

Column-level encryption

This article describes the column-level encryption feature of IBM® Informix® Dynamic Server (IDS) beginning with version 10.

What is column-level encryption?

You can use column-level encryption to improve the confidentiality of your data. New built-in SQL functions provide methods for data in columns to be stored in an encrypted format.

What it does

The `SET ENCRYPTION PASSWORD` statement declares a *password* to support data confidentiality through built-in functions that use the Triple-DES or AES algorithms for encryption and decryption. New built-in `ENCRYPT` functions provide methods for encrypting and decrypting the following character data types or smart large object data types:

- CHAR
- NCHAR
- VARCHAR
- NVARCHAR
- LVARCHAR
- BLOB
- CLOB

If you set column-level encryption passwords, data in the columns is stored in an encrypted format. Only users who can provide a secret password can view, copy, or modify encrypted data.

When you set an encryption password, you can also specify a password hint. If you specify a hint, you can store the hint with the encrypted password or in another location. The password must be a minimum of 6 bytes and can be a maximum of 128 bytes.

When you set a password, IBM® Informix® Dynamic Server (IDS) transfers the password and any hint to a 128-bit key that is used to encrypt the password and hint. Passwords and hints are not stored as plain text in any table of the system catalog. The key is a time-based random value per instance. The database server initializes the key when the server starts; the key is destroyed when the database server shuts down.

The Encrypt Virtual Processor

IDS uses an Encrypt Virtual Processor to handle encryption and decryption. If the `encrypt` option of the `VPCLASS` parameter is not defined in the `ONCONFIG` configuration file, the database server starts one `ENCRYPT` VP the first time that any encryption or decryption functions defined for column-level encryption are called. Use the `VPCLASS` configuration parameter with the `encrypt` keyword to configure encryption VPs. For example, to add five `ENCRYPT` VPs, add the following in the `ONCONFIG` file:

```
VPCLASS encrypt,num=5
```

You can dynamically add the same number of `ENCRYPT` VPs by using the `onmode` utility:

```
onmode -p 5 encrypt
```

Storage Considerations

An encrypted value uses more storage space than the corresponding plain text value. This occurs because all of the information needed to decrypt the value, except the encryption key, is stored with the value. Therefore, before you set the encryption password and encrypt data, you must be sure the encrypted data

Column-level encryption

can fit in the column.

How to use it

Use the `ENCRYPT_AES()` or the `ENCRYPT_TDES()` function to define encrypted data.

Use the `DECRYPT_BINARY()`, and `DECRYPT_CHAR()` functions to query encrypted data.

ExamplesEncrypting a Column

The following example demonstrates how to use the encryption functions with a column that contains a social security number:

```
create table emp ( name char(40), salary money, ssn lvarchar(64) );
set encryption password to "one two three 123";
insert into emp (ssn)
values ("Alice", 50000, encrypt_aes ('123-456-      7890'));
insert into emp (ssn)
values ("Bob", 65000, encrypt_aes ('213-656-0890'));
```

Querying an encrypted column

The following example demonstrates how to use the decrypt function to query encrypted data:

```
select name, salary, decrypt_char(ssn, "one two three 123")
from emp
where decrypt_char(ssn) = '123-456-7890';
```

or

```
set encryption password to "one two three 123";
select name, salary, decrypt_char(ssn)
from customer
where decrypt(ssn) = '123-456-7890';
```

Debugging Information

› XTRACE OpenSSL API failure only

› XTRACE Example call:

```
DTR_HEAVY(XTF_SQL, XTF_CALL, ("Encryption CIPHER is not defined"));
```

› Example trace command:

```
xtrace heavy -c XTF_SQL -f XTF_CALL
```