local CA and click **Sign Request**.

6 Select **Certificate Purpose** *Server* and paste the certificate request into the **Certificate Request** field, as shown below.



7 Click **Sign Request**. This will take you to the CA Certificate Information section.

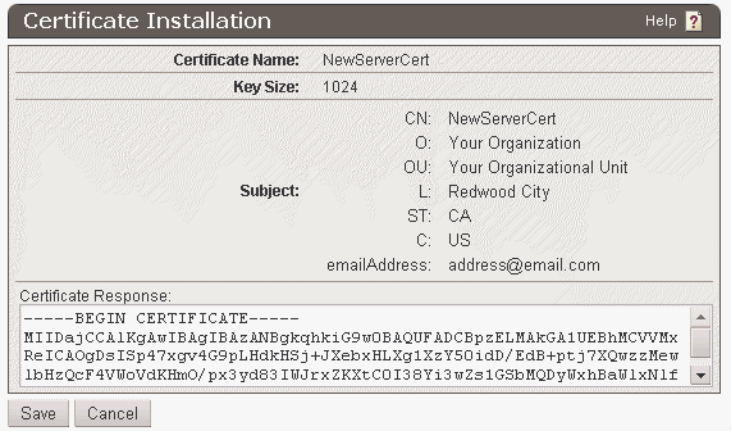
8 Copy the actual certificate (highlighted below). Include the header and footer.



9 Navigate back to the Certificate List section (Security, Certificates & CAs, Certificates). Select your certificate request and click **Properties**.

10 Click **Install Certificate**.

11 Paste the actual certificate, as shown below. Click **Save**.



The Certificate List section will now indicate that NewServerCert is an active certificate.

12 Navigate to the NAE Server Settings section (Device, NAE Server, NAE Server). Click **Edit**.

13 Check **Use SSL** and select your new server certificate in the **Server Certificate** field. Click **Save**.

14 Navigate back to the Local Certificate Authority List section (Security, Certificates & CAs, Local CAs). Select your new CA and click **Download**. Place the CA certificate in a secure directory on your client.

15 Update the following parameters in your IngrianNAE.properties file:

- `Protocol=ssl`
- `CA_File=<path to CA cert>\localca.crt`

Note: Whenever you update the properties file, you must restart the database for the changes to take effect.

You can test your configuration by running the sample.exe application.

You must create an NAE User and an encryption key on the DataSecure appliance. Then use the following command:

```
sample IngrianNAE.properties username password keyname algorithm iv
HelloWorld
```

The DataSecure appliance will return an encrypted value for `HelloWorld`.

For example:

```
sample IngrianNAE.properties user1 qwerty key1 AES/CBC/PKCS5Padding
abcdefghabcdefgh HelloWorld

98 ed 2f 96 be 56 91 4f 20 d5 42 6f 42 e2 a6 ca
```

SSL with Client Certificate Authentication Overview

This SSL implementation requires that both the server and the client produce certificates. Each certificate is signed by a trusted CA known to both the server and the client. Most likely, you will use one CA to sign both certificates. During the SSL handshake, the certificates are exchanged. Both the client and the server use the CA certificate to validate one another's certificate, thus authenticating the other party.

▶ For more information about setting up SSL, see "SSL Overview" on page 121.

To enable client certificate authentication, *you must first successfully configure SSL*. Then, you must make additional configuration changes to the client and server.

Tip: SafeNet, Inc. recommends that you increase security *only after* confirming network connectivity. You should establish an SSL connection before enabling Client Certificate Authentication. Otherwise, an SSL configuration mistake could interfere with your Client Certificate Authentication setup and complicate the troubleshooting process.

To configure the client, you must:

- Create a client certificate.

  This may involve the following steps:
  - Generating a Client Certificate Request with req.exe.
  - Signing a Certificate Request and Downloading the Certificate.

  You can create a certificate request using the req.exe utility or OpenSSL. You can then sign the request with the local CA on the DataSecure appliance. Once signed, the certificate request becomes a valid certificate.

  If you are not using a local CA, consult your CA documentation for instructions on signing requests and exporting certificates.

- Update `IngrianNAE.properties` file as follows:
  - `Cert_File=<location and name of the client certificate>`
  - `Key_File=<location and name of the client's key file>`
  - `Passphrase=<the passphrase used to unlock the client's key file>`

  Note: Whenever you update the properties file, you must restart the database for the changes to take effect.

To configure the server, you must:

- Place a copy of the CA certificate on your server.

  This may involve the following steps:
  - Installing a CA Certificate on the Server.
  - Adding a CA to a Trusted CA List Profile.

- Update the NAE Server Authentication Settings section on the Management Console (Device, NAE Server, NAE Server).

  You'll need to select either *Used for SSL session only* or *Used for SSL session and NAE username* in the **Client Certificate Authentication** field. The profile listed in the **Trusted CA List Profile** field must include the CA used to sign the client certificate. You can update the other fields as needed.

SSL with Client Certificate Authentication Procedures

This section describes the procedures you will follow when configuring SSL with Client Certificate Authentication. It explains the following processes:

- Generating a Client Certificate Request with req.exe
- Signing a Certificate Request and Downloading the Certificate
- Installing a CA Certificate on the Server
- Adding a CA to a Trusted CA List Profile

Generating a Client Certificate Request with req.exe

To generate a client certificate request:

1. Open a command prompt window and navigate to the directory where the Certificate Request Generator utility (req.exe) is installed.
2. Generate an RSA key and a client certificate request using the following command:

```
req -out clientreq -newkey rsa:1024 -keyout clientkey
```

where clientreq is the name of the certificate request being created, and clientkey is the name of the private key associated with the certificate request.

If you are using OpenSSL, use the following command:

```
openssl req -out clientreq -newkey rsa:1024 -keyout clientkey
```

Note: The certificate request and private key will both be created in the working directory by default. You can generate them in another directory by including a location in the request and key names. For example, to create them in the C:\client_certs folder, use the following command:

```
openssl req -out C:\client_certs\clientreq -newkey rsa:1024 -
keyout C:\client_certs\clientkey
```

The key generation process will then request the following data:

- A PEM passphrase to encode the private key.

  The passphrase that encodes the private key is the first passphrase you provide after issuing the command above. This will be the Passphrase parameter in the IngrianNAE.properties file.
- The distinguished name.

  The distinguished name is a series of fields whose values are incorporated into the certificate request. These fields include country name, state or province name, locality name, organization name, organizational unit name, common name, email address, surname, user ID, and IP address.

- For more information about deriving NAE usernames and authenticating client IP addresses, see “Authentication Overview” in the DataSecure Appliance User Guide.

  If you will derive the NAE username from the client certificate, be sure to enter a value in the appropriate field when prompted.

  If you will require client certificates to contain a source IP address, be sure to enter the IP address when prompted.

  - A challenge password.

    This challenge password is NOT used in the DataSecure environment.
  - An optional company name.

Signing a Certificate Request and Downloading the Certificate

This section describes how to sign a certificate request with a local CA and then download the certificate. You must download the certificate *immediately* after it is signed by the CA.

To sign a certificate request with a local CA:

1. Open the certificate request in a text editor.
2. Copy the text of the certificate request. The copied text must include the header (-----BEGIN CERTIFICATE REQUEST-----) and the footer (-----END CERTIFICATE REQUEST-----).
3. Log in to the DataSecure appliance as an administrator with Certificate Authorities access control.
4. Navigate to the Local Certificate Authority List (Security, Certificates & CAs, Local CAs). Select the local CA and click **Sign Request** to access the Sign Certificate Request section.
5. Modify the fields as shown:
   - **Sign with Certificate Authority** - Select the CA that signs the request.
   - **Certificate Purpose** - Select *Client*.
   - **Certificate Duration (days)** - Enter the life span of the certificate.
   - **Certificate Request** - Paste all text from the certificate request, including the header and footer.
6. Click **Sign Request**. This will take you to the CA Certificate Information section.
7. Click the **Download** button to save the certificate on your local machine. You should place the certificate in a secure location and modify access appropriately.

Note: Use the `Cert_File` parameter in the IngrianNAE.properties file to indicate the name and location of the client certificate.

Installing a CA Certificate on the Server

If the client certificate was signed by a non-local CA, you must install the CA certificate on the DataSecure appliance. To install a CA Certificate:

1. Log in to the DataSecure appliance as an administrator with Certificate Authorities access control.
2. Navigate to the Install CA Certificate section on the Certificate Authority Configuration page (Security, Certificates & CAs, Known CAs).
3. Enter the Certificate Name.
4. Paste all text from the certificate in the Certificate field, including the header and footer.
5. Click the **Install** button.

Adding a CA to a Trusted CA List Profile

To add the CA that signed the client certificate to the Trusted CA List Profile:

1. Log in to the DataSecure appliance as an administrator with Certificate Authorities access control.
2. Navigate to the Trusted Certificate Authority List Profiles section on the Certificate and CA Configuration page (Security, Certificates & CAs, Trusted CA Lists).
3. Select the profile to which you want to add the CA.
4. Click the **Properties** button.
5. Click the **Edit** button in the Trusted Certificate Authority List section.
6. Select the CA in the Available CAs field and click the **Add** button. This moves your CA from the Available CAs field to the Trusted CAs field.
7. Click the **Save** button.

Note: To enable SSL with Client Certificate Authority, the Profile containing the CA that signed the client certificate must be selected as the **Trusted CA List Profile** on the NAE Server Authentication Settings section.

SSL with Client Certificate Authentication Walkthrough for DataSecure Clients

This walkthrough assumes the following:

- You have read the SSL with Client Certificate Authentication overview section.
- You have successfully completed the SSL Walkthrough for DataSecure Clients. *You must use the Local CA created in that walkthrough.* The instructions below assume that the client and server certificates were signed by the same local CA.

There are a few different ways that you could configure SSL with Client Certificate Authentication. For example, you can use a CA that does not reside on the DataSecure appliance or you can create a new one. This walkthrough makes such decisions for you. By following these instructions, you will:

- Create a Client Certificate Request using req.exe.
- Create a Client Certificate by signing the Client Certificate Request with the Local CA on the DataSecure appliance.
- Add the Local CA to the Trusted CA List.

Once you have completed and understood this walkthrough, you might decide to alter some of the steps to better fit your organization's policies.

To configure client certificate authentication:

1. Open a command prompt window on the client and navigate to the directory that contains req.exe.
2. Generate an RSA key and a client certificate request using the following command:

   ```
   req -out ssl\clientreq -newkey rsa:1024 -keyout ssl\clientkey
   ```

   where the certificate request, clientreq, and the private key, clientkey, are created in the ssl directory.

   If you are using OpenSSL, use the following command:

   ```
   openssl req -out ssl\clientreq -newkey rsa:1024 -keyout
   ssl\clientkey
   ```

   The key generation process will then request the following data:

   - A PEM passphrase to encode the private key.

     The passphrase that encodes the private key is the first passphrase you provide after issuing the command above. This will be the Passphrase parameter in the IngrianNAE.properties file.

- The distinguished name.

  The distinguished name is a series of fields whose values are incorporated into the certificate request. These fields include country name, state or province name, locality name, organization name, organizational unit name, common name, email address, surname, user ID, and IP address.

▶ For more information about deriving NAE usernames and authenticating client IP addresses, see “Authentication Overview” in the DataSecure Appliance User Guide.

  If you will derive the NAE username from the client certificate, be sure to enter a value in the appropriate field when prompted.

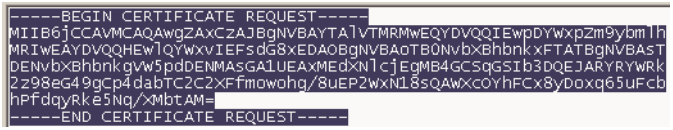
  If you will require client certificates to contain a source IP address, be sure to enter the IP address when prompted.

- A challenge password.

  This challenge password is NOT used in the DataSecure environment.

- An optional company name.

3 Open the client certificate request file and copy the actual request (highlighted below). Include the header and footer.



4 Log in to the Management Console as an administrator with Certificate Authorities access control.

5 Navigate to the Local Certificate Authority List section (Security, Certificates & CAs, Local CAs). Select NewLocalCA and click **Sign Request**. (NewLocalCA is the CA you created in the SSL Walkthrough.)

6 Select **Certificate Purpose** *Client* and paste the certificate request into the **Certificate Request** field, as shown below.

7. Click **Sign Request**. This will take you to the CA Certificate Information section.
8. Click **Download** to download your new client certificate (signed.crt) to your client. Place the certificate in a secure directory on your client.
9. Navigate to the Trusted Certificate Authority List Profiles section (Security, Certificates & CAs, Trusted CA Lists). Select **Profile Name** *Default* and click **Properties**.
10. Click **Edit** in the Trusted Certificate Authority List section.
11. Select *NewLocalCA* in **Available CAs** and click **Add**. Click **Save**.
12. Update the following parameters in the IngrianNAE.properties file:
    - `Cert_File=<path to client cert>\client.crt`
    - `Key_File=<path to client key>\clientkey`
    - `Passphrase=<the passphrase used to unlock the client's key file>`

    Note: Whenever you update the properties file, you must restart the database for the changes to take effect.
13. Return to the Management Console and navigate to the NAE Server Authentication Settings section (Device Management, NAE Server) and enter the following values:
    - **Client Certificate Authentication**: Used for SSL Session only
    - **Trusted CA List Profile**: Default

    Important! The CA used to sign the client certificate must be a member of the **Trusted CA List Profile**.

You can test your configuration by running the sample.exe application.

You must create an NAE User and an encryption key on the DataSecure appliance. Then use the following command:

```
sample IngrianNAE.properties username password keyname algorithm iv
HelloWorld
```

The DataSecure appliance will return an encrypted value for `HelloWorld`.

For example:

```
sample IngrianNAE.properties user1 qwerty key1 AES/CBC/PKCS5Padding
abcdefghabcdefgh HelloWorld

98 ed 2f 96 be 56 91 4f 20 d5 42 6f 42 e2 a6 ca
```

APPENDIX A

DATABASE OBJECTS

The SafeNet Database Connector for Oracle includes database objects that enable you to access some features of the NAE Server directly from the Oracle Enterprise Manager. These objects include the tables, views, sequences, procedures and functions detailed in this document. This chapter covers the following topics:

Tables and Views

ING_AUTHORIZED_USER

This table is used to define all users authorized to perform encrypt and decrypt operations.

Column Name	Data Type	Description
ATHRZ_USER_ID	NUMBER	NOT NULL
DB_USER_NM	VARCHAR2(30)	NOT NULL
ENCRPT_USER_NM	VARCHAR2(128)	NOT NULL
ENCRPT_ACCESS_CD	VARCHAR2(128)	NOT NULL
DGTL_SGNTR_NM	VARCHAR2(256)	NOT NULL
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_AUTH_USER

This view allows a user to select only his or her data from the ING_AUTHORIZED_USER table. This view contains the following columns.

- ATHRZ_USER_ID
- DB_USER_NM
- ENCRPT_USER_NM
- ENCRPT_ACCESS_CD
- DGTL_SGNTR_NM
- CREATE_DT
- CREATE_USER_NM
- LAST_MDFY_DT
- LAST_MDFY_USER_NM

ING_BULK_LOAD_CONFIG

This table is used to specify a static parameter set for a named bulk encryption process. There is one entry on this table for each Bulk Loader job you define.

Column Name	Data Type	Description
BULK_JOB_NM	VARCHAR2(30)	NOT NULL
JOB_TYPE_CD	VARCHAR2(12)	NOT NULL
ENCRPT_TBL_ID	NUMBER	NOT NULL
INPUT_TBL_NM	VARCHAR2(30)	NOT NULL
OUTPUT_TBL_NM	VARCHAR2(30)	NOT NULL
OUTPUT_TBL_TBSP_NM	VARCHAR2(30)	NOT NULL
BATCH_SIZE_QTY	NUMBER	NOT NULL
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_BULK_LOAD_RUN

This table is used to specify the actual parameter set supplied for a specific bulk job run request. This table has the parameter set that can be used for resuming a specific job run. Each record in this table has a corresponding record in the ING_JOB table.

For each time the Bulk Loader is run, there is one entry in this table.

Column Name	Data Type	Description
JOB_ID	NUMBER	NOT NULL
BULK_JOB_NM	VARCHAR2(30)	NOT NULL
INPUT_TBL_NM	VARCHAR2(30)	NOT NULL
OUTPUT_TBL_NM	VARCHAR2(30)	NOT NULL
OUTPUT_TBL_TBSP_NM	VARCHAR2(30)	NOT NULL
ACTV_PRCS_ID	VARCHAR2(128)	
BULK_LOAD_EXIT_STATUS_ID	NUMBER	Exit status or return code
BULK_LOAD_EXIT_STATUS_TXT	VARCHAR2(255)	
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_COLUMN_DEFAULT

This table is used to specify details about columns that have default values in their insert triggers.

Column Name	Data Type	Description
DFLT_CLMN_ID	NUMBER	NOT NULL
ENCRPT_TBL_ID	NUMBER	NOT NULL
CLMN_NM	VARCHAR2(128)	NOT NULL
CLMN_NULL_DFLT_FL	CHAR(1)	
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_ENCRYPTED_COLUMN

This table is used to specify details about each column that is encrypted.

Column Name	Data Type	Description
ENCRPT_CLMN_ID	NUMBER	NOT NULL
ENCRPT_TBL_ID	NUMBER	NOT NULL
ORGNL_CLMN_NM	VARCHAR2(30)	NOT NULL
NEW_CLMN_NM	VARCHAR2(30)	NOT NULL
ENCRPT_KEY_NM	VARCHAR2(128)	NOT NULL
ENCRPT_ALGRTHM_NM	VARCHAR2(128)	NOT NULL

Column Name	Data Type	Description *(continued)*
ENCRPT_MODE_NM	VARCHAR2(30)	
ENCRPT_PAD_NM	VARCHAR2(30)	
ENCRPT_IV_TYPE_CD	CHAR(1)	
ENCRPT_IV_NM	RAW(30)	
ENCRPT_IV_CLMN_NM	VARCHAR2(30)	
ENCRPT_NULL_RPLCMNT_NM	VARCHAR2(30)	
ERR_RPLCMNT_ID	NUMBER	
CLMN_ORGNL_DATA_TYPE_NM	VARCHAR2(30)	NOT NULL
CLMN_ORGNL_LEN_QTY	NUMBER	NOT NULL
CLMN_ORGNL_PRCSN_QTY	NUMBER	
CLMN_ORGNL_SCALE_QTY	NUMBER	
CLMN_NULL_FL	CHAR(1)	NOT NULL
CLMN_NEW_DATA_TYPE_NM	VARCHAR2(30)	
CLMN_NEW_LEN_QTY	NUMBER	
CLMN_DATA_MGRT_FL	CHAR(1)	NOT NULL
ENCRPT_FL	CHAR(1)	
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	
DMN_IDX_TBLSPC_NM	VARCHAR2(30)	
DMN_REG_IDX_NM	VARCHAR2(30)	
DMN_IDX_NM	VARCHAR2(30)	
DMN_IDX_CREATE_FL	VARCHAR2(1)	
DMN_IDX_FL	VARCHAR2(1)	

ING_ENCRYPTED_TABLE

This table is used to specify which tables contain encrypted columns.

Column Name	Data Type	Description
ENCRPT_TBL_ID	NUMBER	NOT NULL
DB_NM	VARCHAR2(30)	
TBL_OWNR_NM	VARCHAR2(30)	NOT NULL
ORGNL_TBL_NM	VARCHAR2(30)	NOT NULL
NEW_TBL_NM	VARCHAR2(30)	NOT NULL
ADD_ID_CLMN_FL	CHAR(1)	NOT NULL

Column Name	Data Type	Description *(continued)*
ID_CLMN_NM	VARCHAR2(30)	NOT NULL
ID_CLMN_DATA_TYPE_NM	VARCHAR2(30)	NOT NULL
ID_CLMN_LEN_QTY	NUMBER	NOT NULL
ID_CLMN_PRCSN_QTY	NUMBER	
ID_CLMN_SCALE_QTY	NUMBER	
TMPRY_TBL_NM	VARCHAR2(30)	
TMPRY_TBL_TBLSPC_NM	VARCHAR2(30)	
INTRM_VW_NM	VARCHAR2(30)	
INSRT_TRG_NM	VARCHAR2(30)	
UPDT_TRG_NM	VARCHAR2(30)	
PRIMARY_KEY_COL_NMS	VARCHAR2(2000)	
PRIMARY_KEY_NM	VARCHAR2(30)	
TMPRY_TBL_CREATE_FL	CHAR(1)	NOT NULL
VW_CREATE_FL	CHAR(1)	NOT NULL
SERVER_VERSION	VARCHAR2(20)	
SERVER_MIGRATION_VERSION	VARCHAR2(20)	
UDF_VERSION	VARCHAR2(20)	
SEQ_NM	VARCHAR2(30)	
OLD_DATA_EXIST_FL	CHAR(1)	
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_ERROR_LOG

This table is used to log errors during column specific encrypt and decrypt operations.

Column Name	Data Type	Description
ERR_LOG_ID	NUMBER	NOT NULL
ENCRPT_CLMN_ID	NUMBER	
DB_USER_NM	VARCHAR2(30)	NOT NULL
RQST_TYPE_CD	CHAR(1)	NOT NULL
ERR_DT	DATE	NOT NULL
ERR_DESC	VARCHAR2(1024)	
ERR_MSG_TXT	VARCHAR2(1024)	
ERR_STACKTRACE_TXT	VARCHAR2(2000)	

Column Name	Data Type	Description *(continued)*
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_JOB

This table is used to describe the details of a data migration operation. For each time the Bulk Loader is run, there is one entry in this table.

Column Name	Data Type	Description
JOB_ID	NUMBER	NOT NULL
ENCRPT_TBL_ID	NUMBER	NOT NULL
JOB_TYPE_CD	VARCHAR2(12)	NOT NULL
JOB_STATUS_CD	VARCHAR2(12)	NOT NULL
JOB_START_DT	DATE	
JOB_END_DT	DATE	
TOTAL_ROWS_QTY	NUMBER	
ROWS_MDFY_QTY	NUMBER	
BATCH_SIZE_QTY	NUMBER	
JOB_CNCLE_FL	CHAR(1)	
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_JOB_BATCH

This table is used to record the status of a migration job. For each Bulk Loader job that is run, there is one entry in this table for each batch that is processed, and there is one entry for pre–processing information, such as input and output table validation.

For example, if you have 100,000 rows to process in a batch job, and the batch size is 10,000, then there are 11 entries in this table for the job: 10 entries for the 10 batches, and 1 entry for the pre–processing information.

Column Name	Data Type	Description
JOB_ID	NUMBER	NOT NULL
BATCH_SEQ	NUMBER	NOT NULL
JOB_STATUS_CD	VARCHAR2(12)	NOT NULL

Column Name	Data Type	Description *(continued)*
BATCH_START_DT	DATE	
BATCH_END_DT	DATE	
BATCH_SIZE_QTY	NUMBER	
SQL_ERR_ID	NUMBER	
SQL_ERR_STATE_TXT	VARCHAR2(200)	
ERR_CLASS_NM	VARCHAR2(200)	
ERR_MSG_TXT	VARCHAR2(1024)	
ERR_STACKTRACE_TXT	VARCHAR2(2000)	
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_JOB_COLUMN

This table is used to describe column-level encryption details. Primarily used during key rotation.

Column Name	Data Type	Description
JOB_ID	NUMBER	NOT NULL
ENCRPT_CLMN_ID	NUMBER	NOT NULL
ENCRPT_KEY_NM	VARCHAR2(128)	
ENCRPT_IV_NM	RAW(30)	
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_JOB_LOG

This table is used to log any output from the job run. This table contains the complete job log information for every run of every Bulk Loader job. This table can grow to be quite large; for example, one run of one job consisting of 10 batches could yield 300 rows of data or more in this table.

Column Name	Data Type	Description
JOB_ID	NUMBER	NOT NULL
LOG_SEQ	NUMBER	NOT NULL
BATCH_SEQ	NUMBER	
LOG_TXT	VARCHAR2(2000)	NOT NULL

Column Name	Data Type	Description *(continued)*
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_JOB_STATUS

This table is a reference table for valid values of Job Type.

Column Name	Data Type	Description
JOB_STATUS_CD	VARCHAR2(12)	NOT NULL
JOB_STATUS_NM	VARCHAR2(128)	
JOB_STATUS_DESC	VARCHAR2(1024)	
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_JOB_TYPE

This table is a reference table for valid values of Job Type.

Column Name	Data Type	Description
JOB_TYPE_CD	VARCHAR2(12)	NOT NULL
JOB_TYPE_NM	VARCHAR2(128)	
JOB_TYPE_DESC	VARCHAR2(1024)	
CREATE_DT	DATE	
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

ING_PROPERTY

This table is used to store system properties.

Column Name	Data Type	Description
PRPTY_KEY_NM	VARCHAR2(100)	NOT NULL
PRPTY_VAL_NM	VARCHAR2(100)	
CREATE_DT	DATE	

Column Name	Data Type	Description *(continued)*
CREATE_USER_NM	VARCHAR2(30)	
LAST_MDFY_DT	DATE	
LAST_MDFY_USER_NM	VARCHAR2(30)	

Sequences

- ING_AUTHORIZED_USER_SEQ – Used to generate the primary keys for the ING_AUTHORIZED_USER table.
- ING_COLUMN_DEFAULT_SEQ – Used to generate the primary keys for the ING_COLUMN_DEFAULT table.
- ING_ENCRYPTED_COLUMN_SEQ – Used to generate the primary keys for the ING_ENCRYPTED_COLUMN table.
- ING_ENCRYPTED_TABLE_SEQ – Used to generate the primary keys for the ING_ENCRYPTED_TABLE table.
- ING_ERROR_LOG_SEQ – Used to generate the primary keys for the ING_ERROR_LOG table.
- ING_JOB_SEQ – Used to generate the primary keys for the ING_JOB table.

Procedures

Our procedures enable you to interface with the NAE Server directly from the Oracle database by accessing the procedures provided in the SafeNet Database Connector. You can use these procedures to map an Oracle database username to an DataSecure username, execute a bulk load job, cancel a bulk load job, and add or update a bulk load job definition.

Some of these procedures are contained in the ING_BULK_LOAD_PKG and ING_JOB_CONTROL packages. Others stand alone in the database.

Set DataSecure User Info (set_ingrian_user_info)

This stored procedure is used to add a mapping between an Oracle database user and a DataSecure user. This procedure is executed by the DBTools user interface when mapping database users to DataSecure users. All of the data stored by this procedure is encrypted and each row in the table is digitally signed to ensure the security of the data.

Parameter Name	Data Type	In/Out	Description
dbUserName	varchar2	In	The Oracle username.
userName	varchar2	In	The DataSecure username
userPassword	varchar2	In	The NAE user's password.

Example

```
EXEC    ingrian.set_ingrian_user_info
      ('dba',
      'NAEuser',
      'NAEPassword');
----------------------------------------
PL/SQL procedure successfully completed.
```

Bulk Load (ing_bulk_load)

This procedure initiates the bulk load process. It is part of the bulk load package (ING_BULK_LOAD_PKG).

Important! All input parameters for this procedure are case sensitive, and they must be specified in the order they appear below.

Parameter Name	Data Type	In/Out	Description
jobName	varchar2	In	Name of the job to be run. This job name must have been defined using the bulk load setup procedure (ing_bulk_load_setup).
resume	char	In	A Y/N flag that indicates where the job should start from if the previous job run did not complete successfully. Set the flag to **Y** to restart the job at the point of failure. Set the flag to **N** to completely re-run and process all records.
errorCode	number	Out	An integer that indicates the return status of the procedure execution. A value of **0** indicates a successful job run. Any non-zero value indicates a failure.
errorMsg	varchar2	Out	A textual description of the status code.

Example

```
DECLARE retStat NUMBER; retTxt VARCHAR2(255);

BEGIN   ing_bulk_load.bulk_load
      ('MyBulkJob',
      'Y',
      retStat,
      retTxt);

END;
/
```

Cancel Bulk Load Job (ing_bulk_load_cancel)

This stored procedure cancels a Bulk Loader job. Set the cancel flag for the specified job name. This causes the job to stop processing after the batch currently being processed has been completed and committed. To restart the job at the point it was stopped, execute the master.dbo.ing_bulk_load stored procedure, setting the resumeFlag to **Y**.

It is part of the bulk load package (ING_BULK_LOAD_PKG).

Important! The input parameter for this procedure is case-sensitive.

Parameter Name	Data Type	In/Out	Description
jobName	varchar2	In	Name of the job to be cancelled. This job name must have been previously defined using the stored procedure master.dbo.ing_bulk_load_config.
errorCode	number	Out	An integer that indicates the return status of the procedure execution. A value of **0** indicates a successful job run. Any non-zero value indicates a failure.
errorMsg	varchar2	Out	A textual description of the status code.

Example

```
SET     Serveroutput on
EXEC    ing_bulk_load_pkg.ing_bulk_load_cancel('MyBulkJob');
```

Describe Bulk Load Job (ing_bulk_load_describe)

This stored procedure retrieves the job data from the `ING_BULK _LOAD_CONFIG` table. Only a job's owner can access this information. Other users attempting to execute this procedure receive an authorization error.

This procedure is part of the bulk load package (`ING_BULK_LOAD_PKG`).

Important! The input parameter for this procedure is case-sensitive.

Parameter Name	Data Type	In/Out	Description
jobName	varchar2	In	Name of the job to be described. This job must have been previously defined using the stored procedure ing_bulk_load_config.

Example

```
SET     Serveroutput on
EXEC    dbms_output.enable(1000000);
EXEC    ing_bulk_load_pkg.ing_bulk_load_describe('MyBulkJob');
```

Set Up Bulk Load Job (ing_bulk_load_setup)

This stored procedure defines a Bulk Loader job. Add a new bulk load job definition to the ing_bulk_load_config table, or update an existing bulk job definition. The job definition defines all parameters required to execute the bulk load procedure (ing_bulk_load).

It is part of the bulk load package (ING_BULK_LOAD_PKG).

Important! All input parameters for this procedure are case sensitive, and they must be specified in the order they appear below.

Parameter Name	Data Type	In/Out	Description
jobName	varchar2	In	Name of the bulk load job. The value must contain between 1 and 30 characters. If the job name does not exist in the ING_BULK_LOAD_CONFIG table, a new job definition is added. If the job name does exist, the job definition is updated with the new parameters.
jobType	varchar	In	Type of operation. Determines if the job encrypts cleartext data or decrypts ciphertext. Valid values are **encrypt** or **decrypt**.
schemaName	varchar2	In	Name of the schema on which the table resides.
encryptedTableName	varchar2	In	The base table. This table has already been encrypted by the Data Migration Utility and shares column names and types with the input table. The Bulk Loader applies the same keys and algorithms used to encrypt the base table columns to the corresponding columns in the input table.
inputTableName	varchar2	In	The input table. This is the table or view to be encrypted or decrypted. The input table must reside on the database named in the databaseName parameter.
outputTableName	varchar2	In	The output table. The results of the bulk load process are stored here. If the table does not exist prior to processing, the Bulk Loader creates it using the table space, database and owner parameters supplied here.
outputTableSpaceName	varchar2	In	The table space in which the output table resides. If the output file does not exist in the database described in the outputTableName parameter, the Bulk Loader creates the output file in this table space. This parameter is ignored if Bulk Loader does not have to create the output table.
batchSize	number	In	The number of rows to be processed between commits. Valid values are between 1 and 100,000.
errorCode	number	Out	An integer that indicates the return status of the procedure. An exitStatusCode of 0 indicates a successful job run. An exitStatusCode < 0 indicates failure.
errorMsg	varchar2	Out	A textual description of the status code.

Example

```
DECLARE retStat NUMBER;
      retTxt VARCHAR2 (255);

BEGIN   ing_bulk_load_pkg.ing_bulk_load
      ('MyBulkJob',
      'encrypt',
      'dba',
      'EMPLOYEES',
      'EMPLOYEES_IN',
      'EMPLOYEES_OUT',
      'USERS',
      1000,
      retStat,
      retTxt);

END;
/
      ================================
      16:40:33   STARTING PROCEDURE initialize
      16:40:33   Debug Level is 1
      16:40:33   ENDING PROCEDURE initialize
      16:40:33   STARTING PROCEDURE add_bulk_load_config
      16:40:33   STARTING PROCEDURE get_enc_table_id
      16:40:33   ENDING PROCEDURE get_enc_table_id
      16:40:33   STARTING PROCEDURE create_bulk_load_config
      16:40:33   ENDING PROCEDURE create_bulk_load_config
      16:40:33   ENDING PROCEDURE add_bulk_load_config

      PL/SQL procedure successfully completed.
```

Functions

Our user defined functions (UDFs) enable you to interface with the NAE Server directly by accessing the procedures in the SafeNet Database Connector. You can use these UDFs to get session status, retrieve information from the `IngrianNAE.properties` file, generate random binary values, convert data types, and encrypt and decrypt data.

Encryption and Decryption UDFs

The encryption and decryption UDFs enable you to access the cryptographic functionality of the NAE Server from the database. There are two types of cryptographic UDFs: command line and name.

The command line UDFs take the key name, algorithm name, and IV as parameters and apply these values to the input data. The names of these UDFs include the prefix ‘cl’. The input parameters are detailed in the table below.

The name UDFs take the schema, table name, column name, and IV as parameters and use these values to recall the encryption metadata (key name, algorithm, and IV). That metadata is then applied to the input data. The names of these UDFs include the prefix ‘nm’. The input parameters are detailed below.

The encryption and decryption UDFs take the following parameters:

Parameter Name	Data Type	Description
inputData	*varies by UDF*	The value to be encrypted or decrypted. The data type is abbreviated in the UDF name. This parameter is used for both the command line and the name UDFs.
keyName	varchar2	The name of the key to use to encrypt or decrypt the data. This key must exist on the NAE Server, and the requester must have permission to use the key for the operation requested. This parameter is used for the command line UDFs.
algorithmName	varchar2	A string representing the algorithm, the mode in which the key is used (CBC or ECB) and what kind of padding to use. This parameter is used for the command line UDFs.
schemaName	varchar2	The schema on which the table resides. This parameter is used for the name UDFs.
tableName	varchar2	The table on which the column resides. This parameter is used for the name UDFs.
columnName	varchar2	The column to be operated on. This parameter is used for the name UDFs.
iv	raw	The value to use as the Initialization Vector (IV) for data being encrypted or decrypted with a symmetric key (DES, DESede or AES algorithms) in CBC mode. For other algorithms or symmetric keys used in ECB mode, this value is ignored. The IV you specify for operations that use DES or DESede keys must be eight bytes; for AES keys, the IV must be sixteen bytes. If you are going to specify an IV here, the IV must be base 16 encoded. This parameter is used for the both the command line and the name UDFs.

The following table lists the UDFs by data type:

Data Type	Function	Action
CHAR	ing_e_chr_cl430	encrypt
	ing_e_chr_nm430	encrypt
	ing_d_chr_cl430	decrypt
	ing_d_chr_nm430	decrypt
DECIMAL	ing_e_dec_cl430	encrypt
	ing_e_dec_nm430	encrypt
	ing_d_dec_cl430	decrypt
	ing_d_dec_nm430	decrypt
DATETIME	ing_e_dtm_cl430	encrypt
	ing_e_dtm_nm430	encrypt
	ing_d_dtm_cl430	decrypt
	ing_d_dtm_nm430	decrypt
NCHAR	ing_e_ncr_cl430	encrypt
	ing_e_ncr_nm430	encrypt
	ing_d_ncr_cl430	decrypt
	ing_d_ncr_nm430	decrypt
NVARCHAR	ing_e_nvc_cl430	encrypt
	ing_e_nvc_nm430	encrypt
	ing_d_nvc_cl430	decrypt
	ing_d_nvc_nm430	decrypt
VARCHAR	ing_e_vrc_cl430	encrypt
	ing_e_vrc_nm430	encrypt
	ing_d_vrc_cl430	decrypt
	ing_d_vrc_nm430	decrypt

Note: A fresh install has only _430 UDFs. An upgrade from 4.2 to 4.8 will contain both _430 UDFs and _420 UDFs for backwards compatibility.

You would not normally call these UDFs directly; they are integrated into the data migration and Bulk Loader functionality. Examples of direct calls to name and command line UDFs are shown below.

Direct Call to ing_e_vrc_nm430

```
SELECT   ingrian.ing_e_vrc_nm430
      ('yourVarchar',               /* inputData */
      'yourSchema',                 /* schemaName */
      'yourTable',                  /* tableName  */
      'yourColumn',                 /* columnName  */
      '24A40F6DFB05F379630AC0533E7F6357') /* iv */

AS       ENCRYPTED_VALUE

FROM     dual;
      ---------------------------------
      ENCRYPTED_VALUE
      9F71F06D9F1E123B68D910F0EC34E8C5
```

Direct Call to ing_d_vrc_nm430

```
SELECT   ingrian.ing_d_vrc_nm430
      ('9F71F06D9F1E123B68D910F0EC34E8C5', /* inputData*/
      'yourSchema',                 /* schemaName*/
      'yourTable',                  /* tableName*/
      'yourColumn',                 /* columnName*/
      '24A40F6DFB05F379630AC0533E7F6357') /* iv */

AS       DECRYPTED_VALUE

FROM     dual;
      ------------------
      DECRYPTED_VALUE
      yourVarchar
```

Direct Call to ing_e_vrc_cl430

```
SELECT   ingrian.ing_e_vrc_cl430
      ('yourVarchar',               /* inputData*/
      'yourAESKey',                 /* keyName  */
      'AES/CBC/PKCS5Padding',       /* algorithmName*/
      '24A40F6DFB05F379630AC0533E7F6357') /* iv */

AS       ENCRYPTED_VALUE

FROM     dual;
      ---------------------------------
      ENCRYPTED_VALUE
      9F71F06D9F1E123B68D910F0EC34E8C5
```

Direct Call to ing_d_vrc_cl430

```
SELECT   ingrian.ing_d_vrc_cl430
      ('9F71F06D9F1E123B68D910F0EC34E8C5',/* inputData*/
      'yourAESKey',                 /* keyName  */
      'AES/CBC/PKCS5Padding',       /* algorithmName*/
      '24A40F6DFB05F379630AC0533E7F6357') /* iv */

AS       DECRYPTED_VALUE

FROM     dual;
      ------------------
      DECRYPTED_VALUE
      yourVarchar
```

Data Type Conversion UDFs

Our encryption and decryption UDFs automatically convert the input data to RAW using the conversion UDFs shown below. You would not normally call these UDFs directly, they are integrated into the data migration and Bulk Loader functionality.

The following table describes the process of converting various data types before the encryption UDFs are applied.

Data Type	Conversion Method
VARCHAR2, CHAR, NVARCHAR2, and NCHAR	Converted directly to RAW.
NUMBER	Converted to a string and then converted to RAW.
DATE	Converted to a standard 31-byte string format, then converted to RAW.

The following table lists the conversion UDFs.

Name	Description
ingnumbertoraw	Converts a NUMBER value to a RAW value.
ingrawtonumber	Converts a RAW value to a NUMBER value.
ingdatetoraw	Converts a DATE value to a RAW value.
ingrawtodate	Converts a RAW value to a DATE value.
ingstringtoraw	Converts a string to a RAW value.
ingrawtostring	Converts a RAW value to a string value.
ingnstringtoraw	Converts an nstring value to a RAW value.
ingrawtonstring	Converts a RAW value to an nstring value.

Get Session Status (get_session_status)

This stored procedure, available only to the `INGRIAN` user, is used to get the current status of a given Oracle session.

Parameter Name	Data Type	In/Out	Description
pi_nSessionID	number	In	The audsid of the requested session.

Function Output	Data Type	In/Out	Description
Return Value	varchar2	Out	The status of the requested Oracle session from the sys.v_$session view.

Example

```
SELECT  get_session_status(2591)

FROM    dual;
      ------------------------
      GET_SESSION_STATUS(2591)
      ACTIVE
```

Get Ingrian Properties (inggetproperty)

This stored procedure is used to return the value associated with some of parameters in the `IngrianNAE.properties` file. You can obtain values for all parameters except the following restricted properties:

- Client_Cert_Passphrase
- Client_Cert_Alias
- CA_File
- Cert_File
- Key_File
- Passphrase

Parameter Name	Data Type	In/Out	Description
propertyName	varchar2	In	The name of the parameter. Parameter names are case sensitive.

Function Output	Data Type	In/Out	Description
Return Value	varchar2	Out	The value associated with the requested parameter.

Example

```
SELECT   inggetproperty
      ('NAE_IP')

FROM     dual;

      ------------------------
      INGGETPROPERTY ('NAE_IP')
      192.168.1.100
```

Generate Random (ing_gn_rndm)

This stored procedure is used to generate a random binary value of the specified number of bytes. This procedure is used mainly to generate random IVs. You can request a number of bytes between 1 and 2000.

Parameter Name	Data Type	In/Out	Description
numBytes	number	In	The byte length of the binary value.

Function Output	Data Type	In/Out	Description
Return Value	raw	Out	The resulting number.

Example

```
SELECT  ing_gn_rndm (8)

FROM    dual;

      ------------------
      ING_GN_RNDM(8)
      F3E3B728F8B5AFA7
```

APPENDIX B

Troubleshooting

This chapter contains examples of some of the problems you might encounter when deploying the SafeNet Database Connector for Oracle and how you might be able to work around these issues. The problems and solutions listed here are organized by error message. This appendix contains the following topics:

unable to open RPC connection

The most likely cause for this error is that the Oracle external stored procedure listener is not started. Start the external procedure listener using the command below and then try the Bulk Loader again. The SQL statement used to define a job takes the following form:

```
lsnrctl start <extproc_listener_name>
```

invalid DLL path

This error message is the likely result of a misconfigured Oracle external stored procedure listener. To fix this, you must modify the listener.ora file, which is located here: `$ORACLE_HOME/network/admin`. Open listener.ora in a text editor, and scroll down until you find the line below:

```
(PROGRAM = extproc)
```

Add the line below immediately under the line above:

```
(ENVS="EXTPROC_DLLS=ANY")
```

Now restart the listener and retry the Bulk Loader.

unable to find library

There are several potential causes for this error, some of which are described below:

- Missing Bulk Loader or supporting libraries (in $ORACLE_HOME/lib/ingrian).
- Incorrect permissions on the Bulk Loader or supporting libraries.
- Invalid configuration of Bulk Loader library in the database. You can check this by running the following command in SQL*PLUS:

  ```
  select * from user_libraries
  ```

- There should be one entry in the result set, and this entry's "Path" column should contain `$ORACLE_HOME/lib/ingrian/libingbatchproc.so`

This error case requires attention from SafeNet. Should you encounter this error, please contact Customer Support and provide the output of the commands below:

```
file $ORACLE_HOME/lib/ingrian/*
file $ORACLE_HOME/bin/oracle
uname -a
file /unix
```

no such file or directory

There are several potential causes for this error, some of which are described below:

- Missing Bulk Loader or supporting libraries (in $ORACLE_HOME/lib/ingrian).
- Incorrect permissions on the Bulk Loader or supporting libraries.
- Invalid configuration of Bulk Loader library in the database. You can check this by running the following command in SQL*PLUS:

  ```
  select * from user_libraries
  ```

- There should be one entry in the result set, and this entry's "Path" column should contain `$ORACLE_HOME/lib/ingrian/libingbatchproc.so`

This error case requires attention from SafeNet. Should you encounter this error, please contact Customer Support and provide the output of the commands below:

```
file $ORACLE_HOME/lib/ingrian/*
file $ORACLE_HOME/bin/oracle
uname -a
file /unix
```

file in path does not exist

There are several potential causes for this error, some of which are described below:

- Missing Bulk Loader or supporting libraries (in $ORACLE_HOME/lib/ingrian).
- Incorrect permissions on the Bulk Loader or supporting libraries.
- Invalid configuration of Bulk Loader library in the database. You can check this by running the following command in SQL*PLUS:

  ```
  select * from user_libraries
  ```

- There should be one entry in the result set, and this entry's "Path" column should contain `$ORACLE_HOME/lib/ingrian/libingbatchproc.so`

ld: unresolved external ...

This error case requires attention from SafeNet. Should you encounter this error, please contact Customer Support and provide the output of the commands below:

```
file $ORACLE_HOME/lib/ingrian/*
file $ORACLE_HOME/bin/oracle
uname -a
file /unix
```

ld: unable to load library ... - wrong/unsupported format

Check to see if `$ORACLE_HOME/lib` or `$ORACLE_HOME/lib32` is included in your ld path environment variable (`LD_LIBRARY_PATH` on Linux/Solaris; `SHLIB_PATH` on HPUX, `LIBPATH` on AIX). If one of these directories appears in the ld path environment variable, then this error is caused by using the wrong Bulk Loader or supporting libraries for your platform/database architecture. Make sure that you downloaded the correct version of the SafeNet Database Connector. If you installed the wrong Bulk Loader, report this as a problem as it requires further diagnostics.

Run the following commands and submit their output with the problem report:

```
file $ORACLE_HOME/lib/ingrian/*
file $ORACLE_HOME/bin/oracle
uname -a
file /unix
```

lost RPC connection to …

Check your IngrianNAE.properties file to make sure that the value of the NAE_IP parameter matches the IP address of the NAE Server you are expecting to connect to. If the NAE_IP parameter is incorrect, enter the correct value and save your changes. Reload the IngrianNAE.properties file by issuing the command below:

```
loadProperties <ing_pwd>
```

You should also ensure that the Oracle user has read and write permissions to the temporary directory (`/tmp` on most systems, but can be changed via `$TMP` environment variable). Now run the Bulk Loader again.

If it turns out that the `NAE_IP` parameter is correct and the Oracle user has the appropriate read and write permissions, and you are still seeing the error, you should refer the issue to SafeNet for investigation. Before you do, though, it is important that you capture data that is written to the log when you run the Bulk Loader. Set the `Log_Level` parameter to HIGH in the `IngrianNAE.properties` and reload the `IngrianNAE.properties` file with the command below.

```
loadProperties <ing_pwd>
```

Now run the Bulk Loader again. Submit the following log file to SafeNet as part of the service request: `/tmp/icsp.log`.

network error during callback from external procedure

This error condition is caused by an internal error in the Bulk Loader library. Open `IngrianNAE.properties` in a text editor and set the `Log_Level` parameter to HIGH. Now reload the `IngrianNAE.properties` file by issuing the command below:

```
loadProperties <ing_pwd>
```

Now run the Bulk Loader again. At this point, SafeNet needs to review the issue. Please submit the following log file to SafeNet as part of the service request: `/tmp/icsp.log`.

unable to find library 'libstdc++.sl.5'

The full error message is:

```
13:35:43   ERROR -404 - Internal Error. Error code: bulk_load;
Component: ORA-06520: PL/SQL: Error loading external library
ORA-06522: Unable to find library 'libstdc++.sl.5'.; Error Message:
```

If a library required by the Bulk Loader cannot be found, check the directory `$ORACLE_HOME/lib/` and make sure the required shared library exists.

Any shared libraries installed by SafeNet should be in the directory `$ORACLE_HOME/lib/ingrian`, and be referenced by links in the `$ORACLE_HOME/lib/` directory.

Verify that all of the files referenced by these symbolic links exist in the `$ORACLE_HOME/lib/ingrian` directory:

```
$ ls -l | grep ingrian
lrwxr-xr-x   1 oracle     dba             55 Apr 20 18:39
libstdc++.sl.5 ->
/export/oracle/product/9.2.0/lib/ingrian/libstdc++.sl.5
```

Also, be sure that every shared library in the `$ORACLE_HOME/lib/ingrian` directory has a corresponding symbolic link in the `$ORACLE_HOME/lib` directory, except `libingbatchproc.so`, which does not need one. The number of shared libraries in the `$ORACLE_HOME/lib/ingrian` directory varies by platform, so there are at least one and as many as four.

Next, verify that the shared library path environment variable contains a reference to the `$ORACLE_HOME/lib` directory. The environment variable varies by OS:

- Solaris, Linux – `$LD_LIBRARY_PATH`
- HPUX – `$SHLIB_PATH`
- AIX – `$LIBPATH`

If all of these items have been verified, and the problem is still occurring, modify the listener.ora file – generally found at `$ORACLE_HOME/network/admin` – to include the line below in the definition of the External stored procedure listener:

```
(ENVS="EXTPROC_DLLS=ANY")
```

The new entry now looks similar to this:

```
SID_LIST_EXTPROC_LISTENER =
     (SID_LIST =
       (SID_DESC =
         (SID_NAME = PLSExtProc)
        (ORACLE_HOME = /opt/oracle/product/9.2.0)
         (PROGRAM = extproc)
         (ENVS=”EXTPROC_DLLS=ANY”)
       )
     )
```

After making this change, restart the affected listener, and try running the Bulk Loader again.

error loading external library

When running the Bulk Loader on Oracle, the job fails without processing any data with the following error:

```
ERROR -404 - Internal Error. Error code: bulk_load; Component:
ORA-06520: PL/SQL: Error loading external library
ORA-06522: Exec format error;
Error Message:
```

The “Exec format error” indicates a file version incompatibility. Verify that the shared library path is set correctly for the Bulk Loader file being used. If Oracle is 64 bit and the Bulk Loader shared library is 64 bit, then the shared library path should reference the 64 bit version of the Oracle libraries.

If 64 bit Oracle and 64 bit Bulk Loader are used, then the paths below should reference $ORACLE_HOME/lib, not $ORACLE_HOME/lib32.

- Solaris, Linux – `$LD_LIBRARY_PATH`
- HPUX – `$SHLIB_PATH`
- AIX – `$LIBPATH`

To see if a file is 32 or 64 bit, run the command:

```
file <filename>
```

For example:

```
$ file libingbatchproc.so
libingbatchproc.so: ELF-64 shared object file - PA-RISC 2.0 (LP64)
```

The ELF-64 identifies this as a 64 bit file. A 32 bit file returns output like this:

```
$ file extproc32
extproc32: PA-RISC1.1 shared executable dynamically linked -not stripped
```

Also, be sure that the version of extproc being run is also 64 bit. Check the listener.ora file – generally located in `$ORACLE_HOME/networks/admin` – for a line that looks like this:

```
(PROGRAM = extproc)
```

Be sure it references extproc and not extproc32. Also, use the commands below to verify that the extproc program is really 64 bit, and not a renamed version of the 32 bit extproc:

```
$ cd $ORACLE_HOME/bin
$ file extproc
extproc: ELF-64 executable object file - PA-RISC 2.0 (LP64)
```

Data Migration on Oracle

Io exception: The Network Adapter could not establish the connection

When trying to create a connection to an Oracle database from the Database Tools section of the Management Console, the error message above is returned. This error indicates that the connection attempt failed without ever reaching the Oracle listener.

1. Verify that the IP address and port specified in the connection information is correct. If you have specified a DNS name, try using the IP address instead.
2. Verify that the TNSListener is running on the Oracle server and that it is accepting connections. Verify that it is running by issuing the command “lsnrctl status” for a default listener or “lsnrctl status <listener name>” for a listener with a name other than “LISTENER”. Try connecting to Oracle from another machine to verify that connections are being accepted.
3. Verify that there is network connectivity between the NAE Server and the Oracle Server using the ping utility available on the DataSecure appliance from the Network Diagnostics section of the Maintenance page on the Management Console.
4. Verify that there is no software (firewall, port blocking, IP blocking, etc.) that might be preventing the NAE Server from establishing a connection with the Oracle database. The listener port, typically 1521, might be blocked for connection requests from a specific IP address or a range of addresses. If possible, use a utility like Oracle's tnsping to try and connect to the Oracle Server from another machine in the same subnet as the NAE Server.

Index