FORTRA

Powertech Encryption for IBM i 4.01

User Guide

Copyright Terms and Conditions

202311070807

Table of Contents

Welcome to Powertech Encryption for IBM i

Powertech Encryption for IBM i allows organizations to implement encryption quickly using intuitive screens and commands, while providing a high degree of protection. Every effort has been made in Powertech Encryption for IBM i to minimize the application changes needed, allowing an organization to implement encryption successfully for less time and money.

> NOTE: Data encryption has traditionally been very difficult and time-consuming to implement. In the past, major application changes were required to expand database field sizes and implement complicated API calls to encrypt/decrypt data. Additionally, organizations were not meeting stringent Key Management requirements by not properly securing and controlling their encryption Keys.

Powertech Encryption for IBM i Features

Powertech Encryption for IBM i includes the comprehensive features needed to satisfy stringent requirements for encryption and key management. The primary capabilities of Powertech Encryption for IBM i are:

- Automated encryption of database fields within IBM i DB2 database files and tables
- Support for automatic, transparent field encryption through use of DB2 Field Procedures
- Decryption of fields as full values or masked values, based on authorization lists
- Encryption of IBM i files, objects and libraries (backup encryption)
- Encryption of files located on the Integrated File System (IFS)
- Tokenization, encryption and storage of data from remote systems (IBM i, Windows, Linux, etc.)
- Integrated Key Management with Role-based administration
- Ability to integrate with enterprise key managers from other vendors
- Compliance with Advanced Encryption Standard (AES) and Data Encryption Standard (TDES)
- Intuitive IBM i menus and commands with on-line help text
- Program calls and ILE procedures (APIs) for decrypting data within native applications
- Stored procedures and SQL functions for decrypting data through SQL

- Restrict programmers from accessing decrypted values, even if they have *ALLOBJ authority
- Security alert messages to email addresses, message queues, SYSLOG and journals
- Comprehensive audit trails and reporting

Instructions

Before encrypting production data with Powertech Encryption for IBM i, it is important for you to have a full understanding of the product and how to implement good key management practices. This will help ensure that your data is properly protected and can be decrypted only by authorized users.

Review this manual for complete details on how to use Powertech Encryption for IBM i's commands and screens. On most command screens, press F9 to display additional fields. If you are a developer, also refer to the *Programmers Guide* for instructions on how to use Powertech Encryption for IBM i's procedures and programs (APIs) for encrypting/decrypting data from within your applications. If needed, contact Powertech Support for the *Programmers Guide*.

Using Powertech Encryption for IBM i

The following topics describe how to configure and use Powertech Encryption for IBM i.

Getting Started

This section describes how to quickly configure Powertech Encryption for IBM i's Symmetric Key Management settings and establish your first Data Encryption Key. These examples use commands; however, the same functions can be accessed using Powertech Encryption for IBM i's menu screens.

By the end of this section, you will know how to:

1. **Configure settings and keys.** Use the CHGKEYPCY command to review and/or change the Key Policy settings. See Configuring Settings and Keys.
2. **Add and configure key officers.** Use the WRKKEYOFR to indicate which users can create and manage keys. See Adding and Configuring Key Officers.
3. **Prepare and generate a Master Encryption Key.** Use the LODMSTKEY command to prepare a Master Encryption Key (MEK) by loading the passphrase parts, then use the CRYPTO/SETMSTKEY command to generate the MEK using the Loaded Passphrase Parts. See Configuring Master Encryption Keys.
4. **Create a Key Store.** Use the CRTKEYSTR command to create a Key Store to contain the Data Encryption Keys (DEK)). See Creating Key Stores.
5. **Create a Data Encryption Key.** Use the CRTSYMKEY command to create a Data Encryption Key (DEK) and save it into the Key Store. See Creating a Data Encryption Key (DEK) and Saving it to the Key Store.

NOTE:

- Commands also have online help text which can be accessed with the F1 key when a command is prompted.
- Sign in with user that has *ALLOBJ authority.
- To protect access to Powertech Encryption for IBM i using authorization lists, see Appendix E: Controlling Access using Authorization Lists.
- A default key store cannot be added to the Key Policy until it is created, which must happen after setting a master key. When you create the key store, you have the option to make it the 'default' key store, which populates this value automatically.

Configuring Settings and Keys

The Key Policy allows an organization to control the environment settings for Powertech Encryption for IBM i's Key Management System. These settings are encrypted with the Product Encryption Key (PEK) and are stored in the CRYPTO library by default.

Configuring Key Policy Settings

1. Prompt command **CHGKEYPCY** with F4. The Change Key Policy (CHGKEYPCY) panel appears.
2. Specify the policy settings for the Symmetric Key Management System. Consider the following recommendations:
   a. For ease of testing, set MEK Number of Passphrase parts to **1**. In setting up a production environment, set MEK Number of Passphrase Parts to **2** or **3**. This helps protect the security of the MEK (Master Encryption Key) since it can only be regenerated if all passphrase parts are entered.
   b. When setting up a production environment, set MEK Each Part by Unique User to ***YES**. (For testing, if only 1 passphrase is used, there is only one user). To protect the security of a MEK, you should require that each passphrase part is entered by a different user profile. Except in testing or extreme cases, a single user should not know all of the passphrase parts used to generate a MEK.
   c. Set default key store to ***NONE**. You will be able to indicate a key store is to be used as the default key store when you create the key store later in these steps. We suggest you use a default Key Store name, so the requester (i.e. the programmer or application) will not need to specify the name of the Key Store when requesting a DEK (Data Encryption Key) to use for encryption or decryption. This is not only a matter of convenience for the requester, but this can also help protect the known location of the default Key Store object.
   d. Set DEK can be Randomly Generated to ***YES**. DEKs should be randomly generated to provide the highest level of protection. A randomly generated DEK would be very difficult (virtually impossible) to recreate.
   e. Set DEK can be Passphrase Based to ***NO**. A passphrase-based DEK can be regenerated if the party knows the passphrase and algorithm used to generate the DEK. Therefore a passphrase-based DEK is not as secure as a randomly generated DEK. You should only allow passphrase-based generated DEKs if those DEKs need to be regenerated on other platforms.
   f. Set DEK can be Manually Entered to ***NO**. A manually-entered DEK value is the least secure since this DEK value could be used to decrypt data without using Powertech Encryption APIs and security mechanisms. You should only allow the

manual entry of DEK values when needing to store/use DEKs which were generated on other platforms.

g. Set DEK Values can be Retrieved to ***NO** or ***KEK**. If a DEK value is retrievable, then the DEK actual Key value could be used to decrypt data without using Powertech Encryption APIs and security mechanisms. This should only be allowed if the key values need to be shared with another computer system (which is not an IBM i) that needs to encrypt or decrypt data using the same key.

h. In a test environment, set DEK Encrypt and Decrypt Usage by Owner to ***YES**. In a production environment, to provide separation of duties and to help protect the security of encrypted data, set these values to ***NO**. In production, the creator of a DEK should not be able to use the DEK to encrypt and decrypt data.

i. Set DEK can be Deleted to ***NO**. The organization may have existing data which is encrypted with a DEK. The accidental deletion of a DEK may result in unrecoverable data.

j. Set Limit all-object authority to ***NO** in your test environment for ease of testing. This allows all user profiles that have *ALLOBJ special authority to access key stores and authorization lists. In a production environment, set Limit all-object authority to ***YES** to enforce the use of object authority and authorization lists, to limit even *ALLOBJ profile users' access to sensitive data.

You can also use the command line to quickly apply your settings:

EXAMPLE:
For a test environment:
CRYPTO/CHGKEYPCY MEKPRT(1) MEKUNQUSR(*YES) DEKRNDGEN(*YES) DEKPASBSD(*NO) DEKMANENT(*NO) DEKRTVVAL(*NO) DEKENCOWN(*YES) DEKDECOWN(*YES) DEKDLTALW(*NO) LMTALLOBJ(*YES)

EXAMPLE:
For a production environment:
CRYPTO/CHGKEYPCY MEKPRT(2) MEKUNQUSR(*YES) DEKRNDGEN(*YES) DEKPASBSD(*NO) DEKMANENT(*NO) DEKRTVVAL(*NO) DEKENCOWN(*NO) DEKDECOWN(*NO) DEKDLTALW(*NO) LMTALLOBJ(*YES)

Adding and Configuring Key Officers

Key Officers are those users that are authorized to create and manage Master Encryption Keys (MEKs), Key Stores, Data Encryption Keys (DEKs) and the Field Encryption Registry.

NOTE: A user does not need to be a Key Officer to encrypt and decrypt data.

Adding Key Officers

1. Submit the command **CRYPTO/ADDKEYOFR**. This opens the Add Key Officer (ADDKEYOFR) panel, where you can add a new key officer.
2. Specify the new key officer's user profile and review/configure the remaining settings. See Work with Key Officers (WRKKEYOFR) panel for details.
3. Press Enter to add the Key Officer.

> **NOTE:** You can also use the following commands to work with Key Officers:
>
> - **CHGKEYOFR** - Change Key Officer
> - **DSPKEYOFR** - Display Key Officer
> - **RMVKEYOFR** - Remove Key Officer
> - **WRKKEYOFR** - Work with Key Officers

Configuring Master Encryption Keys

A Master Encryption Key (MEK) is an AES 256 bit Symmetric Key used to protect (encrypt) the Data Encryption Keys (DEKs) contained in a Key Store. An organization can create up to 8 MEKs per environment on the IBM i, (though it is common to create and use just 1 MEK). For instance, an MEK could be created to encrypt the Order Entry DEKs contained in a Key Store, and a second MEK could be created to encrypt the Payroll DEKs contained in another Key Store.

An MEK is generated by Powertech Encryption for IBM i using passphrases entered by designated Key Officers. Depending on the organization's key policy, up to 8 different passphrases can be required (by different users) in order to generate an MEK.

The Master Encryption Keys (MEK) are stored in a product-supplied validation list (*VLDL) object. The MEKs are encrypted with the Product Encryption Key (PEK).

> **NOTE:** To display existing Master Key Attributes, see Display Master Key Attributes (DSPMSTKEY). You can clear an existing Master Encryption Key using Clear Master Encryption Key (CLRMSTKEY).

Master Encryption Key (MEK) Versions

Each MEK can have up to three versions which are named *NEW, *CURRENT and *OLD:

*NEW Version

The *NEW version of an MEK is the version in which passphrases are being entered (loaded) by users with the LODMSTKEY (Load Master Key) command. The *NEW version cannot be used to encrypt DEKs within Key Stores. In order to convert the *NEW version into the *CURRENT version, an authorized Key Officer must set the Master Key using the CRYPTO/SETMSTKEY command.

*CURRENT Version

The *CURRENT version of an MEK is the current version that can be associated with Key Stores.

*OLD Version

The *OLD version of an MEK is the prior *CURRENT version of the MEK. The *OLD version cannot be associated with new Key Stores. However, DEKs in current Key Stores may still be encrypted under the *OLD version until they are translated (using the TRNKEYSTR command).

> EXAMPLE: Once an *OLD version of an MEK exists, you must translate Key Stores using the *OLD MEK to use the *CURRENT version of the MEK. This must be done before you convert a *NEW MEK to the *CURRENT version.

Preparing a Master Encryption Key (MEK) by Loading the Passphrase Parts

1. Submit the command **LODMSTKEY**.
2. Configure the fields as needed. Enter the ID Number, MEK Passphrase Part, and Passphrase, as well as whether the passphrase will replace an existing passphrase for the part specified. See Load Master Encryption Key (LODMSTKEY) panel for the details and rules required for these fields.

   > WARNING:
   > The passphrase parts used to load a MEK should be recorded in a safe place (not on the IBM i). An MEK will not be usable if it's copied or restored to another IBM i serial number. If you want to recreate the same MEK on another IBM i serial number (i.e. in a disaster recovery situation), these same passphrase parts will have to be re-entered (loaded) in the same order.

3. Press Enter to load the Master Encryption Key.

> NOTE: Maintenance of Master Encryption Keys is logged into an audit file.

You can also use the command line to quickly apply these settings. In a test environment, consider using a single passphrase to simplify your initial setup and testing:

EXAMPLE:
CRYPTO/LODMSTKEY MEKID(1) MEKPRT(1) PASSPHRASE(Passw0rd)

In a production environment, multiple passphrases are recommenced for enhanced security. Repeat the LODMSTKEY command for MEKPRT(2) to load all the required parts, per your Key Policy.

Generating the MEK using the Loaded Passphrase Parts

WARNING: The SETMSTKEY command will replace the *OLD version of the MEK with the *CURRENT version. Before running this command, you should first use the TRNKEYSTR command to translate (re-encrypt) any DEKs in the Key Stores which are still encrypted with the *OLD version of the MEK.

After all of the required passphrase parts have been entered (loaded) for a MEK, the *CURRENT version of the MEK can be set with the SETMSTKEY command. To do so:

1. Submit the command **CRYPTO/SETMSTKEY**.
2. Indicate the id number of the Master Encryption Key (MEK) to set. See Set Master Encryption Key (SETMSTKEY) panel for details, including an explanation of the specific programmatic actions performed by the command.
3. Press Enter.

NOTE: After running SETMSTKEY… If existing DEKs in Key Stores are encrypted with the MEK, then you should execute the TRNKEYSTR command to translate (re-encrypt) the DEKs in the Key Stores.

You can also use the command line to quickly apply these settings:

EXAMPLE:
CRYPTO/SETMSTKEY MEKID(1)

Creating Key Stores

Data Encryption Keys (DEK) are contained within Key Stores. An organization can create one or more Key Stores on the IBM i. For instance, one Key Store could be used to contain

DEKs for protecting Order Entry data, and a second Key Store could be used to contain DEKs for protecting Payroll data.

Each Key Store is created as a Validation List (*VLDL) object on the IBM i. The name of the *VLDL object is specified on the CRTKEYSTR (Create Key Store) command. The DEKs contained in a Key Store are encrypted with a user-specified Master Encryption Key (MEK).

As mentioned above, when you create a key store, you can indicate that it will be set as the Default Key Store in your Key Policy. Use the SETDFT parameter to set your key store to be the default key store.

Creating a Key Store to contain the Data Encryption Keys (DEK)

1. Submit the command **CRTKEYSTR**.
2. Indicate the Key Store Name, Library, MIK ID, and other values. See Create Key Store (CRTKEYSTR) panel for more details.
3. Press Enter to create the Key Store.

You can also use the command line to quickly apply these settings:

EXAMPLE:
CRYPTO/CRTKEYSTR KEYSTR(KEYSTORELIB/KEYSTORE) CRTLIB(*YES) MEKID(1) SETDFT(*YES)

NOTE: For information on managing Key Stores—for example, to translate, display, or delete a Key Store—see Working with Key Stores.

To control access using Authorization Lists, or by granting authority to a Key Store

The CRTKEYSTR command, by default, grants *PUBLIC access to the keystore as it is created. This allows all users access to the keystore. To control access to encrypting and decrypting data, specify Authorization Lists for Full and Masked access to data when you set up the Field or IFS encryption. Since a Key Store is created as a validation list (*VLDL) object on the IBM i, you can additionally control the authority to a Key Store by using IBM's EDTOBJAUT (Edit Object Authority) command. See Editing the Authority on a Key Store.

NOTE: Setting the object authority on a Key Store will control which users can manage keys in the Key Store, as well as which users can utilize the keys within the Key Store for encrypting and decrypting data.

For a complete discussion regarding using Key Store Authority and Authorization Lists to control encryption and decryption, see Controlling Access to Decrypted Values.

Creating a Data Encryption Key (DEK) and Saving it into the Key Store

1. Submit the command **CRTSYMKEY**.
2. Indicate the Key Label The key will be created in the *DEFAULT key store if one was specified, or include the Key Store Name, and other values. See Create Symmetric Key (CRTSYMKEY) panel for more details.
3. Press Enter to create the Symmetric Key.

> EXAMPLE:
> **CRYPTO/CRTSYMKEY KEYLABEL(KEYLABEL)**

Creating a Keystore Using two Default Passphrases

While a single passphrase may be appropriate for internal testing, in a production environment, multiple passphrases are recommended. For example, you can use the following procedure to create a new Keystore using the default two key passphrase.

The following requires two users with *ALLOBJ authority.

1. Sign in with Profile1 user.

```
CRYPTO/ADDKEYOFR USRPRF(<Profile1>) MNTPCYALR(*YES) MNTKEYOFR
(*YES)
CRYPTO/LODMSTKEY MEKID(1) MEKPRT(1) PASSPHRASE(Passw0rd)
CRYPTO/ADDKEYOFR USRPRF(<Profile2>) MNTPCYALR(*YES) MNTKEYOFR
(*YES)
```

2. Sign in with Profile2 user

```
CRYPTO/LODMSTKEY MEKID(1) MEKPRT(2) PASSPHRASE(Passw0rd2)
```

3. Sign in with Profile1 user

```
CRYPTO/SETMSTKEY MEKID(1)
CRYPTO/CRTKEYSTR KEYSTR(KEYSTORELIB/KEYSTORE) CRTLIB(*YES)
MEKID(1)
CRYPTO/CRTSYMKEY KEYLABEL(KEYLABEL) KEYSTR
(KEYSTORELIB/KEYSTORE)
```

Find Sensitive Database Fields

A menu is provided that allows you to find some database fields which may contain sensitive data, such as credit card numbers, social security numbers and Canadian social insurance numbers. To access this menu, execute the command:

```
GO CRYPTO/CRYPTO9
```

Encrypting Database Fields

After setting up a Data Encryption Key, database fields can then be encrypted using one or more of the following approaches:

- To set up database field encryption quickly (using IBM's Field Procedure functions) use the automated commands WRKLIBFILS (to select a file from a list in a library) or WRKFILFLDS (to work with all fields in a file).
- To review existing database field encryption or to set up field encryption manually, use the WRKFLDENC (Work with Field Encryption Registry) command.

If you intend to write your own programs ILE procedures or SQL functions:

- Call native ILE procedures or programs (APIs) to encrypt/decrypt data within applications *
- Call SQL functions or stored procedures to encrypt/decrypt data using SQL *

* If needed, contact Powertech Support for the *Programmers Guide*.

Encrypting IFS Files

See the IFS Encryption Guide for information on encrypting IFS objects.

Encrypting Libraries, Objects, and Files

After setting up a Data Encryption Key, you can also use one of the following commands to encrypt IBM i libraries, objects and files to tape or disk:

- ENCSAVLIB – For encrypting libraries
- ENCSAVOBJ – For encrypting specific objects within a library
- ENCSTMF – For encrypting IFS stream files

Create Security Alerts

Using the WRKCCALR command, you can optionally set up Security Alerts, which can send immediate notifications when security-related changes or authority errors occur in Powertech Encryption for IBM i.

Test vs. Production Environment

In setting your Key Policy for a test environment, earlier in these instructions, we suggested using a one part passphrase, allowing *ALLOBJ users to encrypt/decrypt data with no additional authorization, and allowing the key owner to encrypt and decrypt data. We recommend changing these settings to be more secure if you are changing your installation from a test environment to a production environment. If you LOAD and SET a new MEK to add new passphrases and unique users, be sure to translate existing key stores.

Symmetric Key Management

Symmetric Key Cryptology (also known as Secret Key or Private Key Cryptology) is a form of cryptology in which the same Key can be used to encrypt and decrypt data.

Symmetric Keys must be strong enough for the intended application. Because the strength of the Symmetric Key is determined by its length, the longer the key, the harder it is for high-speed computers to break the code. Within Powertech Encryption for IBM i, Symmetric Keys may be generated up to 256 bit lengths to provide a high level of protection.

The Symmetric Key values must be kept secret to prevent unauthorized decryption of sensitive data. Controls must therefore exist to protect the confidentiality and access to the Symmetric Keys. Powertech Encryption for IBM i provides an integrated and comprehensive Symmetric Key Management System to establish those controls.

> NOTE:
> A **Cipher** is a pair of algorithms (mathematical processes) used to encrypt and decrypt data.
>
> A **Key** is the information needed to control the detailed operations of the Cipher. In contrast to human-generated passwords, Keys are more secure since they are computer-generated and are represented as an obscure series of bits (1001110…).

Powertech Encryption for IBM i's Symmetric Key Management System allows organizations to:

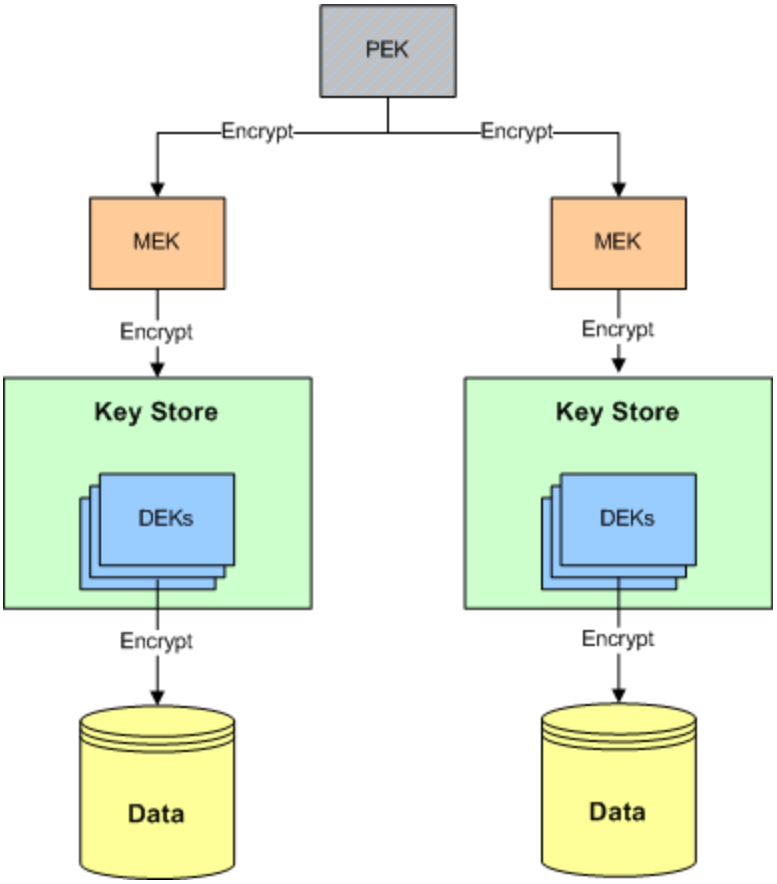
- Establish policy settings on how Symmetric Keys can be created and utilized
- Indicate which users can create and manage Symmetric Keys
- Randomly generate strong Symmetric Keys

- Protect Symmetric Keys using Master Encryption Keys
- Protect the recreation of a Master Encryption Key by requiring passphrases from up to 8 users
- Organize Symmetric Keys into one or more Key Stores
- Restrict access to Key Stores using IBM i object authority
- Restrict the retrieval of the actual Symmetric Key values
- Provide separation of duties (e.g. the creator of a Symmetric Key can be restricted from using the Key to encrypt and/or decrypt data)
- Control which users can utilize Symmetric Keys to encrypt and decrypt data
- Produce detailed audit logs

IMPORTANT: The encryption algorithm of the symmetric key is used to determine the encryption algorithm with which the IFS files will be encrypted. For instance, if a symmetric key is created with AES-256, then the IFS files will also be encrypted with AES-256. See IFS Encryption for more information.

Symmetric Key Hierarchy

Powertech Encryption for IBM i provides a multi-level security architecture to protect Symmetric keys on the IBM i. The diagram for this hierarchy is outlined below (with descriptive text following the diagram).



A Data Encryption Key (DEK) is a Symmetric Key which is used to encrypt and decrypt data. An organization can create one or more DEKs using Powertech Encryption for IBM i. For instance, a DEK could be created to encrypt/decrypt credit card numbers and a second DEK could be created to encrypt/decrypt social security numbers.

A DEK should be randomly generated by Powertech Encryption for IBM i in order to provide the highest degree of protection. Depending on your organization's key policy, you can additionally have Powertech Encryption for IBM i generate a DEK which is based on a passphrase entered by the user.

Data Encryption Keys (DEK) are contained within Key Stores. You can create one or more Key Stores on the IBM i using Powertech Encryption for IBM i. For instance, one Key Store could be used to contain DEKs for protecting Order Entry data, and a second Key Store could be used to contain DEKs for protecting Payroll data.

A Key Store is created as a *VLDL (Validation List) object on the IBM i. You can control access to the Key Store *VLDL object using IBM i object security.

A Master Encryption Key (MEK) is a special Symmetric Key used to protect (encrypt) the Data Encryption Keys (DEKs) contained in a Key Store. An organization can create up to 8 MEKs per environment on the IBM i. For instance, a MEK could be used to encrypt the Order Entry DEKs contained in a Key Store, and a second MEK used to encrypt the Payroll DEKs contained in another Key Store.

A MEK is generated by Powertech Encryption for IBM i using passphrases entered by designated users. Depending on the organization's key policy, up to 8 different passphrases can be required (by different users) in order to generate a MEK.

MEKs are stored in a *VLDL (Validation List) object on the IBM i called CRVL001.

A PEK is used by Powertech Encryption for IBM i to protect (encrypt) the Master Encryption Keys (MEKs) and user-defined settings (i.e. Key Policy, Key Officers, Security Alerts, etc).

Powertech Encryption for IBM i automatically generates the PEK using a combination of the IBM i serial number and a secret value. The PEK only resides in memory as-needed and is never stored.

Data Encryption Keys

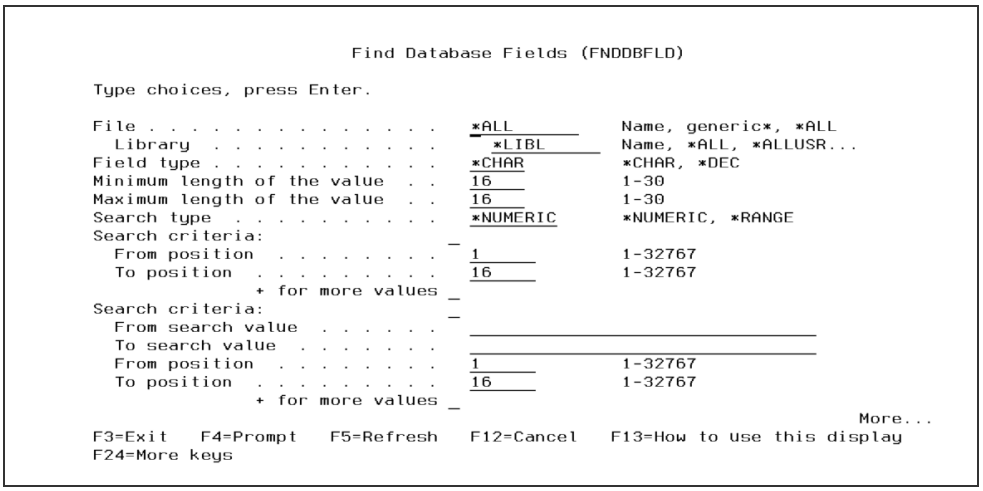
A Data Encryption Key (DEK) is a Symmetric Key which is used to encrypt and decrypt data. An organization can create one or more DEKs using Powertech Encryption for IBM i. For instance, a DEK could be created to encrypt/decrypt credit card numbers and a second DEK could be created to encrypt/decrypt social security numbers.

A DEK should be randomly generated by Powertech Encryption for IBM i in order to provide the highest degree of protection. Depending on your organization's key policy, you can additionally have Powertech Encryption for IBM i generate a DEK which is based on a passphrase entered by the user.

Find Database Fields

The Find Database Field (FNDDBFLD) command allows you to find database fields (in physical files and tables) that contain values which meet your search criteria. This is especially useful for finding fields that contain unencrypted sensitive data such as credit card numbers, social security numbers and Canadian social insurance numbers. For instance, you can quickly perform a search for any numeric fields that contain a 16 digit number (e.g., credit card numbers) or perform a search for any alpha fields that contain a numeric pattern like 999-99-999 or 999999999 (e.g., Social Security numbers).

TIP: The FNDDBFLD command is provided on a menu with several variations of parameters for finding unencrypted credit card numbers, social security numbers, etc. To access this menu, run the command of: GO CRYPTO/CRYPTO9



Multiple files and libraries can be searched with the FNDDBFLD command. It can search both numeric and character fields.

It is recommended to run the FNDDBFLD command in batch using the SBMJOB command.

Do the following steps to use the FNDDBFLD command:

1. Prompt (**F4**) the command **CRYPTO/FNDDBFLD**.
2. Press **F1** on any parameter for complete online help text.
3. Type in the parameter values, then press Enter to execute.

The search results are formatted in a report (spooled file). The report will indicate the following information for each database field that meets the selection criteria:

- File name
- Library name
- Field name
- Relative record number (RRN) of the data found (for the first match)
- The data found in the field (for the first match)

NOTE:
When searching numeric fields, FNDDBFLD will ignore fields that contain decimal positions.
If you have multi-member files, only the first member in each file will be searched.

Field Encryption

Field values can be encrypted and decrypted using a variety of methods in Powertech Encryption for IBM i, providing a great deal of flexibility for an organization. For each database field, you can choose the technique to utilize based on your application requirements.

The preferred option for field encryption is to use Powertech Encryption for IBM i's innovative Field Encryption Registry, which allows an organization to indicate (register) the database fields to encrypt. When a field is "activated" in the Registry, Powertech Encryption for IBM i will perform a mass encryption of the current values for that field. Powertech Encryption for IBM i can then automatically encrypt the field values on an ongoing basis as new database records are added and when existing field values are changed.

The automated encryption function in Powertech Encryption for IBM i's Field Encryption Registry will eliminate the need to make changes to your application programs for data encryption. If DB2 Field Procedures (available in IBM i V7R1) are utilized, the values can also be automatically decrypted without program changes. Otherwise, simple program changes can be made to decrypt values using Powertech Encryption for IBM i's APIs.

You can optionally modify your applications to encrypt data through program (API) calls to Powertech Encryption for IBM i's encryption procedures and programs. Powertech Encryption for IBM i also includes stored procedures and SQL functions, which can be called from within native applications or other external clients (i.e., graphical or web-based front ends) for encryption/decryption.

Encryption Basics

Encrypted data (Cipher Text) is in alphanumeric format. Since the encryption algorithms use the full character set, you will see encrypted data as a combination of letters, special characters and numbers.

EXAMPLE:
Before: The quick brown fox jumped over the lazy dog

After: „Œ \ËKä°BBY ý\åê·Ñ,C‹Ÿ^{F+rAÀJ[13]˜()$j1Ï(¾Y½i›”®t

Encryption Algorithms

Powertech Encryption for IBM i implements the AES and TDES encryption algorithms (ciphers). Both of these algorithms follow standard (non-proprietary) specifications as published by the United States National Institute of Standards and Technology (NIST).

The TDES (Triple DES) standard was introduced in 1998. It is so named because it applies the Data Encryption Standard (DES) cipher algorithm three times to each data block. TDES is slowly disappearing from use because of performance issues and weaker key sizes.

The AES (Advanced Encryption Standard) standard was introduced in 2001. AES is the first publicly accessible and open cipher approved by the US Government for top secret information. AES offers high performance and provides strong key lengths up to 256 bits. AES is a very popular cipher for field encryption because of those attributes.

Encryption Modes

The AES and TDES standards offer different operating “modes” that you can choose from. Powertech Encryption for IBM i provides support for the CUSP, ECB and CBC modes.

CUSP Mode (Cryptographic Unit Support Program)

CUSP mode is supported in the AES algorithm. This is a stream-based mode, which means that the length of the encrypted data will equal the length of the input data. This mode is useful if the field data is not divisible by a block length and if you want to store the encrypted values in your existing field (if not using a DB2 Field Procedure).

With CUSP mode, you can optionally specify an Initialization Vector (IV). An Initialization Vector (IV) is an arbitrary value that you can enter, which will be used as an additional input to the encryption algorithm. Therefore, the encrypted output is dependent on the combination of the Initialization Vector, Encryption Key and the Plain Text (the data you want to encrypt).

ECB Mode (Electronic Code Book)

ECB mode is supported in both AES and TDES algorithms. This is a block-based * mode. With ECB mode, you cannot specify an Initialization Vector.

CBC Mode (Cipher Block Chaining)

CBC mode is supported in both AES and TDES algorithms. This is a block-based * mode. With CBC mode, you can optionally specify an Initialization Vector (IV) to use as an additional input to the encryption algorithm.

*Notes on Block-based Modes

When using the AES algorithm with CBC or ECB modes, the length of the encrypted data will be a minimum of 16 bytes long. Its “block-based” length will be divisible by 16 or 24. For instance:

> **NOTE:** For CBC and ECB modes: If the alphanumeric field length is not divisible by the block length, then you can choose to store the encrypted values in a separate external file or use a DB2 Field Procedure.

Original Field Length	Encrypted Length
10 bytes	16 bytes
16 bytes	16 bytes
17 bytes	24 bytes
24 bytes	24 bytes
31 bytes	32 bytes

When using the TDES algorithm with CBC or ECB modes, the length of the encrypted data will be a minimum of 8 bytes long. Its “block-based” length will be divisible by 8. For instance:

Original Field Length	Encrypted Length
5 bytes	8 bytes
8 bytes	8 bytes
9 bytes	16 bytes
16 bytes	16 bytes

Field Encryption Registry

Powertech Encryption for IBM i’s Field Encryption Registry allows an organization to specify (register) the database fields that require encryption. There are several configurable options that you can specify for each database field added to the registry.

One option is to have Powertech Encryption for IBM i create SQL triggers on your database file that will automate the encryption of database field values when records are inserted (added) and when field values are updated (changed) in the database. This allows you to minimize application changes by only focusing on those programs that need to retrieve decrypted values.

For customers on IBM i release V7R1 or higher, another option is to have Powertech Encryption for IBM i place a DB2 Field Procedure on the database field to automate both the encryption and decryption of its values. This option has the potential to eliminate any application changes.

Storage of Encrypted Values

The Field Encryption Registry provides a user-specified option for indicating where the encrypted field values should be stored. You can choose to store the encrypted values in the 'encoded' portion of the field (using a DB2 Field Procedure), within the existing field space, or in a separate external file. The flexibility in the Registry allows you to specify a different storage option for each field that is encrypted.

Store with DB2 Field Procedure

When defining a database field in the Field Encryption Registry, you can choose to store the encrypted values within the "encoded" portion of the field using a DB2 Field Procedure. This option is valid if your system is running IBM i release V7R1 or higher. This approach works for alphanumeric, numeric, date, time and timestamp field types.

> **NOTE:** Before using DB2 Field Procedures in a production environment, read Appendix B: DB2 Field Procedures to understand the potential performance issues and risks.

Store in the Existing Field

When defining a database field in the Field Encryption Registry, you can choose to store the encrypted values within the existing field space, as long as the field type is alphanumeric (char) and meets the following algorithm, mode and length requirements:

- AES algorithm with CUSP mode.
- AES algorithm with CBC or ECB modes, as long as the maximum length of the values in the field is divisible by the block lengths of 16 or 24.
- TDES algorithm, as long as the maximum length of the values in the field is divisible by the block length of 8.

Store in an External File

When defining a database field in the Field Encryption Registry, you can specify that the encrypted values should be stored in a separate external physical file, which will be created and maintained by Powertech Encryption for IBM i. Storing encrypted field values in a separate external file has the following advantages:

- Numeric field types can also be encrypted.
- If using ECB or CBC modes, the field lengths do not have to be divisible by the encryption algorithm's block length.
- The Data Encryption Key (DEK) can be rotated at any time without having to re-encrypt all of the existing field values. Up to 99,999 DEKs can be specified for a field.

- Additional useful information can be stored in the external file, such as recording the user ids and timestamps when field values are updated (encrypted) and retrieved (decrypted).

When using the external file storage option, a separate physical file will be created for each database field which is activated in the Field Encryption Registry. The name of the external file can be specified by the user or can be generated by Powertech Encryption for IBM i.

If the external file names are requested to be generated by Powertech Encryption for IBM i, then the naming convention used is CRXXnnnnn, where “CRXX” is constant and “nnnnn” is a sequential number from 1 to 99999.

The object description of the created external file will contain the name of the database library/file and the name of the database field, for which it is storing encrypted values.

The record layout for the external file is listed below:

Field	Example value	Optional
Field Identifier	CREDIT_CARD	
Index number	7	
Key id	2	
Last updated by user	BILL	
Last updated time	2009-07-10-18.09.39.375000	
Last retrieved by user	MARY	Yes
Last retrieved time	2009-07-15-01.22.32.567000	Yes
Record hash	œôÊA·	Yes
Encrypted value	{¥háö,q'M™TVà#	

Powertech Encryption for IBM i will create a record in the field’s corresponding external file for each encrypted field value. Each record in the external file will be assigned a unique sequential Index number. This Index number should additionally be stored within the existing field in your application’s database file. Using the prior example, the index number of 7 should be stored in the existing database field.

When the decrypted field value needs to be retrieved, the Index number (stored in the existing database field) will be passed by your application to a procedure in Powertech Encryption for IBM i. It will use this index number to fetch the encrypted value from the external file. Powertech Encryption for IBM i will then decrypt this field value and return it to the application (if authorized).

> WARNING: If storing the encrypted values into an external file, then your existing field (to encrypt) should be large enough to hold the generated Index numbers. For instance, if your existing file does not (and will not) contain more than 999,999 records, then the existing field (to encrypt) should have a minimum length of 6 to hold an Index number up to 999999.

External storage – Optional Logical file

When using the external file storage option, a logical file can optionally be created by Powertech Encryption for IBM i over the external physical file. This logical file will be keyed by the Field Identifier and Encrypted value. This is useful if you need to retrieve a record from (chain out to) the external file using an encrypted field value that specified by a user or application.

Program APIs

If SQL triggers or DB2 Field Procedures are not used to automate the encryption of the field values, then the applications that maintain records in the database file should be modified to call Powertech Encryption for IBM i's APIs * for encrypting data. Read about the InsEncFld, UpdEncFld and DltEncFld APIs if you are storing the encrypted field values externally. Read about the EncFld API if you are storing the encrypted field values within the existing database field.

If DB2 Field Procedures are not used to auto-decrypt, then the applications that need access to the decrypted field values should be modified to call Powertech Encryption for IBM i's APIs * for decrypting data. Read about the GetEncFld, GetEncFldMask and GetEncFldAuth APIs if you are storing the encrypted field values externally. Read about the DecFld, DecFldMask and DecFldAuth APIs if you are storing the encrypted field values within the existing database.

Powertech Encryption for IBM i's APIs are documented in the *Programmers Guide*. If needed, contact Powertech Support for the *Programmers Guide*.

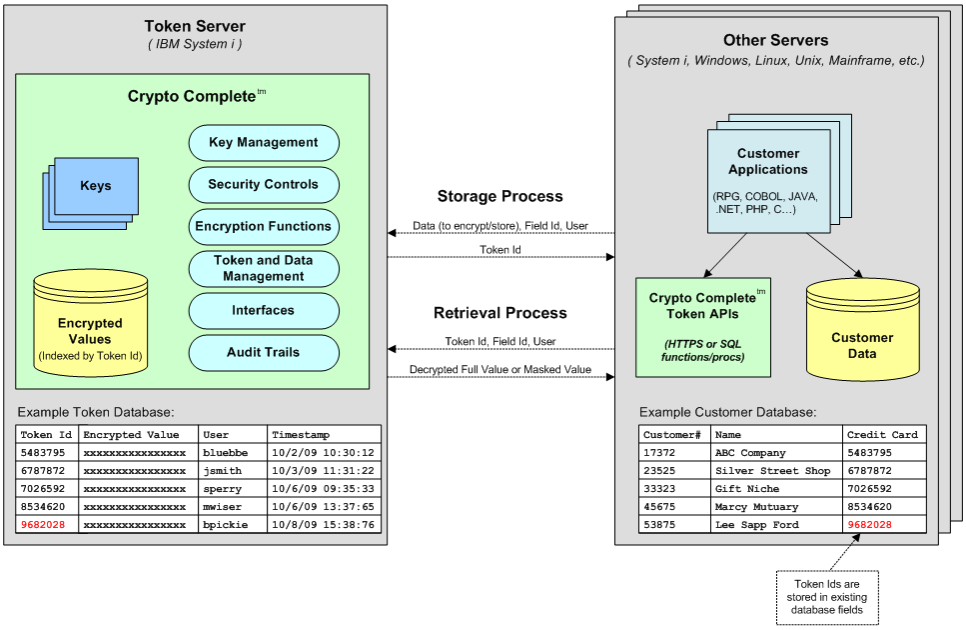
Tokenization - An Overview

Tokenization should be considered when sensitive data is stored on multiple systems throughout an organization. Tokenization is the process of replacing sensitive data with unique identification numbers (e.g. tokens) and storing the original data on a central server, typically in encrypted form. By centralizing all sensitive data onto a single system, tokenization can help thwart hackers and minimize the scope of compliance audits such as PCI.

Powertech Encryption for IBM i offers several advantages when tokenization is required:

- Centralizes key management and policies on a single server
- Supports tokenization of data from diverse systems including IBM i, Windows, Linux, AIX, etc.
- Provides remote connections to token functions through standard HTTP(S) protocol
- Auto-assigns token identifiers from the central token server
- Encrypts and stores tokenized data into scalable DB2 physical files
- Allows securing data elements by User Id, User Group and/or Authorization Lists
- Provides centralized audit logs and message alerts

Listed below is a diagram that illustrates Powertech Encryption for IBM i within a tokenized environment.



For more information regarding Tokenization, please refer to the Powertech Encryption for IBM i Programmers Guide.

Controlling Access to Decrypted Values

There are two levels of security that can be used in Powertech Encryption for IBM i to control access to the decrypted field values. At the 1st level of security, you should give *USE authority to the Key Store objects (which hold the decryption Keys) only for those users (or user groups) that can perform decryption, and *USE authority to the library that contains the Key Store. At the 2nd level of security, you can assign Authorization Lists to the field entries in the Field Encryption Registry and use designated APIs to only return the authorized values.

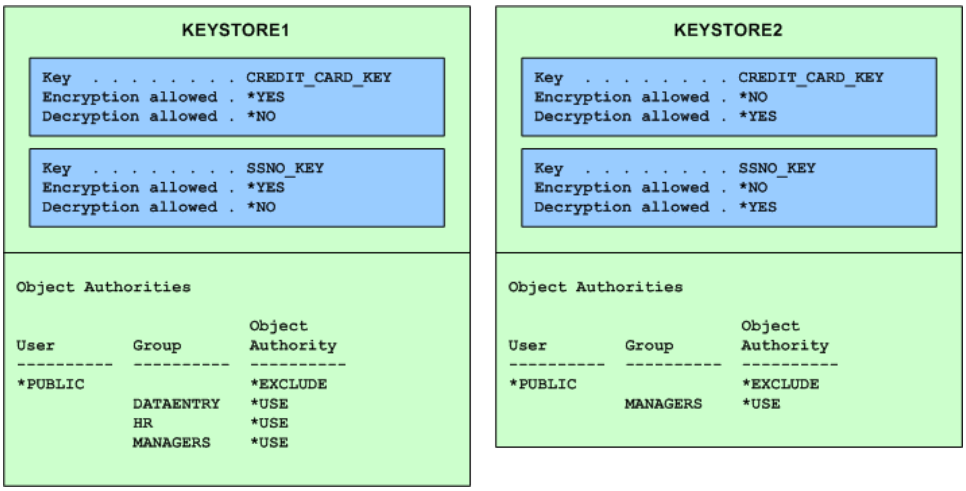
1st Level of Security - Key Store Authority

When a Key is requested to encrypt or decrypt data, Powertech Encryption for IBM i will first check if the user has at least *USE authority to the Key Store which holds the Key. If the user is not authorized to the Key Store, then the Key request will be denied and subsequently the user will not be able to proceed with the encryption or decryption of the data. The authority error will also be logged in the audit journal file.

It is likely that you will want a smaller group of users that can access (decrypt) sensitive data, compared to a larger group of users that can enter (encrypt) this data. This can be accomplished by using two Key Stores with their own respective authorities. In the first Key Store, you can store the Keys needed for encryption and give that Key Store a broader set of authorities. In the second Key Store, you can place the Keys needed for decryption and give that Key Store a much smaller set of authorities.

> **EXAMPLE:**
> For instance, any data entry user may be allowed to enter a credit card number and therefore must be authorized the Key Store holding the encryption key. However, perhaps only Supervisors are allowed to view the credit card numbers and must therefore be authorized to the Key Store holding the decryption key.

Each Key within a Key Store can be designated for encryption only, decryption only, or both encryption and decryption. In the following example, the Keys in KEYSTORE1 may only be used for encryption and the Keys in KEYSTORE 2 may only be used for decryption. Notice that KEYSTORE1 has a broader list of authorities than KEYSTORE2.



(see following steps to implement this example)

Key Store Authority example

Listed below are the steps needed to implement two Key Stores with different authorities:

1. Create KEYSTORE1 to contain the Key(s) needed for encryption.

   EXAMPLE:
   > CRYPTO/CRTKEYSTR KEYSTR(library/KEYSTORE1) MEKID(master-key-id)

2. For KEYSTORE1, grant *USE authority only to those users (or user groups) that can perform encryption.

   EXAMPLE:
   > EDTOBJAUT OBJ(library/KEYSTORE1) OBJTYPE(*VLDL)

3. Create KEYSTORE2 to contain the Key(s) needed for decryption.

   EXAMPLE:
   > CRYPTO/CRTKEYSTR KEYSTR(library/KEYSTORE2) MEKID(master-key-id)

4. For KEYSTORE2, granting *USE authority only to those users (or user groups) that can perform decryption.

   EXAMPLE:
   > EDTOBJAUT OBJ(library/KEYSTORE1) OBJTYPE(*VLDL)

5. Create the Key in KEYSTORE1. Allow the Key to be used for encryption only.

   EXAMPLE:
   > CRYPTO/CRTSYMKEY KEYLABEL(CREDIT_CARD_KEY) KEYSTR (library/KEYSTORE1)
   ENCRYPTALW(*YES) DECRYPTALW(*NO)

6. Copy the Key from KEYSTORE1 to KEYSTORE2.

   EXAMPLE:
   > CRYPTO/CPYSYMKEY FRMLABEL(CREDIT_CARD_KEY) FRMKEYSTR (library/KEYSTORE1)
   TOLABEL(*FRMLABEL) TOKEYSTR(library/KEYSTORE2)

   NOTE: Since Powertech Encryption for IBM i uses Symmetric cryptology, the actual key values for the encryption and decryption keys must be the same. This is why the decryption key must be copied from the first Key Store into the second Key Store, versus re-created.

7. Change the Key in KEYSTORE2 so it can only be used for decryption.

   EXAMPLE:
   > CRYPTO/CHGSYMKEY KEYLABEL(CREDIT_CARD_KEY) KEYSTR (library/KEYSTORE2)
   ENCRYPTALW(*NO) DECRYPTALW(*YES)

8. If using the Field Encryption Registry to encrypt the field values, listed below is an example of specifying the Key in KEYSTORE1 for encryption and the Key in KEYSTORE2 for decryption.

   EXAMPLE:
   > ADDFLDENC… ENCKEYLBL(CREDIT_CARD_KEY) ENCKEYSTR (library/KEYSTORE1)
   DECKEYLBL(CREDIT_CARD_KEY) DECKEYSTR(library/KEYSTORE2)

2nd Level of Security – Field Registry Authorization Lists

Authority settings can control what portion of the field values are available for users and groups. For instance, one group of users could be authorized to the fully decrypted field values, whereas a second group could be authorized to just the masked values, and a third group of users may be restricted from accessing any values for the field. You can control this access through IBM i Authorization Lists and Powertech Encryption for IBM i's Field Encryption Registry.

Listed below is an example of the steps needed to create Authorization Lists and then associate them to a field in the Field Encryption Registry:

1. Create an IBM i Authorization List to control authority to the full decrypted values for a field.

   EXAMPLE:
   > CRTAUTL AUTL(CCFULL) TEXT('Auth. List of Users with full access')

2. For the CCFULL Authorization List, grant *USE authority only to those users (or user groups) that should have access to the full decrypted values.

   EXAMPLE:
   > EDTAUTL AUTL(CCFULL)

3. Create an Authorization List to control authority to the masked values for a field.

   EXAMPLE:
   > CRTAUTL AUTL(CCMASK) TEXT('Auth. List of Users with masked access')

4. For the CCMASK Authorization List, grant *USE authority only to those users (or user groups) that should have access to the masked values.

   EXAMPLE:
   > EDTAUTL AUTL(CCMASK)

5. When adding a field to the Field Encryption Registry, listed below is an example of how to specify the masking format and Authorization Lists for the field.

   EXAMPLE:
   > ADDFLDENC… FLDMASK('**********9999')
   AUTLDEC(CCFULL) AUTLMASK(CCMASK) NOTAUTHFV('#')

   You can also specify the fill value to use (with the NOTAUTHFV parameter) when the user does not have authority to either Authorization list.

Based on the example settings used above, the field value returned on a decryption request will be one of the following:

- The fully decrypted value, if the user has at least *USE authority to the CCFULL Authorization List.
- Otherwise it will return the masked value, if the user has at least *USE authority to the CCMASK Authorization List.
- Otherwise it will return the fill value if the user does not have at least *USE authority to either the CCFULL or CCMASK Authorization Lists. For instance, if the fill value is #, then a 16 byte field will get a return value of ################

NOTE: The user will additionally need at least *USE authority to the Key Store object which holds the key needed for decryption and *USE authority to the library that contains the Key Store.

If using DB2 Field Procedures, then the authorized values will automatically be returned to the application/user on read operations. Otherwise, APIs in Powertech Encryption for IBM i can be used to access the authorized field values for the user. Read about the GetEncFldAuth Field Decryption API (in the *Programmers Guide*) if you are storing the encrypted field values externally. Read about the DecFldAuth Field Decryption API if you are storing the encrypted field values within the existing database. The GetEncFldAuth and DecFldAuth APIs have corresponding program call APIs, SQL functions and Stored Procedures which can optionally be used from your applications, which are also documented in the *Programmers Guide*. If needed, contact Powertech Support for the *Programmers Guide*.

Library, Object and File Encryption

> **NOTE:**
> BRMS customers: Powertech Encryption for IBM i's backup encryption commands can be incorporated into IBM's BRMS package. Contact Fortra for the BRMS integration instructions.

Users can choose between the encryption algorithms of AES128, AES192 and AES256.

Symmetric Keys or Passwords can be used to protect the encrypted data.

Commands are also provided for restoring and decrypting libraries, objects and files that were encrypted using Powertech Encryption for IBM i's encryption commands.

Powertech Encryption for IBM i's encryption and decryption commands can be entered on the IBM i command line, placed in CL programs, incorporated in BRMS and used in job schedulers on the IBM i.

Testing Restores of Encrypted Backups

It is critical that you periodically test the restoration of your encrypted backups. Test the restoration process when any of the following conditions occur:

1. When you initially use Powertech Encryption for IBM i's ENCxxx commands.
2. If you change any parameter settings on the ENCxxx commands.
3. If you change the key or password used on the ENCxxx commands.
4. If you upgrade the IBM i operating system.
5. If you upgrade the Powertech Encryption for IBM i product to a new version.
6. If you receive any patches or bug fixes for the Powertech Encryption for IBM i product.

Restoring Encrypted Objects Requirements

> WARNING: Fortra will not be able to recover your organization's encrypted data if the password or key is lost.

Passwords

If a password is used for encryption on the ENCxxx commands, keep a copy of the password for disaster recovery purposes. You will need this password to decrypt the data when performing a restore operation with the DECxxx commands.

Record this password in your disaster recovery documentation and/or company safe. At least two people in your organization should know the password value.

Keys

If a key is used for encryption on the ENCxxx commands, back up the key store containing the key, and back up the master key used to encrypt the Key Store. The Key Store and master key need to be available on the system before performing a restore operation with the DECxxx commands. The user profile performing the restore operation must have object authority to the keystore object and DECxxx command, or *ALLOBJ special authority. You do not need to configure the user profile performing a restore operation as a Powertech Encryption IBM i security officer.

Maintain the passphrases needed to recreate a master key in a safe location. See Key Backup and Recovery for more information.

Common Questions about Backup Encryption

Can I encrypt and save Document Library Objects (DLO) ?

Yes. You need to first save DLO into a Save file using IBM's SAVDLO command with the DEV(*SAVF) parameter option. Then you can use Powertech Encryption for IBM i's ENCSAVOBJ (Encrypt Object) command to encrypt/save the Save file to a backup device.

How can I minimize the backup window time?

Listed below are several tips on how you can reduce the amount of time for the encrypted backup processes.

1. You should only encrypt those libraries or objects that contain sensitive data. There is no need to encrypt IBM libraries (e.g. QSYS) or other libraries that do not contain confidential data.
2. The ENCSAVLIB and ENCSAVOBJ commands provided in Powertech Encryption for IBM i allow you to save libraries and objects while active. This allows your users to continue to work in a library while it is being saved.
3. If you have sufficient disk space, you can save each library (that requires encryption) into its own Save File object using IBM's SAVLIB command. When it is convenient, you can then encrypt and save those Save File objects to the backup device using Powertech Encryption for IBM i's ENCSAVOBJ (Encrypt Object) command.

Where can Powertech Encryption for IBM i's encryption commands be used?

The encryption commands can be run from the IBM i command line, CL programs, incorporated in BRMS, or placed in Job Scheduler. If your organization utilizes BRMS, you can contact Fortra for the BRMS integration instructions.

We normally perform a complete backup of our system from IBM's backup menu. How can we still do a complete backup while encrypting certain user libraries?

Instead of using IBM's backup menu to run a full backup, you can instead write a CL program that performs a full backup using a combination of IBM's backup commands (to save system libraries and non-sensitive libraries) and Powertech Encryption for IBM i's backup commands (to encrypt and save sensitive user libraries).

Review the source member named BACKUPALL in the source file CRYPTO/QCLSRC for an example of how to perform a full "partially encrypted" backup.

How would I perform a complete restore onto our Disaster Recovery machine?

This depends on the version of Powertech Encryption you have installed. Please follow the correct instructions below for your installed version.

During a disaster recovery for **Powertech Encryption v3.57 or earlier**:

1. Restore IBM's system libraries, user profiles, authorities and configurations that were saved with the SAVSYS.
2. Restore any unencrypted user libraries that were saved with IBM's SAVLIB command. For example: **RSTLIB SAVLIB(*NONSYS) DEV(TAP01)**.
3. Restore the Powertech Encryption for IBM i licensed program. You may be required to run: **RSTLICPGM(4CRYPTO) DEV(TAP01)**.
4. Restore or recreate the master keys.
5. Restore the keystores (if keys are used to protect your backups).
6. Restore any previously encrypted libraries or objects using Powertech Encryption for IBM i's DECRSTLIB or DECRSTOBJ commands.
7. Restore any previously encrypted IFS files using Powertech Encryption for IBM i's DECSTMF command.

During a disaster recovery for **Powertech Encryption v3.58 or later**:

1. Restore IBM's system libraries, user profiles, authorities and configurations that were saved with the SAVSYS.

2. Restore any unencrypted user libraries that were saved with IBM's SAVLIB command. For example: **RSTLIB SAVLIB(*NONSYS) DEV(TAP01)**.
3. Restore or recreate the master keys.
4. Restore the keystores (if keys are used to protect your backups).
5. Restore any previously encrypted libraries or objects using Powertech Encryption for IBM i's DECRSTLIB or DECRSTOBJ commands.
6. Restore any previously encrypted IFS files using Powertech Encryption for IBM i's DECSTMF command.

See the source member named RESTOREALL in the source file CRYPTO/QCLSRC for an example of a complete restore.

Security Alerts

Security Alerts can be configured to send notifications when any Key Management activities are performed. This can include changes to the Key Policy settings, Key Officer settings, Master Encryption Keys, Data Encryption Keys, Field Encryption Registry entries and Alert settings. Alerts can also be sent when authority errors occur in Powertech Encryption for IBM i, such as when an unauthorized user attempts to access a Key Store.

> **NOTE:** Key Management activities and authority errors will always be logged into Powertech Encryption for IBM i's audit journal file, even if you do not configure any Alerts.

See also Work with Security Alerts (WRKCCALR).

Audit Trails

Product Audit Trails

Powertech Encryption for IBM i includes comprehensive auditing to satisfy the most stringent security requirements. Audit log entries are generated for the following events:

- When any Key Policy settings are changed
- When Key Officers are added, changed or removed
- When Security Alerts are added, changed or deleted
- When Master Encryption Keys (MEKs) are loaded or set
- When Key Stores are created or translated
- When Data Encryption Keys (DEKs) are created, changed, exported or deleted

- When Field Encryption Registry entries are added, changed, copied, removed, activated or deactivated
- When IFS Encryption Registry entries are added, changed, copied, removed, activated or deactivated
- When DEKs are changed or translated for Field Encryption Registry entries
- When SQL triggers are removed or added for Field Encryption Registry entries
- When any functions are denied due to improper authority
- When data is encrypted or decrypted with a key that requires logging of those events

The audit log entries are recorded through the journal named CRJN001, which is located in the CRYPTO library by default. The initial journal receiver is named CRJR001, which is attached to the journal for storing the initial audit entries. The journal uses the option of MNGRCV(*SYSTEM) to allow the system to automatically manage the journal receivers.

Each entry in the journal is assigned an “Entry Type”, which indicates the event that generated the audit log entry. The valid Entry Types are listed below.

Entry Type	Description	Command Issued
01	Key Policy setting(s) changed	CHGKEYPCY
02	Key Officer added	ADDKEYOFR
03	Key Officer changed	CHGKEYOFR
04	Key Officer removed	RMVKEYOFR
05	Master Key passphrase part loaded	LODMSTKEY
06	Master Key was Set	SETMSTKEY
07	Master Key cleared	CLRMSTKEY
08	Key Store created	CRTKEYSTR
09	Key Store translated	TRNKEYSTR
10	Symmetric Key created	CRTSYMKEY
11	Symmetric Key changed	CHGSYMKEY
12	Symmetric Key copied	CPYSYMKEY
13	Symmetric Key deleted	DLTSYMKEY
14	Field Encryption Registry – Entry added	ADDFLDENC
15	Field Encryption Registry – Encryption Key changed	CHGFLDKEY

Entry Type	Description	Command Issued
16	Field Encryption Registry – Entry removed	RMVFLDENC
17	Field Encryption Registry – Entry activated	ACTFLDENC
18	Field Encryption Registry – Entry changed	CHGFLDENC
19	Field Encryption Registry – Entry deactivated	DCTFLDENC
21	Symmetric Key exported	EXPSYMKEY
22	Field Encryption Registry – Unable to Activate Entry	ACTFLDENC
23	Field Encryption Registry – Unable to Deactivate Entry	DCTFLDENC
24	Field Encryption Registry – Entry copied	CPYFLDENC
25	Field Encryption Registry – SQL Triggers added to file	ADDFLDTRG
26	Field Encryption Registry – SQL Triggers removed from file	RMVFLDTRG
27	Field Encryption Registry – Field keys translated	TRNFLDKEY
30	Unable to encrypt/decrypt field using stored procedure	TRIGGER
31	Trigger exit program – Error occurred or return code of 'E'rror	TRIGGER
32	Trigger exit program – Return code of 'I'gnore	TRIGGER
33	Trigger exit program – Return code of 'P'rocess with message	TRIGGER
34	Unable to send Security Alert	
35	Security Alert added	ADDCCALR
36	Security Alert changed	CHGCCALR
37	Security Alert deleted	DLTCCALR
40	Data encrypted with Key that requires logging	
41	Data decrypted with Key that requires logging	
42	External Key Manager – Entry added	ADDEKM
43	External Key Manager – Entry changed	CHGEKM
44	External Key Manager – Entry removed	RMVEKM

Entry Type	Description	Command Issued
50	Authority error	
60	IFS Encryption Registry – Entry added	ADDIFSENC
61	IFS Encryption Registry – Encryption Key changed	CHGIFSKEY
62	IFS Encryption Registry – Entry removed	RMVIFSENC
63	IFS Encryption Registry – Entry activated	ACTIFSENC
64	IFS Encryption Registry – Entry changed	CHGIFSENC
65	IFS Encryption Registry – Entry deactivated	DCTIFSENC
66	IFS Encryption Registry – Unable to Activate Entry	
67	IFS Encryption Registry – Unable to Deactivate Entry	
68	IFS Encryption Failed	
69	IFS Decryption Failed	
70	IFS General Error	
71	IFS Exit Point Program added	
72	IFS Exit Point Program removed	
73	IFS Server Program started	
74	IFS Server Program stopped	
75	IFS Debug Mode was changed	
76	IFS Debug File was cleared	
77	Config File entry added	WRKCONFIG
78	Config File entry was changed	WRKCONFIG

The audit log entries can be printed using the supplied **PRTAUDLOG** command, which is documented on the following page.

You can additionally view the audit log entries in the CRJN001 journal using the DSPJRN command. Listed below is an example of displaying journal entries for any authority errors from June 1st 2008 to June 30th 2008:

DSPJRN JRN(CRYPTO/CRJN001) FROMTIME('06/01/08') TOTIME('06/30/08') ENTTYP (50)

System Audit Trails

The Key Policy settings, Master Keys, Key Officers and Security Alerts are stored in a Validation List (*VLDL) object named CRVL001, which is located in the CRYPTO library by default.

The CRVL001 *VLDL object is protected with the PEK and other internal mechanisms to detect unauthorized changes. However, you may want to additionally place a system audit trail on CRVL001 to track all changes performed to this object.

Follow the steps below to place a system audit trail on CRVL001:

1. Establish system auditing on the CRVL001 object using the following command: CHGOBJAUD OBJ(CRYPTO/CRVL001) OBJTYPE(*VLDL) OBJAUD(*ALL)
2. Make sure that object auditing is enabled at the system level by running the command DSPSYSVAL QAUDCTL. This system value should contain the *OBJAUD special value.
3. If the QAUDCTL system value does not include the *OBJAUD special value, then run the command of CHGSECAUD QAUDCTL(*OBJAUD). This command will set the system value of QAUDCTL to *OBJAUD and will create the audit journal of QSYS/QAUDJRN.

Example of displaying journal entries for any validation list changes from June 5th 2009 to June 11th 2009:

DSPJRN JRN(QAUDJRN) FROMTIME('06/05/09') TOTIME('06/11/09') ENTTYP(VO)

See Print Audit Log (PRTAUDLOG).

External Key Managers

External key managers are solutions that store and allow the creation, modification, deletion, and retrieval of cryptographic keys. You can implement external key managers as software or combined hardware/software solutions. Systems performing encryption or decryption interact with external key managers to remotely create, modify, and delete keys on an external key manager, or retrieve those keys to use for encryption and decryption. External key managers are also referred to as "external keystores."

Powertech Encryption for IBM i supports but does not require the use of an external key manager. You can use external key managers to store data encrypting keys, called

"symmetric keys," in Powertech Encryption for IBM i. The local key store on the IBM i where the symmetric key "resides" contains a reference to the symmetric key.

The most widely used protocol for communication between a system that performs encryption and an external key manager is the KMIP protocol, which Powertech Encryption supports.

You must configure external key managers in Powertech Encryption for IBM i to use one or multiple external key managers from Powertech Encryption for IBM i. The configuration contains information, such as the IP address and protocol used, to interface with the external key store. After you configure and verify the connection to the external key manager, you can create symmetric keys that are stored on that external key manager and function the same as other symmetric keys. You do not need to do additional tasks.

In most scenarios, communication between Powertech Encryption for IBM i and the external key manager will be encrypted at the transport level by the use of TLS. This communication requires set up of TLS in the IBM i's Digital Certificate Manager.

See the following references for more information:

- Appendix C: Adding a Client Certificate to an External Key Manager
- Appendix D: Creating a Certificate using the Digital Certificate Manager (DCM)
- To configure external key stores, see External Key Manager Menu.
- To create symmetric keys on an external key store, see Create Symmetric Key (CRTSYMKEY), the External key manager parameter description.
- The Thales Group is a popular provider of external key managers. To configure Thales external key managers to interface with Powertech Encryption for IBM i, see *Integrating Powertech Encryption for IBM i with Thales Key Management Solutions* on the Fortra Community Portal.

Configuring Multiple Production Environments

Your organization may store production data for multiple companies or divisions on the same system. For each company or division, a unique library (otherwise called an “environment”) may have been created to store the production data for that company or division. The user’s library list is most likely used to control which environment’s library is accessed.

You can establish different Powertech Encryption for IBM i configurations for each environment by placing certain product-specific objects into those environment libraries. Review the following scenarios to learn more.

Environment Scenario #1

In this scenario, your organization may want to have a different Field Encryption Registry for each environment on the system. Your organization may also want to share Powertech Encryption for IBM i's Key Policy settings, Key Officers, Master Keys, Security Alerts and Key Stores across all of those environments.

Follow the steps below to implement this scenario:

1. Make sure that no applications are currently using any Powertech Encryption for IBM i programs or functions.
2. The Field Encryption Registry (contained in the CRVL002 object) cannot be in the CRYPTO library when multiple environments are needed. Therefore, set up the first environment by moving the CRVL002 object from the CRYPTO product library into that environment's library using the command below:
   > MOVOBJ OBJ(CRYPTO/CRVL002) OBJTYPE(*VLDL) TOLIB(datalib1)
3. For each additional environment, you will need to create the Field Encryption Registry (CRVL002 object) into that environment's library using the command below:
   > CRTVLDL VLDL(datalib2/CRVL002) TEXT('Field Encryption Registry')
4. If the physical file option is used to store the last index numbers [LSTINDSTG(*PF) parm on the Registry], then the physical file (named CRPF002) cannot be in the CRYPTO library when multiple environments are needed. Follow these steps to set up CRPF002:
   a. Set up the first environment by moving the CRPF002 file from the CRYPTO product library into that environment's library using this command:
      > MOVOBJ OBJ(CRYPTO/CRPF002) OBJTYPE(*FILE) TOLIB(datalib1)
   b. For each additional environment, you will need to create the CRPF002 physical file (for storing the last index numbers) into that environment's library using the command below:
      > CRTPF FILE(datalib2/CRPF002) SRCFILE(CRYPTO/QDDSSRC)

Then follow the steps below to configure the Field Encryption Registry for each environment:

> TIP: If the field ID has an *INACTIVE status, then you can use the CPYFLDENC command to copy field entries from one production Field Encryption Registry to another.

1. Place the environment's library at the top of the library list:
   > ADDLIBLE LIB(datalib1) POSITION(*FIRST)
2. Configure the Field Encryption Registry for the environment:
   > CRYPTO/WRKFLDENC

The Powertech Encryption for IBM i Field Encryption Registries are now ready for use in the environments. Be sure to place the appropriate environment library in the user's library list in order to use that environment's corresponding Field Encryption Registry.

Environment Scenario #2

In this scenario, your organization may not want to share any Powertech Encryption for IBM i keys or configurations between any environments. In other words, your organization may want to have different Key Policy settings, Key Officers, Master Keys, Security Alerts, Key Stores and Field Encryption Registries for each environment on the system.

Follow the steps below to implement this scenario:

1. Make sure that no applications are currently using any Powertech Encryption for IBM i programs or functions.
2. Certain objects cannot remain in the CRYPTO library for this scenario. Therefore, set up the first environment by moving Powertech Encryption for IBM i's Key Policies, Key Officers and Master Keys (contained in the CRVL001 object), the Field Encryption Registry (contained in the CRVL002 object) and the CRPF002 physical file (for storing the last index numbers used) from the CRYPTO product library into that environment's library using the commands below:

   > MOVOBJ OBJ(CRYPTO/CRVL001) OBJTYPE(*VLDL) TOLIB(datalib1)
   > MOVOBJ OBJ(CRYPTO/CRVL002) OBJTYPE(*VLDL) TOLIB(datalib1)
   > MOVOBJ OBJ(CRYPTO/CRPF002) OBJTYPE(*FILE) TOLIB(datalib1)
3. For each additional environment, you will need to create the CRVL001, CRVL002 and CRPF002 objects into that environment's library by using the commands below:

   > CRTVLDL VLDL(datalib2/CRVL001) TEXT('Settings and Master Keys')
   > CRTVLDL VLDL(datalib2/CRVL002) TEXT('Field Encryption Registry')
   > CRTPF FILE(datalib2/CRPF002) SRCFILE(CRYPTO/QDDSSRC)

Then follow the steps below to configure the Key Policy settings, Key Officers, Master Keys, Security Alerts, Key Stores and Field Encryption Registries for each environment:

1. Place the environment's library at the top of the library list. Example:
   > ADDLIBLE LIB(datalib1) POSITION(*FIRST)
2. Set the Key Policy settings for the environment.
   > CRYPTO/CHGKEYPCY (press F4 to prompt)
3. Establish the Key Officers for the environment.
   > CRYPTO/WRKKEYOFR
4. Load the Master Key passphrases for the environment. Example:
   > CRYPTO/LODMSTKEY MEKID(1) MEKPRT(?) PASSPHRASE(??)

5. Set the Master Key for the environment. Example:
   > CRYPTO/SETMSTKEY MEKID(1)
6. Create a Key Store for the environment. Example:
   > CRYPTO/CRTKEYSTR KEYSTR(datalib1/KEYSTR) MEKID(1) TEXT('Key Store')
7. Set up the Data Encryption Keys in the new Key Store. Example:
   > CRYPTO/WRKSYMKEY KEYSTR(datalib1/KEYSTR)
8. Configure the Security Alerts for the Environment.
   > CRYPTO/WRKCCALR

The Powertech Encryption for IBM i keys and configurations are now ready for use for each environment. Be sure to place the appropriate environment library in the user's library list in order to use that environment's corresponding keys and configurations.

Configuring a Development/Test Environment

Your organization may want to use development environments (libraries) to work with and test field encryption. You can establish different Powertech Encryption for IBM i configurations for each development environment by placing certain product-specific objects into those environment libraries.

Follow the steps below to set up a development environment:

1. Make sure that no applications are currently using any Powertech Encryption for IBM i programs or functions.
2. The Field Encryption Registry (contained in the CRVL002 object) cannot be in the CRYPTO library when multiple environments are needed. Therefore, set up the production environment by moving the CRVL002 object from the CRYPTO product library into that environment's library using the command below:
   > MOVOBJ OBJ(CRYPTO/CRVL002) OBJTYPE(*VLDL) TOLIB(prodlib)
3. For each development environment, you will need to create a Field Encryption Registry (CRVL002 object) into that environment's library using the command below:
   > CRTVLDL VLDL(devlib/CRVL002) TEXT('Field Encryption Registry')
4. If the physical file option is used to store the last index numbers [LSTINDSTG(*PF) parm on the Registry], then the physical file (named CRPF002) cannot be in the CRYPTO library when multiple environments are needed. Follow these steps to set up CRPF002:
   a. Set up the production environment by moving the CRPF002 file from the CRYPTO product library into that environment's library using the command below:
   > MOVOBJ OBJ(CRYPTO/CRPF002) OBJTYPE(*FILE) TOLIB(prodlib)
   b. For each development environment, you will need to create the CRPF002

physical file
(for storing the last index numbers used) into that environment's library using this command:
> CRTPF FILE(devlib/CRPF002) SRCFILE(CRYPTO/QDDSSRC)

5. Copy the field entries from the production Field Encryption Registry to the development environment:
> CRYPTO/CPYFLDENC FRMLIB(prodlib) FRMFLDID(fieldid) TOLIB(devlib)
6. If the test database file does not already exist, then create the test database file into the development environment without copying the data. Example:
> CRTDUPOBJ OBJ(filename) FROMLIB(prodlib) OBJTYPE(*FILE) TOLIB(devlib) DATA(*NO) TRG(*NO)
** Otherwise, clear the test database file before proceeding.
7. Copy the test data from the production file into the development file. Example:
> CPYF FROMFILE(prodlib/filename) TOFILE(devlib/filename) MBROPT(*ADD) FROMRCD(*START) TORCD(100) REPLACE(*YES)
8. If the encrypted values are stored in a separate external file, then perform these additional steps:
   a. If the test external file does not already exist, then create the test external file into the development environment without copying the data. Example:
   > CRTDUPOBJ OBJ(extfilename) FROMLIB(prodlib) OBJTYPE(*FILE) TOLIB(devlib) DATA(*NO)
   ** Otherwise, clear the test external file before proceeding.
   b. If a Logical file is used over the external file, then create the external logical file in test also.
   > CRTDUPOBJ OBJ(extlogicalfilename) FROMLIB(prodlib) OBJTYPE (*FILE) TOLIB(devlib)
   c. Copy the test data from the production external file into the development external file. Only copy those external records which match the index numbers in your test database file.
9. Place the development library at the top of the library list:
> ADDLIBLE LIB(devlib) POSITION(*FIRST)
10. If SQL triggers are used for the Field Identifier, then add the SQL triggers to the development file
using the following command:
> CRYPTO/ADDFLDTRG FLDID(fieldid)

> **NOTE:** The Powertech Encryption for IBM i Field Encryption Registry and file(s) are now ready for use in the development library. Be sure to place the appropriate library in the library list in order to use that library's corresponding Field Encryption Registry and files.

Refreshing Data

If you need to refresh the data in the development library and if the test database file is using SQL triggers to automatically encrypt the values, then you should follow the steps below to refresh the data:

1. Place the development library at the top of the library list.
2. Remove the triggers from the test database file using Powertech Encryption for IBM i's RMVFLDTRG command.
3. Refresh the data in the test database file.
4. If an external file is used to store the encrypted values, then refresh the data in the test external file.
5. Recreate the triggers on the test database file using Powertech Encryption for IBM i's ADDFLDTRG command.

Key Backup and Recovery

Automatic Key Backup (to disk)

Powertech Encryption for IBM i will automatically perform backups into Save Files before specific maintenance activities are performed within the product. This will allow you to recover prior values in case of accidental changes.

Key Stores

Your organization's Data Encryption Keys (DEKs) are contained within Key Stores. The Key Stores are Validation List (*VLDL) objects. The names (and library locations) of these *VLDL objects are those names that were specified on the CRTKEYSTR (Create Key Store) command, unless they were later moved or renamed.

Before a Key Store is translated to another MEK, an automatic backup of the Key Store's *VLDL object is performed into a uniquely named Save File. This Save File will have the name of BACKUPxxxx, where xxxx is a sequential number from 1 to 9999. The Save File will be created in the same library as the *VLDL object.

Master Encryption Keys (MEKs)

The Master Encryption Keys (MEKs) are stored in a Validation List (*VLDL) object named CRVL001, which is stored in the CRYPTO library by default.

Before a Master Encryption Key (MEKs) is set (created), an automatic backup of the CRVL001 *VLDL object is performed into a uniquely named Save File. This Save File will have the name of BACKUPxxxx, where xxxx is a sequential number from 1 to 9999. The Save File will be created in the same library as *VLDL object.

Removing BACKUP save files

The BACKUPxxxx save files should be removed periodically in order to minimize disk usage and to ensure that old keys and data do not remain on the system. These BACKUPxxxx save files should only be removed after you are confident that your applications are working properly.

Backup (to external media)

For disaster recovery purposes, it is critical to have a good plan for recreating the Master Encryption Keys (MEKs) and recovering the Key Stores, Field Encryption Registry and external files (which contain encrypted field values). Failure to do so may result in the inability to decrypt data.

> WARNING: Do not backup the CRYPTO library or these *VLDL objects using Powertech Encryption for IBM i's backup encryption (ENCxxx) commands. Only use IBM's commands for backing up these objects.

Listed below is a summary of the objects you should back up frequently:

- Powertech Encryption for IBM i CRYPTO library
- CRVL001 objects type *VLDL
- CRVL002 objects type *VLDL
- CRPF002 object type *FILE
- User created Key Store objects type *VLDL
- If using external files to store your encrypted database field values, then backup those external files (default object prefix of "CRXX" type *FILE)
- If using IBM i Authorization Lists to secure the Key Store objects or if they are used to secure fields in the Field Encryption Registry, then backup those Authorization Lists.

Listed below is detailed information about the objects to back up:

Backup: Powertech Encryption for IBM i library

The Powertech Encryption for IBM i product should be saved as part of your normal backup processes. This product can be saved with the command SAVLIB LIB(CRYPTO).

Backup: Master Encryption Keys (MEKs) and Policy Settings (CRVL001)

The Key Policy settings, Master Keys, Security Alert settings and Key Officers are stored in a Validation List (*VLDL) object named CRVL001, which is stored in the CRYPTO library by default.

> WARNING: The passphrase parts used to load a MEK should be recorded in a safe place off of the System i. All passphrase parts will be needed to recreate a MEK in a disaster recovery situation if installing onto a different IBM i serial number.
>
> The passphrases for a MEK are case-sensitive and must be entered in the same order.

The CRVL001 *VLDL object should be saved as part of your normal backup processes.

The values in the CRVL001 *VLDL object are encrypted with the Product Encryption Key (PEK). The PEK is partially derived from the IBM i serial number. Therefore each IBM i machine has its own unique PEK.

If the CRVL001 *VLDL object is restored onto a different IBM i serial number, then it will not be usable since the PEK will be different. In that case, the Key Policy settings, Master Keys, Security Alerts and Key Officers will need to be manually recreated (see the disaster recovery section).

Backup: Field Encryption Registry (CRVL002)

The Field Encryption Registry is stored in a Validation List (*VLDL) object named CRVL002, which is stored in the CRYPTO library by default (unless other environments were set up). CRVL002 contains important information about the fields that are registered for encryption, such as field names, pointers to Key Labels used to encrypt/decrypt data, index numbers for externally stored encrypted field data, etc.

The CRVL002 *VLDL object should be saved as part of your normal backup processes.

Backup: Last Index Numbers Used (CRPF002)

If you utilize the Field Encryption Registry and are using external files to store encrypted database field values and if the ‘last index numbers used’ are stored in the physical file option [LSTINDSTG(*PF) parm on the Registry], then the CRPF002 physical file should be saved as part of your normal backup processes. This file is stored in the CRYPTO library by default (unless other environments were set up).

Backup: External Files containing Encrypted Values (CRXX*)

If you utilize the Field Encryption Registry and are using external files to store encrypted database field values, then save those external files as part of your normal backup processes. You can find the names of those external files by executing the command of CRYPTO/WRKFLDENC and placing an option 5 next to each field entry to display.

Backup: Key Stores (*VLDL object types)

Your organization's Data Encryption Keys (DEKs) are contained within Key Stores. The Key Stores are Validation List (*VLDL) objects. The names (and library locations) of these *VLDL objects are those names that were specified on the CRTKEYSTR (Create Key Store) command, unless they were later moved or renamed.

The Key Store *VLDL objects should be saved as part of your normal backup processes. Since each Key Store is encrypted with a Master Key (which is encrypted by a PEK), the Key Stores will be protected on your backup media from unauthorized usage.

Backup: Authorization Lists

If you use Authorization Lists to secure your Key Stores or to secure fields in the Field Registry, then save those Authorization Lists as part of your normal backup processes. The Authorization Lists can be saved using the IBM i command of SAVSECDTA or SAVSYS.

Backup: License Keys

Powertech Encryption for IBM i license keys are stored and saved in the CRYPTO library when you save a copy of the library. See instructions below for if you restore the product on the same or to a different system:

- If you restore the product on the same system, you do not need a new key.
- If you restore the product to a different system or another LPAR not included in your license, first contact Fortra Technical Support to get a license key for the new system. You can then add this key in the menu CRYPTO10, option 1. When you add this key prior to restoring the new system, the product continues functioning.

Disaster Recovery

Follow the steps below to restore the Powertech Encryption for IBM i product, user settings, Master Keys and Key Stores in a disaster recovery situation.

WARNING:

- IBM objects, user profiles and authorization lists should be restored before proceeding.
- As of IBM i 7.5, modified validation list objects cannot be saved to previous releases. Powertech Encryption cannot be replicated or moved from a system running IBM i 7.5 to a system running a lower version of IBM i.

If restoring to the same IBM i serial number:

1. Locate the Powertech Encryption for IBM i licensed program on your backup media. If it is not available on the backup media, you can download the product from www.fortra.com.
2. Restore the user-created Key Stores from your backup media. The Key Stores are Validation List (*VLDL) objects.
3. Restore the CRVL001 *VLDL object from your backup media. This holds the Key Policy, Key Officers, Security Alerts and Master Encryption Keys. By default, the CRVL001 object is located in the CRYPTO library (unless other environments were set up).
4. If you use the Field Encryption Registry in Powertech Encryption for IBM i, restore the CRVL002 *VLDL object from your backup media. By default, this object is located in the CRYPTO library (unless other environments were set up).
5. If you use the Field Encryption Registry and are using external files to store encrypted database field values and if the 'last index numbers used' are stored in the physical file option [LSTINDSTG(*PF) parm on the Registry], then restore the file named CRPF002 from your backup media. By default, the CRPF002 file is located in the CRYPTO library (unless other environments were set up).
6. If using external files to store the encrypted database field values, then restore those files (default prefix of CRXX).

NOTE: If using IFS Encryption, contact Fortra Support for additional steps.

If restoring to a different IBM i serial number:

1. Locate the Powertech Encryption for IBM i licensed program on your backup media. If it is not available on the backup media, you can download the product from www.fortra.com.
2. Follow step 3 only if you did not previously obtain and back up a license key, as defined in the Backup (to external media), Backup: License Keys section. Otherwise, skip to step 4.

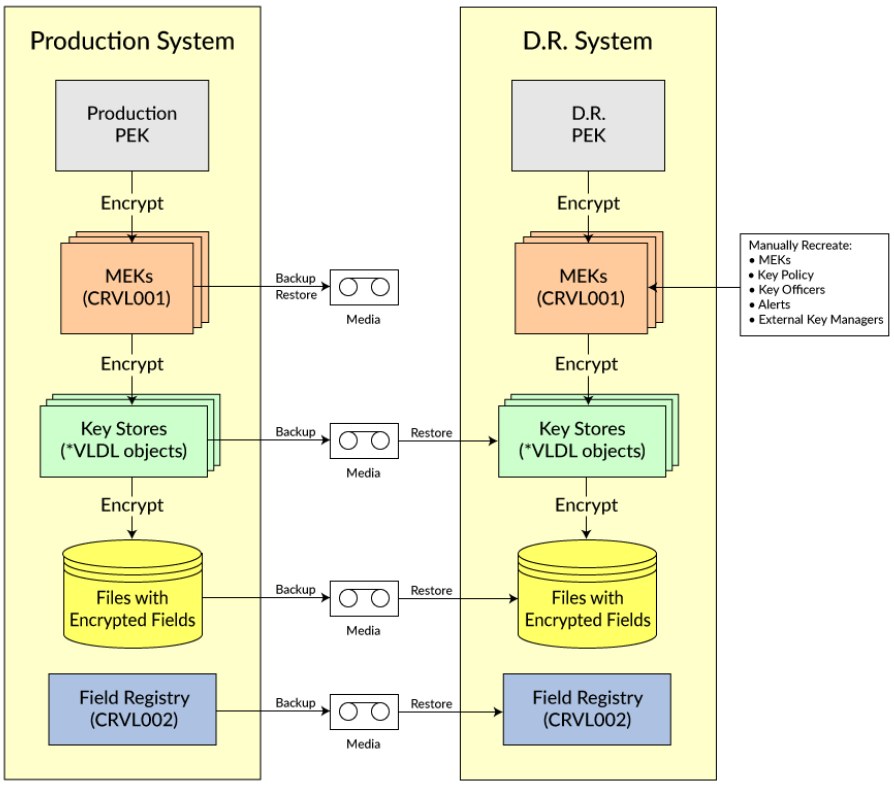
3. If needed, contact Fortra Technical Support to obtain a usable license key for the IBM i machine you are restoring to.
4. Restore the user-created Key Stores from your backup media. The Key Stores are Validation List (*VLDL) objects.
5. If you use the Field Encryption Registry in Powertech Encryption for IBM i, restore the CRVL002 *VLDL object from your backup media. By default, the CRVL002 object is located in the CRYPTO library (unless other environments were set up).

   > **NOTE:** The CRVL001 *VLDL object should not be restored if the IBM i Serial number changed, because CRVL001 was encrypted under a different Product Encryption Key (PEK) on the other machine.

6. If you use the Field Encryption Registry and are using external files to store encrypted database field values and if the 'last index numbers used' are stored in the physical file option [LSTINDSTG(*PF) parm on the Registry], then restore the file named CRPF002 from your backup media. By default, the CRPF002 file is located in the CRYPTO library (unless other environments were set up).
7. If using external files to store the encrypted database field values, then restore those files (default prefix of CRXX).
8. Reestablish the Key Policy settings using the CHGKEYPCY command.
9. Reestablish the Security Alert settings using the WRKCCALR command.
10. Reestablish the Key Officers using the WRKKEYOFR command.
11. Enter the passphrase parts for each Master Encryption Key (MEK) using the LODMSTKEY (Load Master Key) command. These passphrase parts must be entered exactly as they were on the original system using the same part numbers. Passphrases are case sensitive.
12. Generate (set) each Master Encryption Key (MEK) using the SETMSTKEY command.
13. The Key Stores, and the Keys stored within them, should then be available for use.

> **NOTE:** If using IFS Encryption, contact Fortra Support for additional steps.

Disaster Recovery Diagram using Field Procedures



High Availability Setup

Follow the steps below to set up Powertech Encryption for IBM i on a High Availability (HA) system and to replicate the Production settings and related user data properly.

> WARNING: As of IBM i 7.5, modified validation list objects cannot be saved to previous releases. Powertech Encryption cannot be replicated or moved from a system running IBM i 7.5 to a system running a lower version of IBM i.

Initial setup:

Install Powertech Encryption for IBM i onto the HA system. See the *Powertech Encryption Installation Guide*.

A license key for Powertech Encryption for IBM i is required on the HA system.

Manual replication:

Powertech Encryption for IBM i's Key Policy settings, Key Officer settings, Security Alert settings and Master Keys are stored in a Validation List (*VLDL) object named CRVL001. This CRVL001 *VLDL object is encrypted with a Product Encryption Key (PEK), which is unique to each IBM i serial number. Therefore, any changes to those settings on the Production system will need to be manually applied to the HA system using Powertech Encryption for IBM i's commands and screens.

> WARNING: DO NOT replicate the CRVL001 *VLDL object from the Production system to the HA system. Settings in CRVL001 must be manually configured using Powertech Encryption for IBM i's screens.

Follow the instructions below to manually apply those settings on the HA system:

Key Policy Settings

If Powertech Encryption for IBM i's Key Policy settings are changed on the Production system, then those Key Policy settings need to be also changed on the HA system by following these steps:

1. On the Production system, display the Key Policy settings using the command CRYPTO/DSPKEYPCY and record the values displayed.
2. On the HA system, change the Key Policy settings using the command CRYPTO/CHGKEYPCY using the recorded values from the Production system.

Key Officers

If Powertech Encryption for IBM i's Key Officers are added, changed or deleted on the Production system, then those changes need to be applied on the HA system by following these steps:

1. On the Production system, display the Key Officer settings using the command CRYPTO/WRKKEYOFR and record the values displayed.
2. On the HA system, configure the Key Officers by using the command CRYPTO/WRKKEYOFR using the recorded values from the Production system.

Security Alerts

If Powertech Encryption for IBM i's Security Alerts are added, changed or deleted on the Production system, then those changes need to be applied on the HA system by following these steps:

1. On the Production system, display the Security Alert settings using the command CRYPTO/WRKCCALR and record the values displayed.

2. On the HA system, configure the Security Alerts by using the command CRYPTO/WRKCCALR using the recorded values from the Production system.

Master Keys

If Powertech Encryption for IBM i's Master Keys are Set on the Production system, then those Master Keys must also be Set on the HA system by following these steps:

1. On the HA system, enter the passphrase parts for each Master Encryption Key (MEK) using the LODMSTKEY (Load Master Key) command. These passphrase parts must be entered exactly as they were on the Production system.
2. On the HA system, generate (set) each Master Encryption Key (MEK) using the SETMSTKEY (Set Master Key) command.

Automatic replication:

Configure your High Availability (HA) product to replicate the following objects on an ongoing basis:

1. Your user-created Key Stores are created as Validation list (*VLDL) objects. The names (and library locations) of these *VLDL objects are those names that were specified on the CRTKEYSTR (Create Key Store) command, unless they were later moved or renamed. These *VLDL objects should be replicated by your HA product from the Production system to the HA system.
2. Replicate the CRVL002 *VLDL object from the Production system to the HA system. This object contains the Field Encryption Registry. To avoid issues on the initial replication, ensure that CRVL002 Keystores, and any related Authorization Lists have been replicated to the Target HA system prior to any encrypted data arriving. To properly encrypt and decrypt data, the CRVL002 validation list object (and Authorization Lists) will be referenced as any encrypted data files are being replicated over to the Target system.

> NOTE: Immediate replication of CRVL002 (object type *VLDL) can cause replication challenges if all of the following are true:
>
> - you use external files (instead of field procedures);
> - you have a high-volume environment with many new inserts occurring; and
> - the 'Last Index Number' is stored in CRVL002 (instead of the PF choice).

3. If you use the Field Encryption Registry and are using external files to store encrypted database field values and if the 'last index numbers used' are stored in the physical file option [LSTINDSTG(*PF) parm on the Registry], then replicate the physical file named CRPF002.

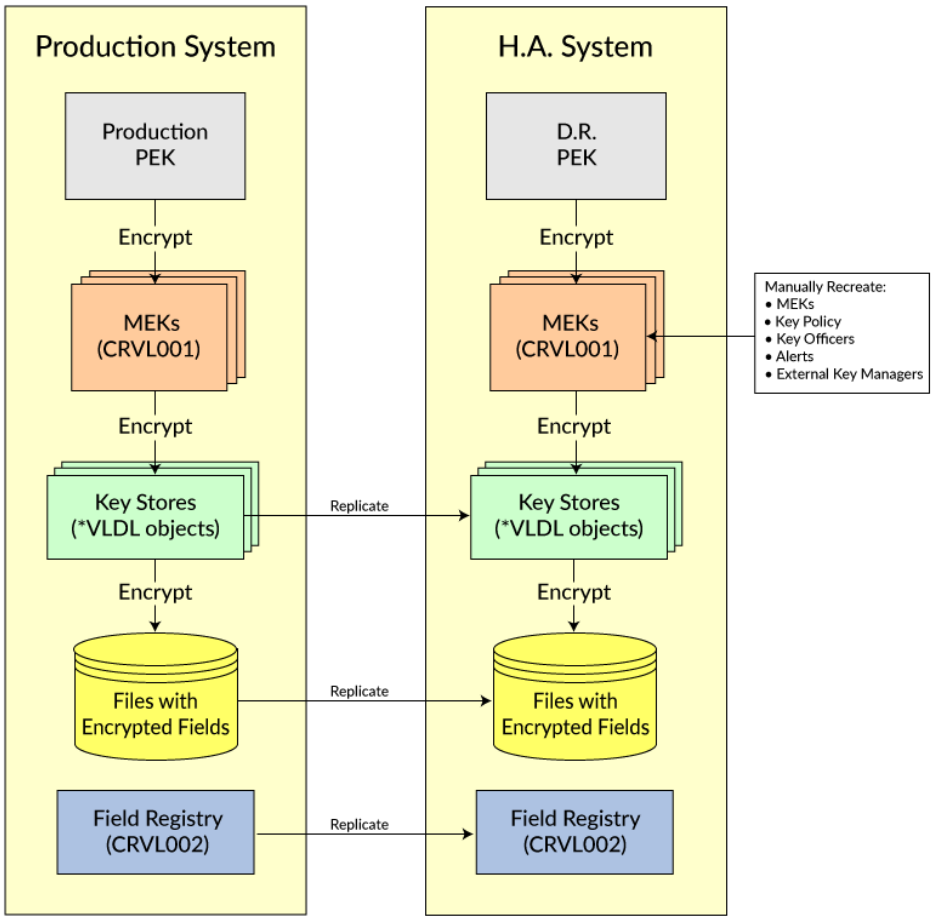
4. Replicate any IBM i Authorization Lists that may be used to secure Key Stores or to secure fields in the Field Encryption Registry. Add your replication software service profile to the full authorization object (*AUTL) used in the field encryption registry for each field in the files being replicated.

   > **NOTE:** When using a replication product for automatic replication, the user profile running the Apply Job must be in the 'Full' authorization list that apply to any encrypted files being replicated.

5. Replicate the Powertech Encryption for IBM i authorization lists PCRADMIN and PCRREPORT.
6. DO NOT replicate the CRVL001 *VLDL object from the Production system to the HA system. The settings in CRVL001 are encrypted with the Product Encryption Key (PEK), which is unique per IBM i serial number.
7. Field encryption is normally implemented using field procedures. However, in a legacy environment, field encryption entries may be using external files. In that case, replicate the external files from the Production system to the HA system. You can find the names of those external files by running the CRYPTO/WRKFLDENC command and placing an option 5 next to each *ACTIVE field entry.
8. If you are using Field Procedures and the Decryption Accelerator (*YES) for any Field Registry Entries, upon initial replication using the Sync feature (not save/restore), perform the following command on the Target HA system before doing the initial sync: **CHGFCNUSG FCNID(QIBM_DB_SECADM) USER(APPLYJOBUSERID) USAGE (*ALLOWED).**
   This command needs to be executed by a user with SECADM or SECOFR authority. For more information, see Appendix G: Decryption Accelerator Prerequisites and Limitations or contact Powertech Technical Support.

> **NOTE:** If you are using IFS Encryption, contact Fortra Support for additional steps.

High Availability Diagram



Verifying CRVL001 Settings

The Verify CRVL001 (VFYCRVL001) command allows you to verify that the CRVL001 object is valid for your installation. This should generally only need to be executed after a system restore.

The CRVL001 object contains your organization's key policy settings, security alert settings, key officers and master keys.

If the VFYCRVL001 command fails, it is most likely because the CRVL001 object was copied from another machine (with a different serial number).

You should NOT copy the CRVL001 *VLDL object between machines with different serial numbers since the settings in CRVL001 are encrypted with the Product Encryption Key (PEK), which is unique per IBM i serial number.

Working with Key Stores

You can use the following procedures to view and manage Key Stores:

To translate a Key Store

> **NOTE:** It is highly recommended to execute TRNKEYSTR immediately after the SETMSTKEY command was used to replace the *CURRENT version of a Master Encryption Key with a *NEW version.

> **IMPORTANT:** Before the Key Store is translated, the Validation List (*VLDL) object that contains the Key Store is backed up into a Save File object (sequentially named) within the Powertech Encryption for IBM i library.

1. Prompt (F4) the command of **CRYPTO/TRNKEYSTR**. The Translate Key Store (TRNKEYSTR) panel appears.
2. Press F1 on any parameter for complete online help text.
3. Press Enter after the parameter values are entered.

> **IMPORTANT:** After executing the TRNKEYSTR command, you should verify that the Key verification values (KEYVV) match between the Key Store and the Master Key by viewing those values with the DSPKEYSTR and DSPMSTKEY commands.

To display Key Store attributes

The DSPKEYSTR command allows authorized users to display the attributes for a Key Store. This is primarily useful for viewing the Master Encryption Key (MEK) id number and version in which the Key Store entries are encrypted with.

Do the following steps to view a Key Store's attributes:

1. Prompt (F4) the command of **CRYPTO/DSPKEYSTR**. The Display Key Store Attributes (DSPKEYSTR) panel appears.
2. Enter the Key Store name to display, and then press Enter.
3. The Key Store's attributes will be displayed.
4. Press F1 on any parameter for complete online help text.

> **NOTE:** The Master Key's Key verification value (KEYVV) value is stored with each Key Store created using that Master Key. When a Key Store is accessed by a user or application, Powertech Encryption for IBM i will compare the KEYVV values between the Key Store and its corresponding Master Key. If the KEYVV values match, then the Master Key is determined as valid for the Key Store.

To delete a Key Store

Since a Key Store is created as a validation list (*VLDL) object on the IBM i, you can delete a Key Store by using IBM's DLTVLDL (Delete Validation List) command.

To delete the Key Store, the user must have authority to the DLTVLDL command and must have *OBJEXIST rights to the Validation List object.

> WARNING: Do not delete a Key Store which may contain Data Encryption Keys (DEKs) that are needed to decrypt existing data.

> WARNING: Back up the Key Store before deleting it.

Do the following steps to delete a Key Store:

1. Backup the Validation List (*VLDL) object to backup media or to a Save File object.
2. Prompt (F4) the command of DLTVLDL.
3. Specify the Key Store name and library, and then press Enter.

Editing the Authority on a Key Store

To edit the authority on a Key Store, you must have authority to the EDTOBJAUT command and must have *OBJMGT rights to the Validation List object.

Do the following steps to edit the authority on a Validation List object that contains a Key Store:

1. Enter the command of **EDTOBJAUT OBJ(library / vldlist) OBJTYPE(*VLDL)**, where library is the name of the library that contains the Validation List and vldlist is the name of the Key Store Validation List.
2. Specify the authorities for the object.
3. Press Enter after the authorities are entered.

Authority recommendations for Key Store Validation List (*VLDL) objects:

- Grant *PUBLIC *USE authority. Also ensure that *PUBLIC has at least *USE authority to the library that contains the Key Store.
- Grant *CHANGE authority only to those users (Key Officers) who are allowed to create new Data Encryption Keys (DEKs) into the Key Store.

> NOTE: If a user attempts to access an unauthorized Key Store through Powertech Encryption for IBM i's screens or APIs, that authority error will be logged into an audit file.

For a complete discussion regarding using Key Store Authority and Authorization Lists to control encryption and decryption, see Controlling Access to Decrypted Values.

Questions and Answers

Listed below are common questions and answers for Powertech Encryption for IBM i.

Can I use an encrypted field as a key on the file (for lookups/chains)?

Yes. If using a DB2 Field Procedure for the field encryption, the operating system will automatically encrypt the lookup value and use that encrypted value to perform the search. No change should need to be made to the application.

If not using a DB2 Field Procedure, you can lookup a database record that is keyed by an encrypted field, but you first have to encrypt the lookup value. For instance, after the user enters the lookup value on the screen, you can encrypt that value (using a Powertech Encryption for IBM i API) and then use the encrypted value to chain out to the file. The programming approach to use will depend on whether the encrypted values are stored within your existing field or are stored in a separate external file. Look at the source member examples of CHAININT and CHAINEXT in the source file CRYPTO/QRPGLESRC.

Are SQL Triggers saved with the Physical File object?

Yes. When setting up an entry in the Field Encryption Registry, you can specify whether SQL triggers should be used to automatically encrypt data when records are added and/or field values are changed. When you activate a field entry in the Registry, the triggers will be created by Powertech Encryption for IBM i over your Physical file using the SQL “CREATE TRIGGER” command.

Unlike traditional external Triggers, SQL Triggers are saved with the Physical file object when you perform a SAVLIB, SAVOBJ or SAVCHGOBJ command. Therefore, you do not need to backup the SQL Triggers separately.

Will Powertech Encryption for IBM i’s SQL Triggers interfere with other Triggers that are already on the file?

SQL Triggers can be placed over a physical file even if there are already existing triggers on the file. When setting up an entry in the Field Encryption Registry, you can either specify the names of the SQL triggers or use the *GEN option to have Powertech Encryption for IBM i generate the trigger names. For the *GEN option, the naming convention used for the SQL Triggers is:

- "FILENAME_FIELDNAME_CryptoInsert" for the Insert Trigger
- "FILENAME_FIELDNAME_CryptoUpdate" for the Update Trigger
- "FILENAME_FIELDNAME_CryptoInsert" for the Delete Trigger

FILENAME is the name of the Physical file and FIELDNAME is the name of the field in the Physical file.

The triggers are created to run as BEFORE triggers, meaning that they run before the actual insert, update or delete of the database record. The insert and update triggers will change the value of the database field (on which the triggers are based) to contain either an encrypted value (when external files are not used) or to contain an index number (when the encrypted value is stored externally).

How will encrypted data be shown on the screens?

Encrypted data is made up of numbers, letters and special characters. Listed below is an example of text that was encrypted using the AES256 algorithm:

> Before: "The quick brown fox jumped over the lazy dog"
> After:"„Œ \ËKä°BBY ý\åê·Ñ,C‹Ÿ^{F+rÀJ[1]Ï(¾Y½i›"®t"

These encrypted values may contain end-of-field delimiters or other special characters, which confuses 5250 terminals and emulators. This data may cause emulators to exhibit strange behavior or lock up.

If you are storing encrypted values in the existing database field space, then you should refrain from showing those encrypted values on 5250 screens. Modify the program if you do not want it to show the encrypted value. Otherwise, you can call one of Powertech Encryption for IBM i's APIs to decrypt the field value and show it on the screen (if the user is authorized).

If you are storing encrypted values in an external file, then the user will see the numeric index number of the external file's record. This may be acceptable to your organization to show this index number. If not, then you can either change the application program to not display the index number OR call the GetEncFld procedure to retrieve and decrypt the field value, which you can then show on the screen (if the user is authorized).

Can I rotate the keys at any time?

Yes. If you use the Field Encryption Registry and choose to store the encrypted field values in an external file, then a Key Identifier is stored with each record in the external file. If a key is changed for a field, then a new key identifier will be stored with any records that are

inserted or updated in the external file. A mass re-encryption of the field values will not be needed.

When retrieving an encrypted value from the external file, Powertech Encryption for IBM i will use the external file record's key identifier to retrieve the correct Key Label for decrypting the value. This technique allows you to change the key for a field in the Registry at any time, even when users are active in the application. Up to 99,999 keys can be rotated for a field.

What kind of maintenance can I perform on a database file that has SQL Triggers attached?

1. The operating system will not allow you to perform a CLRPFM (Clear Physical File Member) command on your database file if it has a Delete trigger on it. A Delete trigger will be on the file if you chose to store the encrypted field values in an external file when defining it in the Field Encryption Registry. If you want to remove all the records from the file, you should perform a mass delete through a programming language or use a SQL Delete command.
2. Do not rename or move a database file if one or more of its fields are activated in the Field Encryption Registry. In order to keep the Field Encryption Registry "in sync" with the new file location, you should follow these steps:
    a. Deactivate the field in the Field Encryption Registry.
    b. Rename or move the database file object.
    c. Change the entry in the Field Encryption Registry to the new file location.
    d. Activate the field in the Field Encryption Registry.
3. Do not perform a CRTDUPOBJ (Create Duplicate Object) command with TRG(*YES) specified. This action would duplicate the triggers to the new object, which would confuse the Field Encryption Registry and the encryption processes. Data may be lost.

Can I encrypt field values for just certain records in a database file?

There are two approaches which you can utilize to only encrypt the values for certain selected records:

Approach #1:

a. Create a logical file with select/omit criteria to choose the records (in the physical) which need to be encrypted.
b. When adding the field to the registry with the ADDFLDENC command, specify the name of that logical file for the DBFILE parameter.

c. When you “activate” the field in the registry using the ACTFLDENC command, it will mass encrypt only the field values for the records selected by the logical file.
d. You will not be able to use triggers to automatically encrypt the values since triggers are not allowed on logical files. Instead, you will need to use the Powertech Encryption for IBM i field encryption APIs (e.g. EncFld, InsEncFld, UpdEncFld, etc.) to encrypt the values, which are documented in the Powertech Encryption for IBM i Programmer’s Guide.

Approach #2:

a. Instead of using the field registry and a logical file, use the EncAES* APIs within your application to encrypt the field values for only the records that meet your application’s selection criteria. These APIs are documented in the Powertech Encryption for IBM i Programmer’s Guide.
b. To decrypt the values, use the DecAES* procedures.

Encryption Terminology

Listed below are the primary encryption terms used throughout this manual.

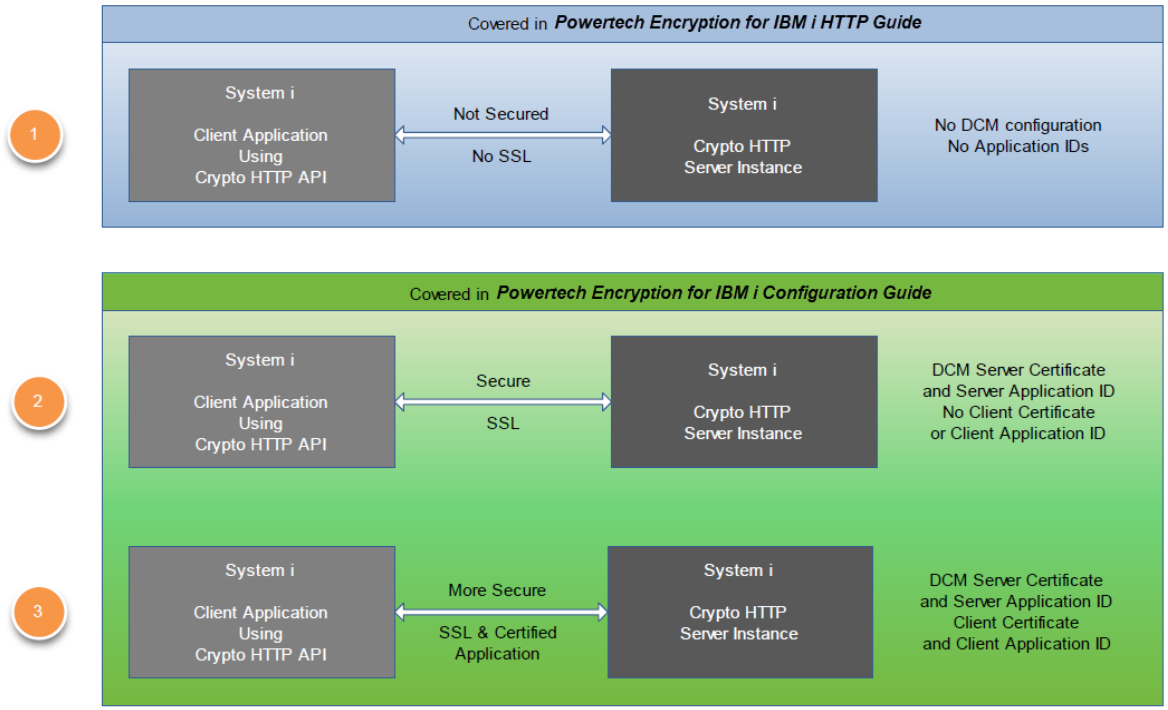
AES	AES is the abbreviation for Advanced Encryption Standard. AES is an encryption algorithm which utilizes symmetric keys. It provides strong protection and is approved by the U.S. Government for protecting sensitive information. See http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf for more information on the AES encryption standard.
AES128	AES encryption using a key length of 128 bits.
AES192	AES encryption using a key length of 192 bits.
AES256	AES encryption using a key length of 256 bits.
Algorithm	A mathematical process used to scramble (encrypt) data.
Data Encryption Key (DEK)	A symmetric key used to encrypt and decrypt data.
CBC mode	CBC is the abbreviation for Cipher-Block Chaining. CBC mode is available in the AES and TDES algorithms. With CBC mode, you can alter the encryption algorithm by supplying an Initialization Vector (IV) value. Therefore, the same input Plain Text and Key can produce different output Cipher Text values, depending on the IV supplied.
Cipher	A pair of algorithms (mathematical processes) used to encrypt and decrypt data.

Cipher Text	The unintelligible (encrypted) text value generated (output) by an encryption algorithm.
Cryptology	The art and science of keeping data secret.
CUSP mode	CUSP is the abbreviation for Cryptographic Unit Support Program. CUSP mode is available in the AES algorithm. CUSP mode is a special type of CBC mode documented in the z/OS ICSF Application Programmer's Guide (SA22-7522). It is used for handling data that is not a multiple of the block length. The length of output Cipher Text in CUSP mode will always equal the length of the input Plain Text. With CUSP mode, you can alter the encryption algorithm by supplying an Initialization Vector (IV) value. Therefore, the same input Plain Text and Key can produce different output Cipher Text values, depending on the IV supplied.
Decryption	The process of converting Cipher Text (unintelligible code) into Plain Text (readable information).
ECB mode	ECB is the abbreviation for Electronic Codebook mode. ECB mode is available in the AES and TDES algorithms. You cannot use Initialization Vectors (IV) with ECB mode, so the same input Plain Text and Key will always produce the same output Cipher Text value.
Encryption	The process of converting Plain Text (readable information) into Cipher Text (unintelligible code).
Hash	An algorithm for calculating a value based on a block of data. If the data changes in any way, then the hash values will not match when it is recalculated. A hash will protect the integrity of data.
Initialization Vector (IV)	An additional value that can be supplied to alter the encryption algorithm in order to produce a different result. In other words, the same input Plain Text and Key can produce different output Cipher Text values, depending on the IV supplied. This is especially useful to ensure the security of small encrypted values.
Key	The information needed to control the detailed operations of the Cipher. In contrast to human-generated passwords, Keys are more secure since they are computer-generated and are represented as an obscure series of bits (1001110…).
Key Store	An object used to organize and store one or more keys.
Master Encryption Key (MEK)	A key used to protect (encrypt) other keys.
Passphrase	Alternative name for password. A string of characters (entered by the user or supplied by a program) that can be used to create a Key.

Plain Text	The readable (decrypted) text value generated (output) by a decryption algorithm.
Symmetric Key	A key which can be used to encrypt and decrypt data. The key must be kept secret or the security is compromised.
TDES	TDES is the abbreviation for Triple DES (Data Encryption Standard). TDES is an encryption algorithm which utilizes symmetric keys. TDES is slowly disappearing from use since AES is up to 6 times faster and offers higher protection than TDES.
Tokenization	Tokenization is the process of replacing sensitive data with unique identification numbers (e.g. tokens) and storing the original data on a central server, typically in encrypted form.

Welcome to the Powertech Encryption for IBM i DCM Configuration Guide

To make use of the Powertech Encryption for IBM i HTTP APIs, it is necessary to configure and run an HTTP Server Instance. Find instructions on how to do configure and run in the Powertech Encryption for IBM i HTTP Guide. The APIs can be used with or without SSL enabled. This Powertech Encryption for IBM i DCM Configuration Guide provides the IBM Digital Certificate Manager tasks necessary to secure the connection between the HTTP APIs and the HTTP server they communicate with. The following diagram illustrates the three main choices for levels of security and where they are covered:



Single IBM i or Multiple System i

The Powertech Encryption for IBM i HTTP APIs can be used in an application to communicate with a Powertech Encryption for IBM i HTTP server instance on the same IBM i system. In this scenario, it is not necessary (though possible) to use SSL. This would be a legitimate scenario for using option 1 in the previous diagram. The remainder of this document addresses situations where multiple IBM i are to be used.

Public or Private Certificates

The choice of using public certificates or private certificates affects the configuration tasks on the client system for option 3 above. If a public certificate is used, then it is not necessary to load it on the client system. This is because the Digital Certificate Manager is pre-loaded with credentials from the major public certificate issuers. If a private certificate is used (generated on the server system), then it typically needs to be loaded into DCM on the client system before the SSL connection will work.

For option 2, the server certificate can be public or private without affecting the client configuration.

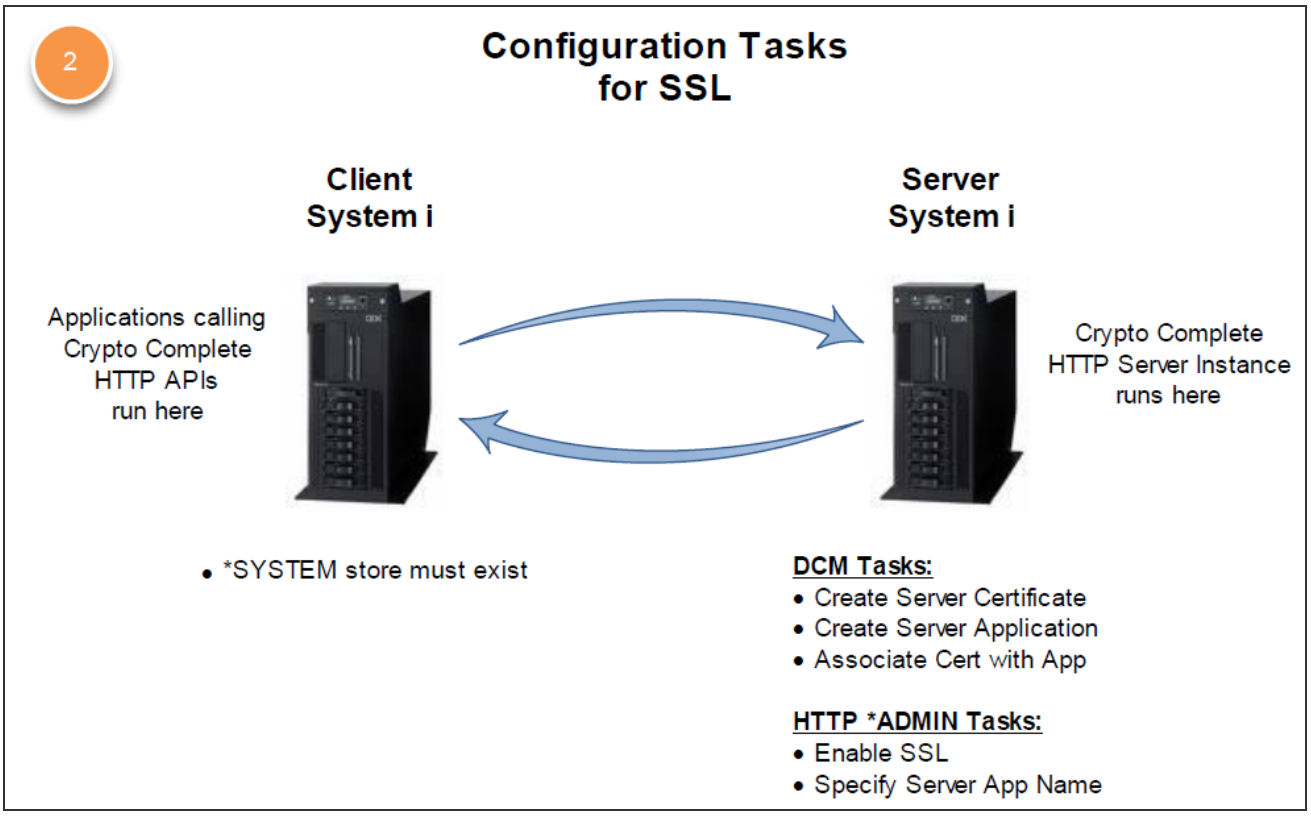
> NOTE: Make sure that the Powertech Encryption for IBM i HTTP server instance is configured with “Do not request client certificate for connection” on the SSL with Certificate Authentication tab in the Security properties. This is done in the IBM Web Administration for iSeries application.

Configuration Tasks

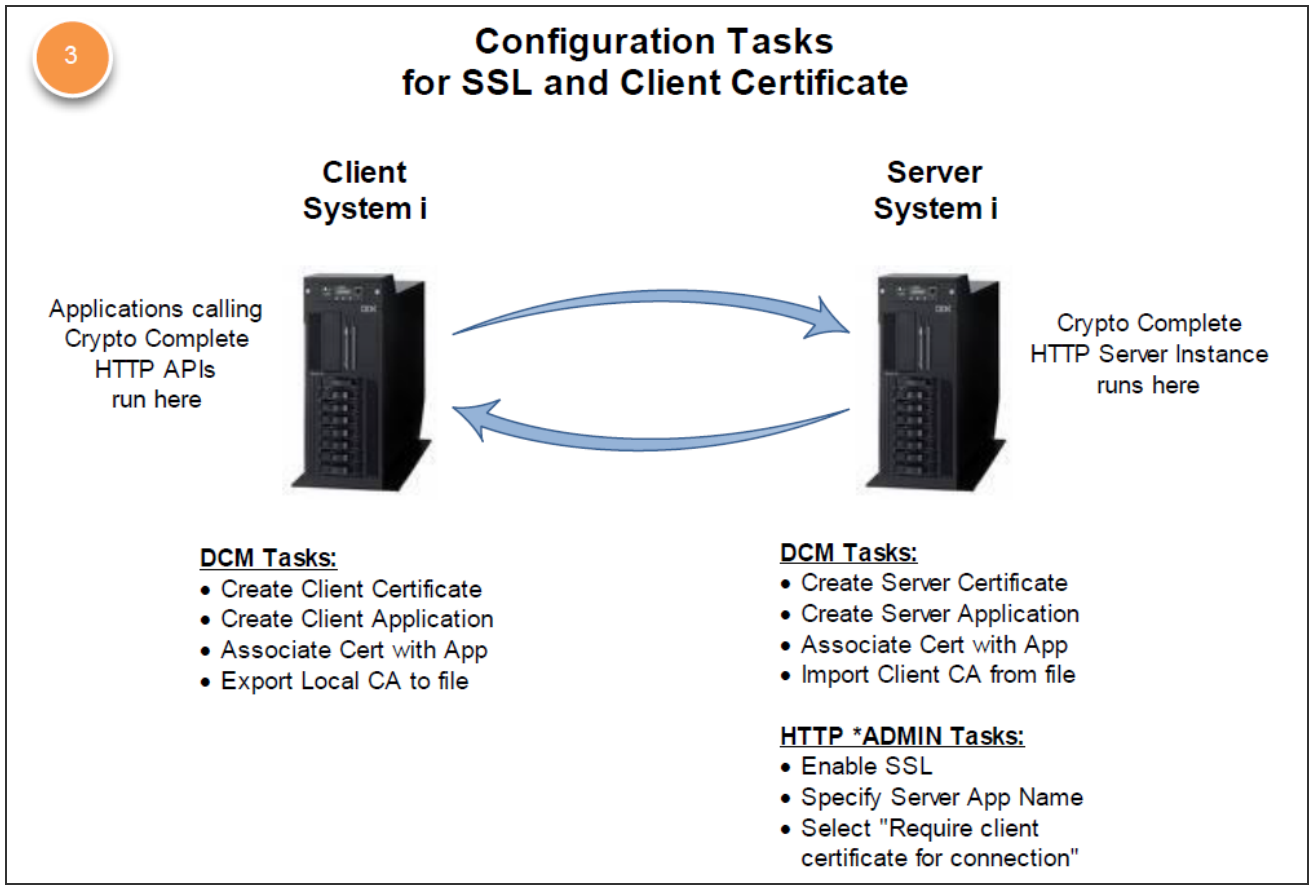
Configuration choices 2 and 3 in the previous diagram will result in the Secure Sockets Layer (SSL) to be used to encrypt the stream of data traveling between the application using Powertech Encryption for IBM i HTTP API’s and the Powertech Encryption for IBM i HTTP server instance.

There are configuration tasks that will need to be done on both IBM i systems in order for Secure HTTP to be enabled between Powertech Encryption for IBM i HTTP APIs on one IBM i (client) to the Powertech Encryption for IBM i HTTP Server instance on a second IBM i (server). Discussing two IBM i systems as client and server can be confusing. The following diagrams will help to clarify the tasks to be performed when configuring SSL for use with Powertech Encryption for IBM i.

The following diagram shows the Digital Certificate Manager tasks that need to be performed to begin using SSL. The HTTP *ADMIN task are covered in the Powertech Encryption for IBM i HTTP Guide.



The following diagram shows more DCM tasks for requiring a client certificate.



The IBM i Administration web server instance must be running in order to work with the Digital Certificate Manager.

Use IBM Web Administration for i

The IBM Digital Certificate Manager (DCM) is a web-based application that is accessible from the Administration Server on System i. You can start the Administration Server from the server's function in iSeries Navigator or you can enter the following command:

STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)

Once the administration server is running, you open your browser and enter this URL:

http://*yourSystemi*:2001/ QIBM/ICSS/Cert/Admin/qycucm1.ndm/ main 0

> **NOTE:** Replace 'yourSystemi' with the name or IP address of your system.

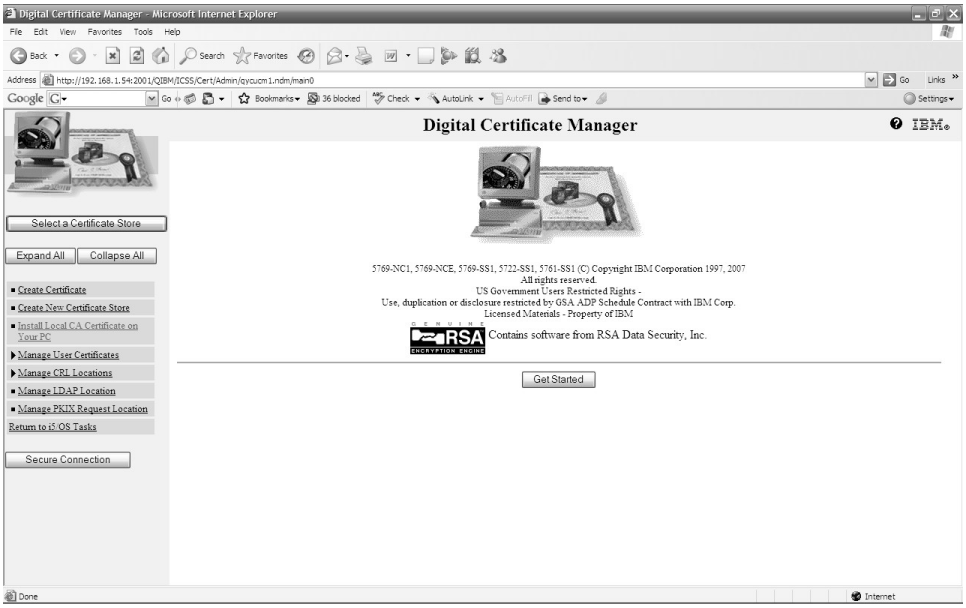
You will be prompted for a user profile and password. The user profile used must have sufficient authority to use DCM.

WARNING: You will need to keep track of whether you are making changes to the server IBM i or the client IBM i *ADMIN server.

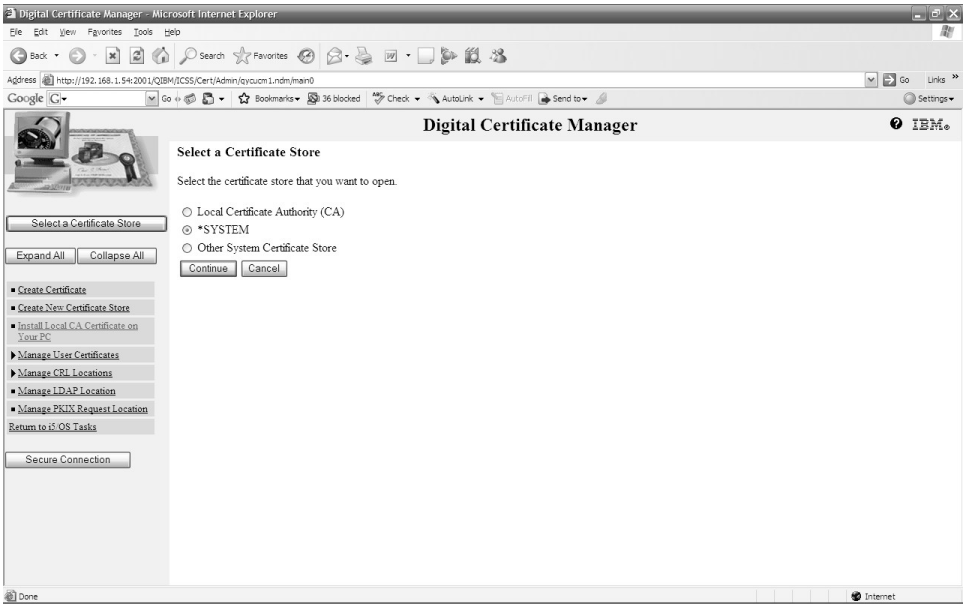
The Basic SSL Configuration

Regardless of whether you will be using option 2 or 3, it is necessary to create a digital certificate and associate it with the Powertech Encryption for IBM i HTTP Server in order to use SSL. This section provides details on how to do this.

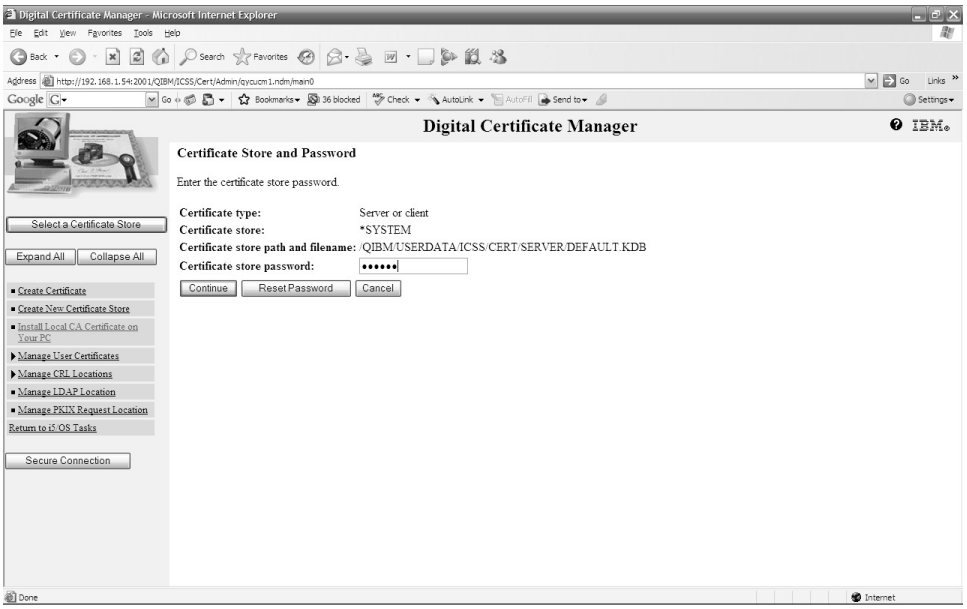
1. Upon signing into the ***server*** IBM i Administration server, the main screen is shown:



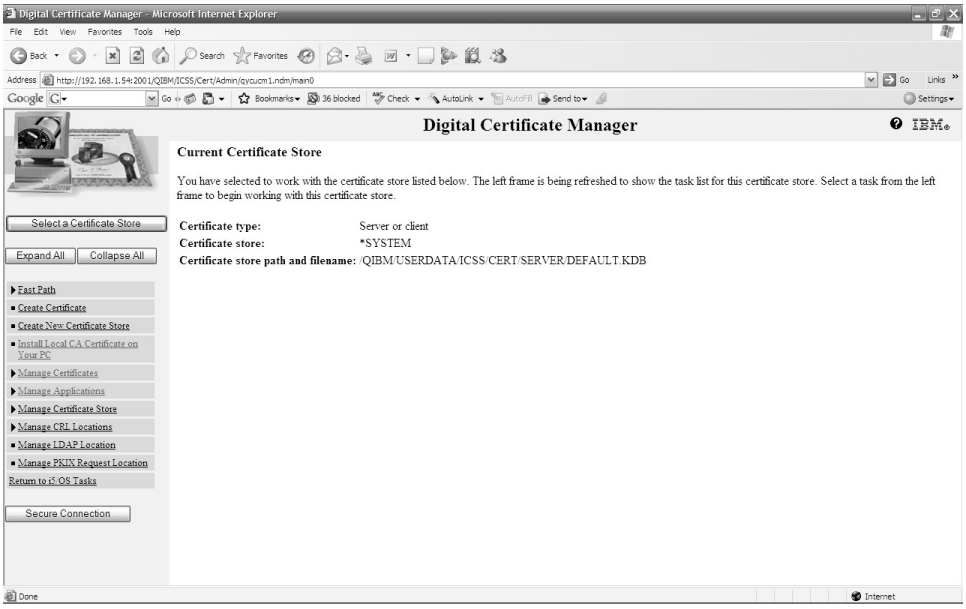
2. Click **Select a Certificate Store**.



3. Click on the *SYSTEM radio button, then click **Continue**.

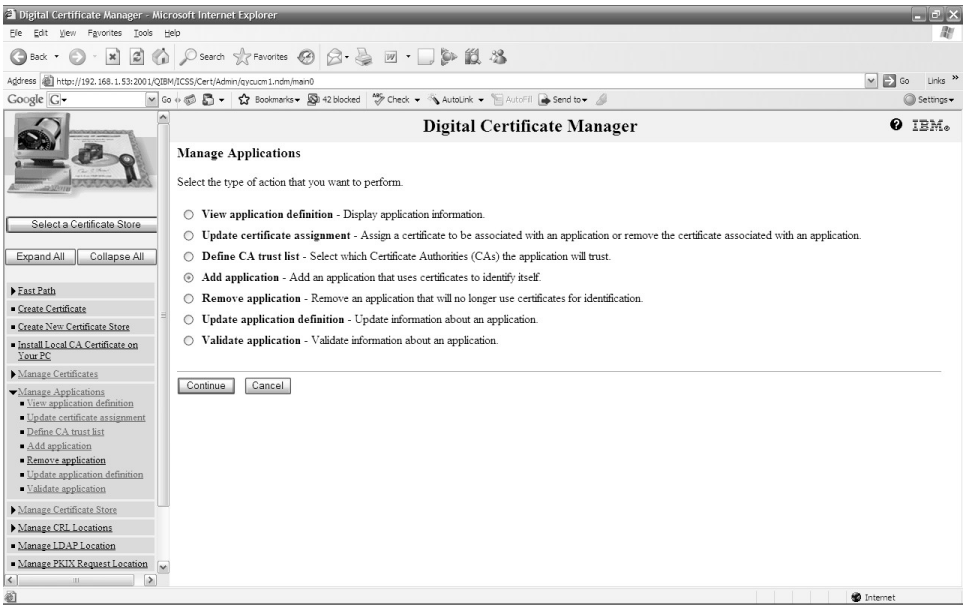


4. Enter the Certificate Store password and click **Continue**. If you do not know the password check with your system administrator.

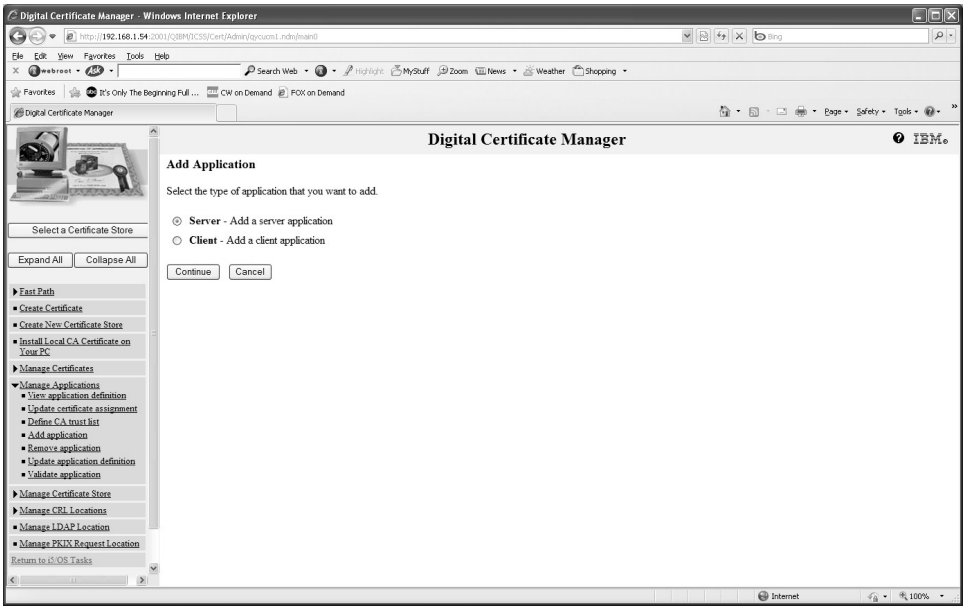


Create an Application

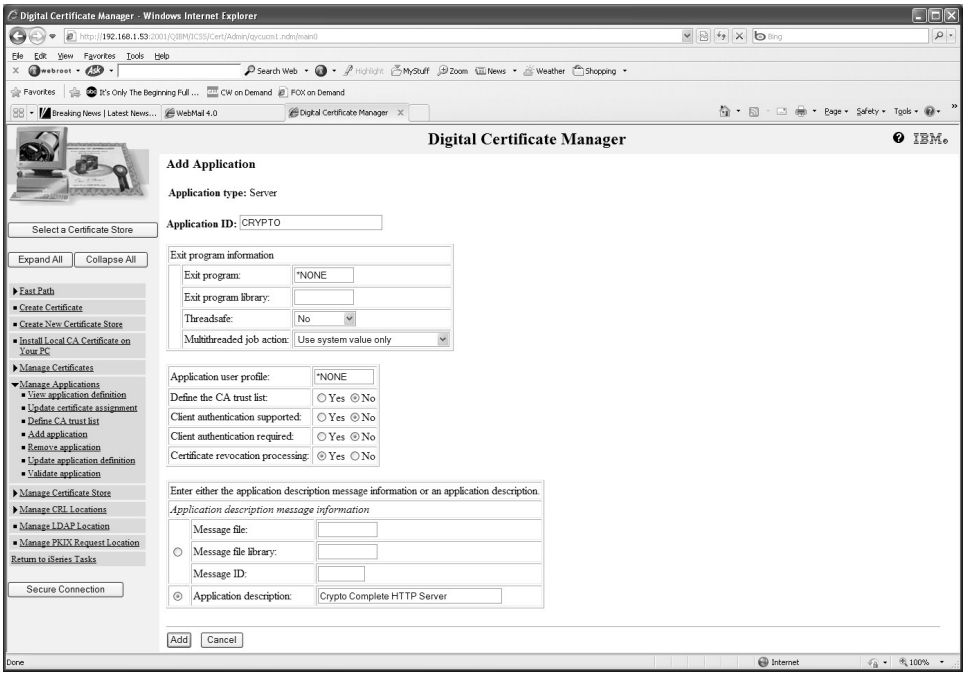
1. To add an application, click on the Manage Applications link in the list on the left side of the screen to expand the list.



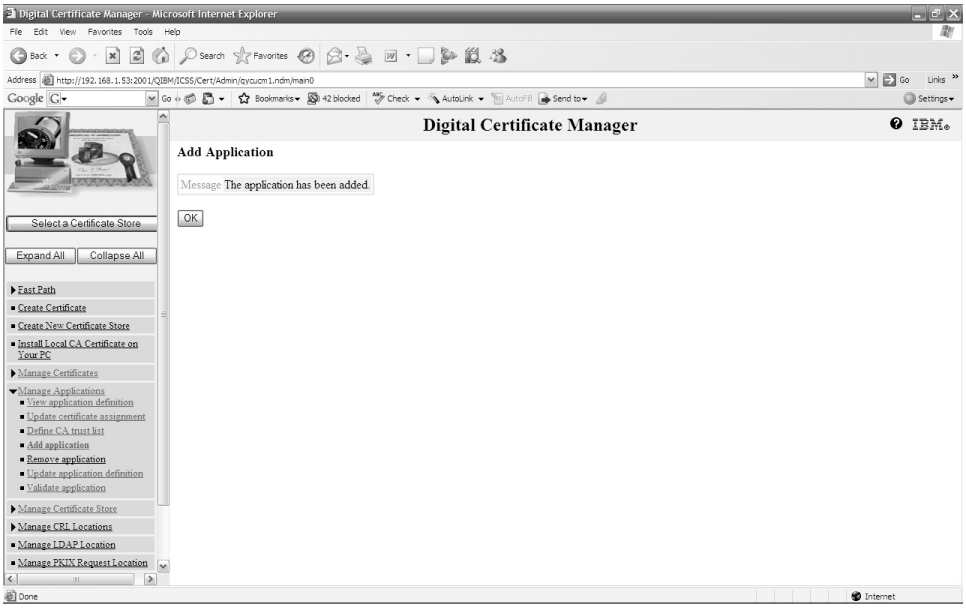
2. Click on the Add application radio button in the center area of the screen and click **Continue**.



3. Click on the Server radio button, then click **Continue**.
4. Enter the values shown on the following screen:



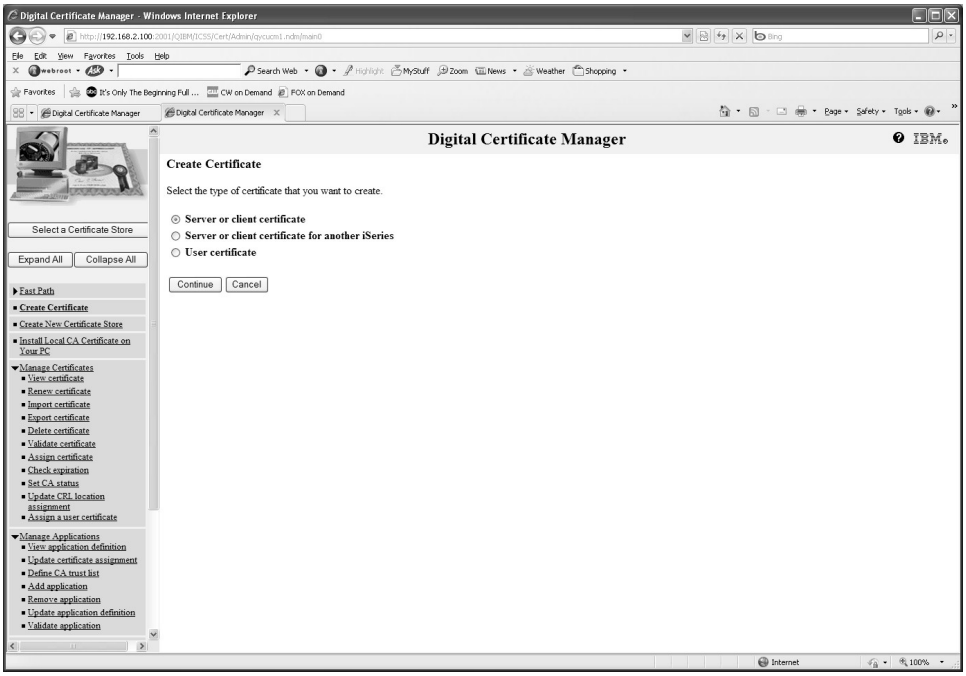
5. Click **Add** to add the entry. You should see the following screen:



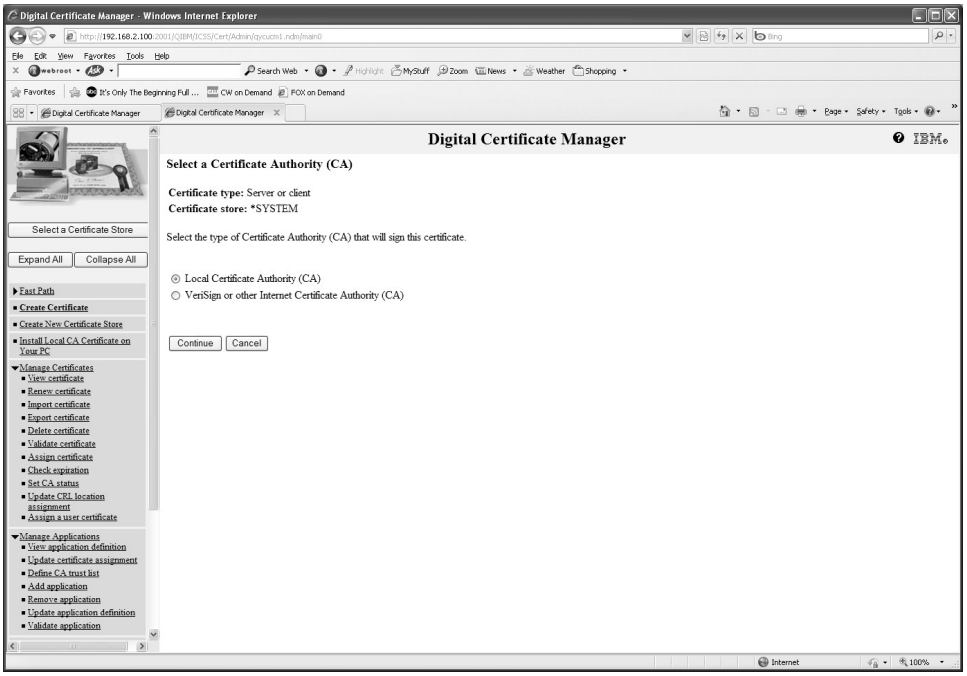
6. Click **OK**. The application has been added.

Create a Server Certificate

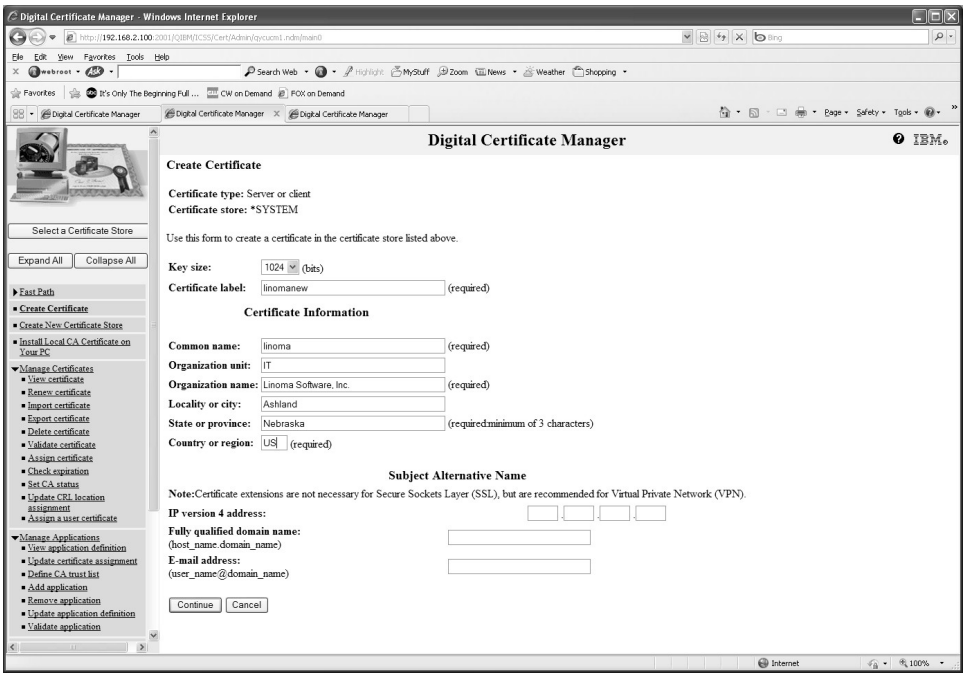
1. To create a server certificate, click on the Create Certificate link on the left side of the screen.



2. Select the Server or client certificate radio button, then click **Continue**.



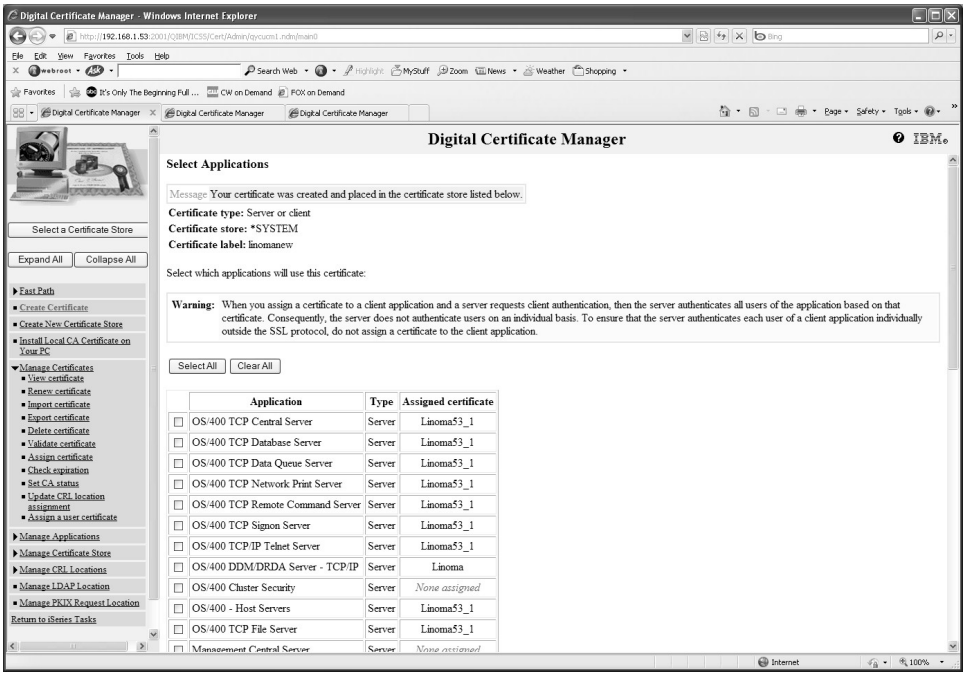
3. Select the Local Certificate Authority (CA) radio button, then click **Continue**.



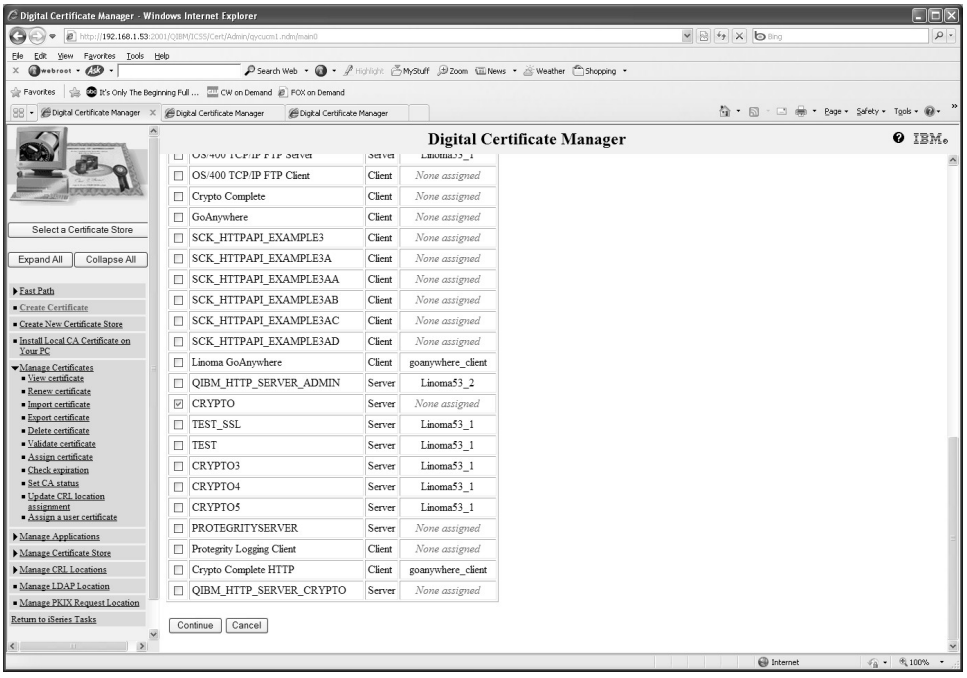
4. Fill in values that are appropriate for your business as shown above, then click **Continue**.

Associate the Certificate with the Application

1. Once a certificate is created, the Digital Certificate Manager will display a list to allow you to select the applications to be associated with it.



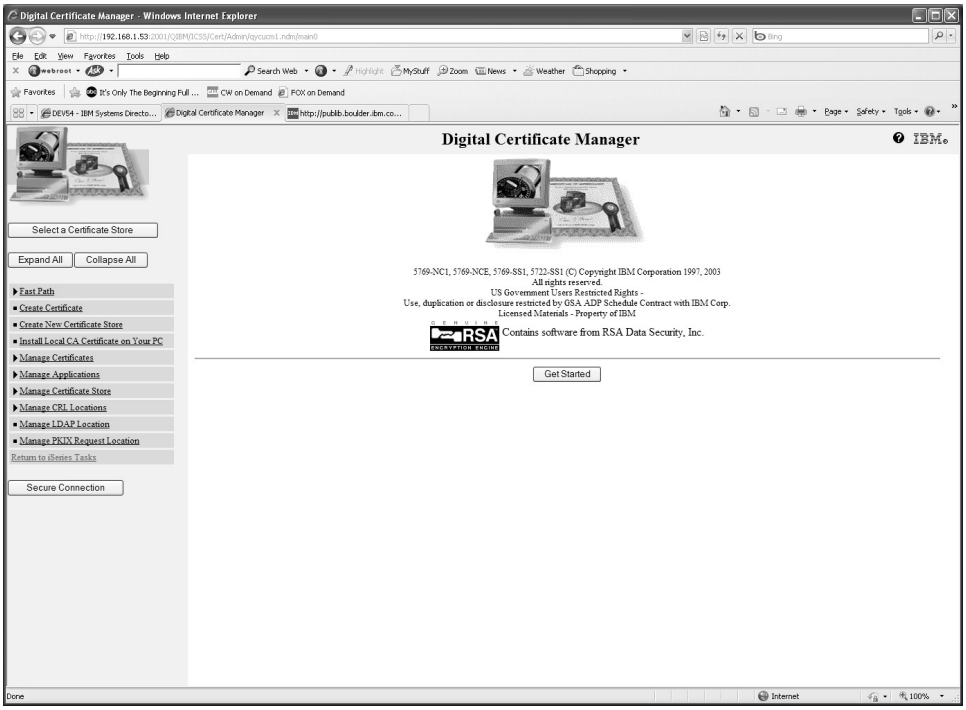
2. Scroll down through the list of applications until you find the application created earlier (i.e. CRYPTO).



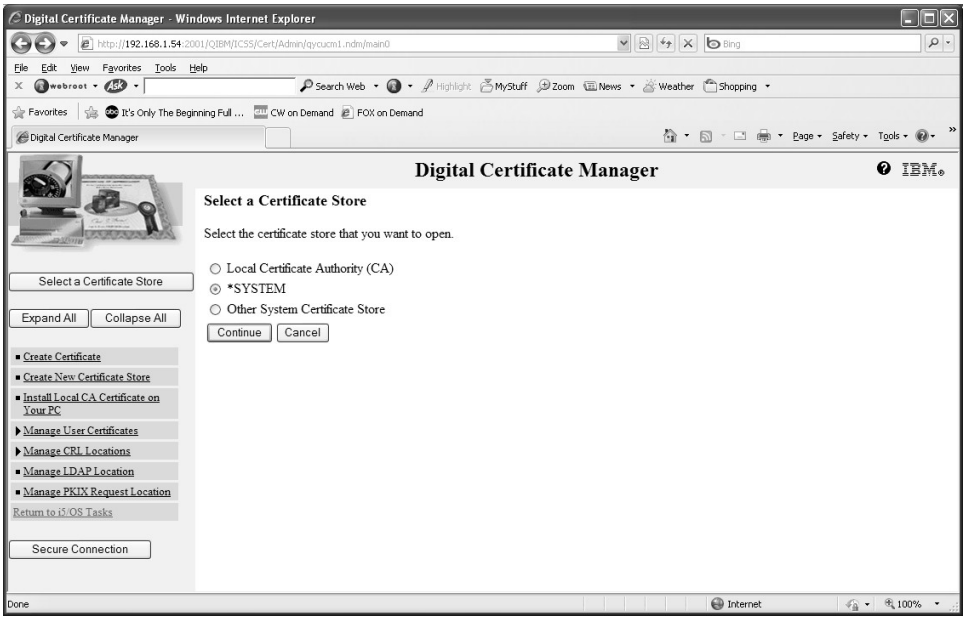
3. Click on the check box to the left of CRYPTO to select it, then click on.
4. Now the Server Certificate has been created and associated with the application created earlier.
5. If configuration option 2 will be used, then the configuration tasks are complete. You are now ready to use the Powertech Encryption for IBM i HTTP APIs from the client System i. Do not specify an application ID when using the APIs with a Basic SSL configuration.
6. If configuration option 3 will be used, continue with the following section.

The SSL With Client Certificate Configuration

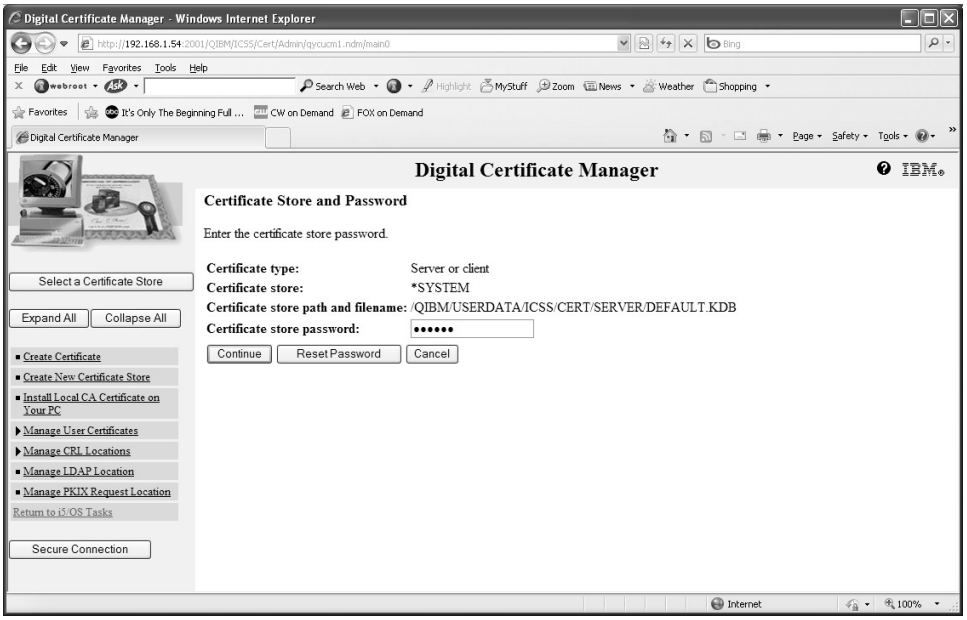
1. Upon signing into the client IBM i Administration server, the main screen is shown:



2. Click **Select a Certificate Store**.



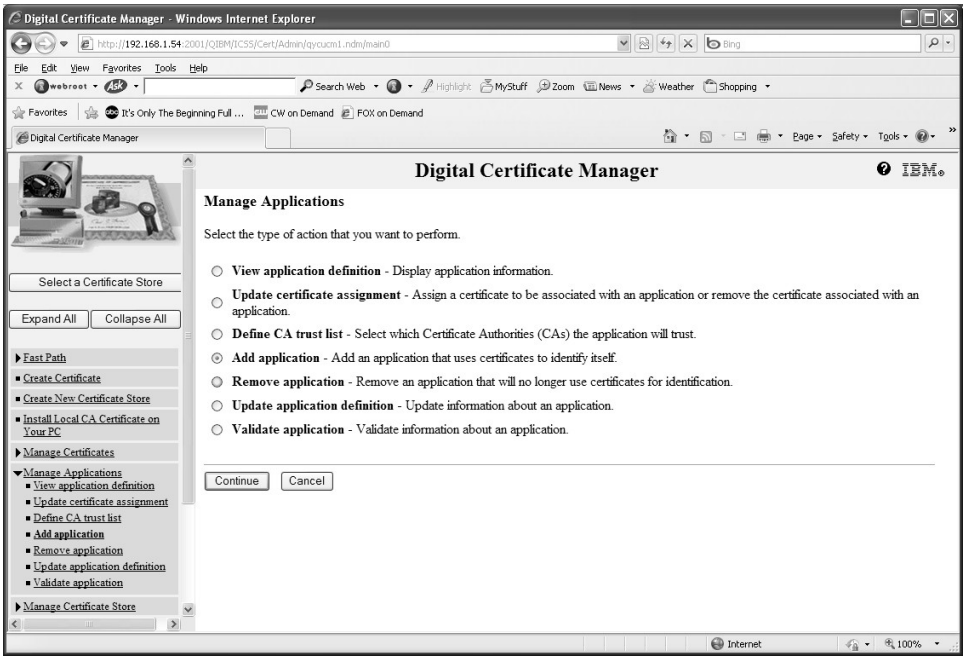
3. Select the *SYSTEM radio button, then click **Continue**.



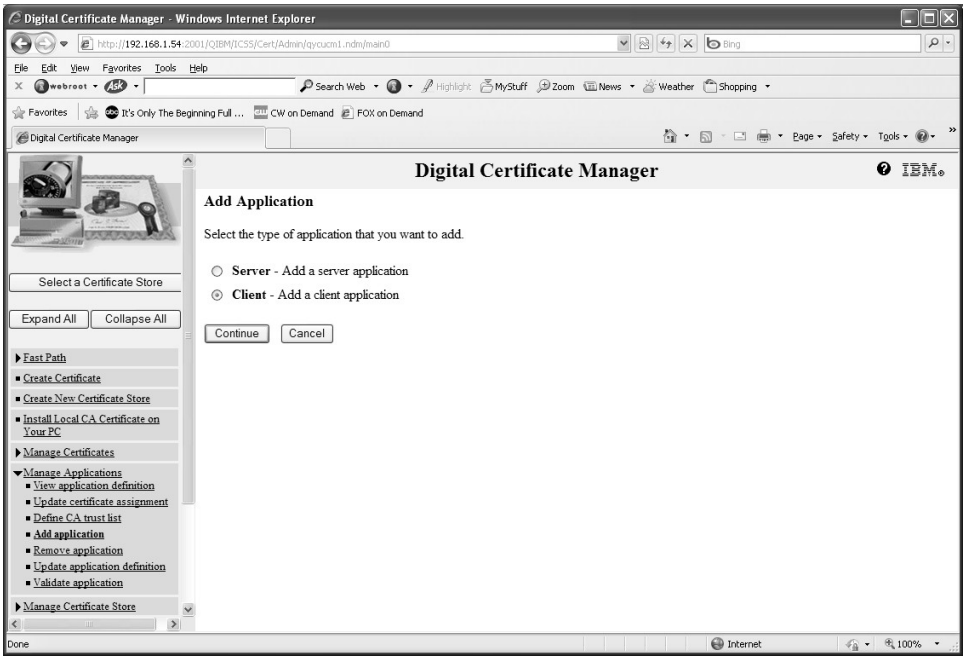
4. Enter the *SYSTEM Certificate Store password and click **Continue**. If you do not know the password check with your system administrator.

Create a Client Application

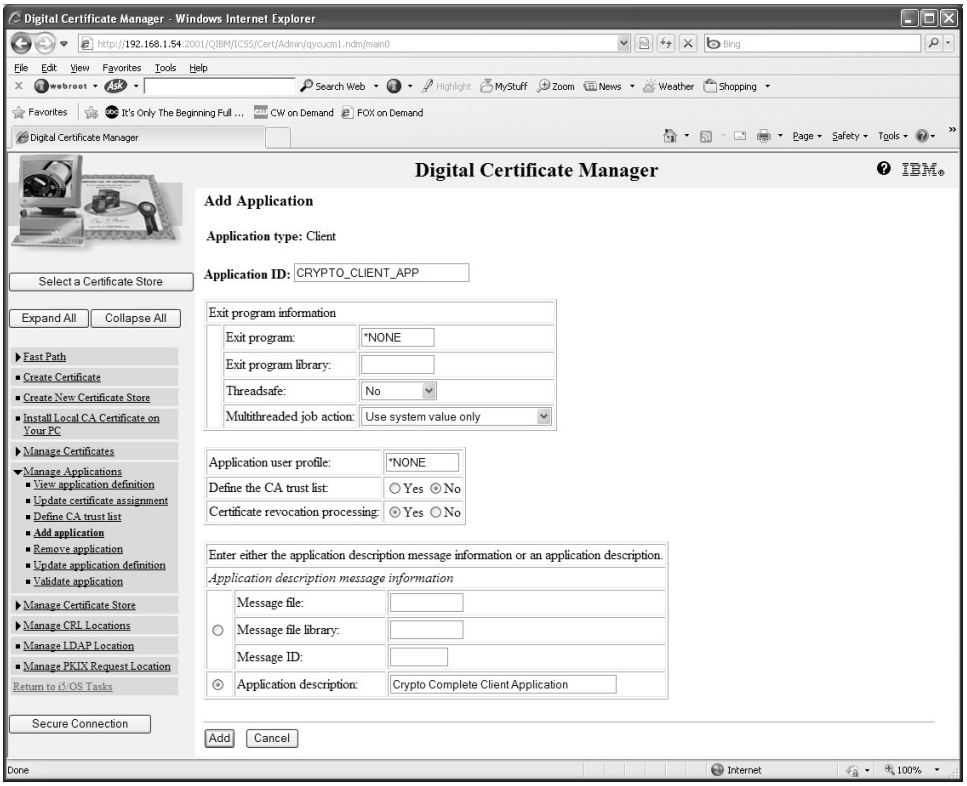
1. To add an application, click on the Manage Applications link in the list on the left side of the screen to expand the list.



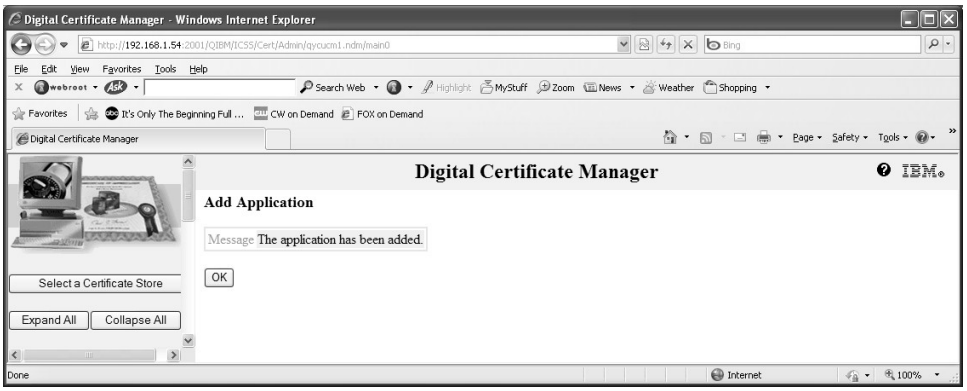
2. Click on the Add application radio button in the center area of the screen and click **Continue**.



3. Select the Client radio button, then click **Continue**.



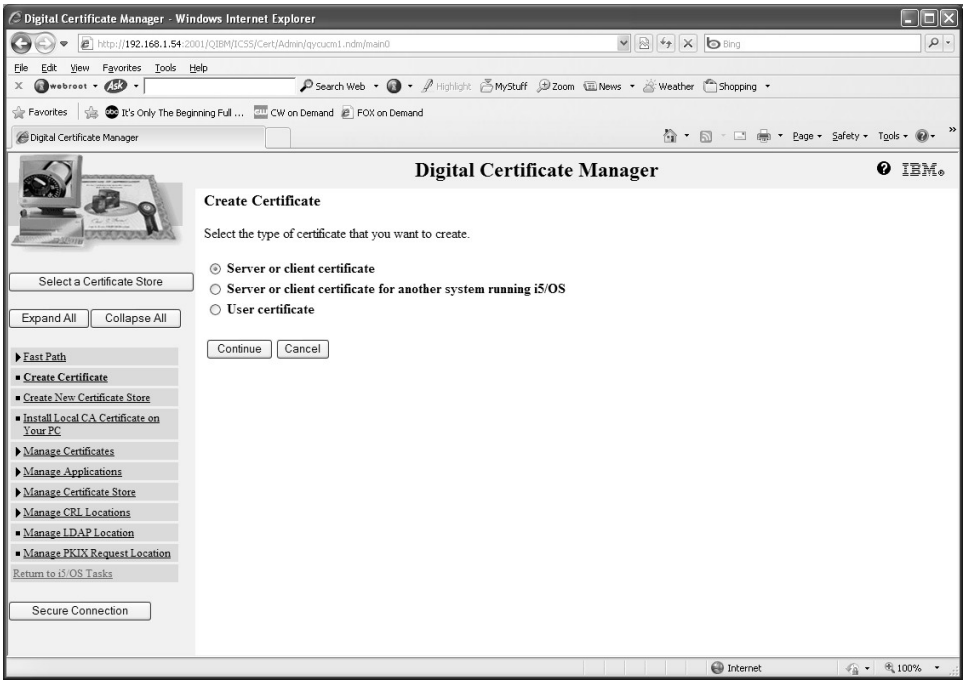
4. Enter a value for Application ID to be used on the Powertech Encryption for IBM i HTTP API calls. This value is how the RPGLE program will make use of the DCM configuration. Enter a description for the application. This is only used within DCM. Click **Add**.



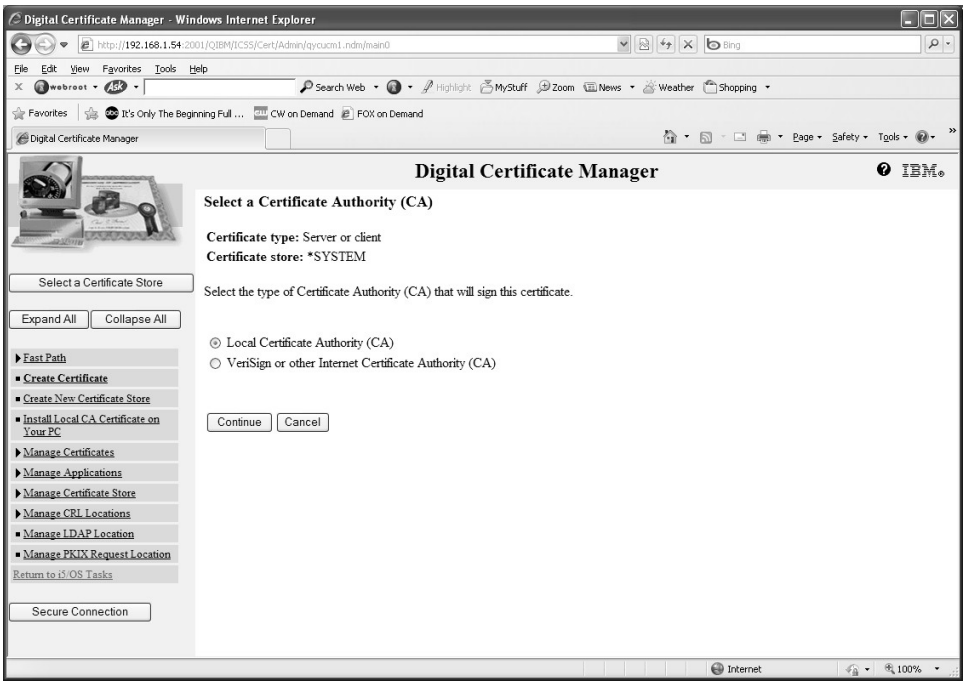
This completes adding to the client application definition to DCM. Next, we will create a client certificate.

Create a Client Certificate

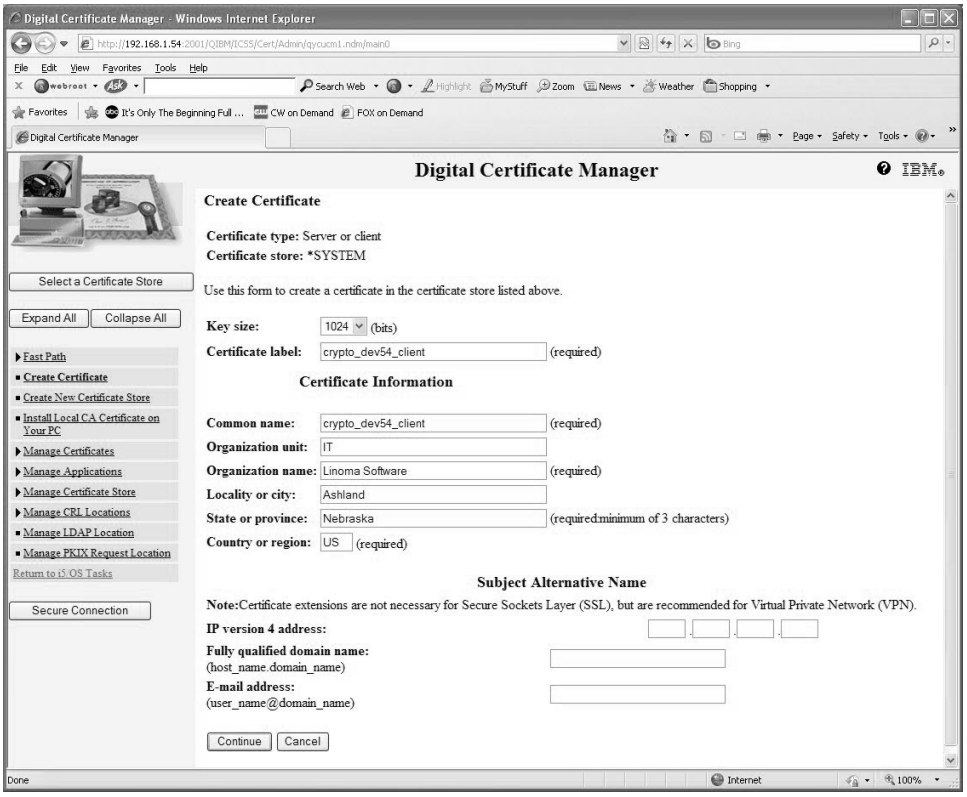
1. Select the Create Certificate link on the left side of the screen.



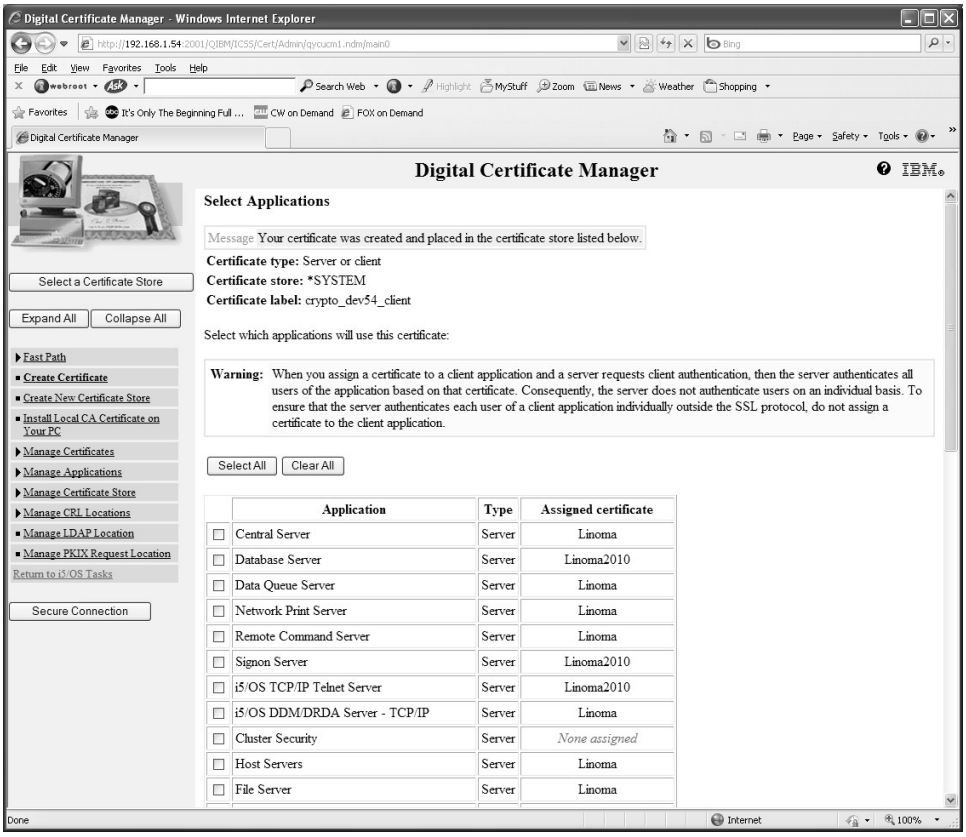
2. Select the Server or client certificate radio button, the click Continue.



3. Select the Local Certificate Authority (CA) radio button, then click **Continue**.

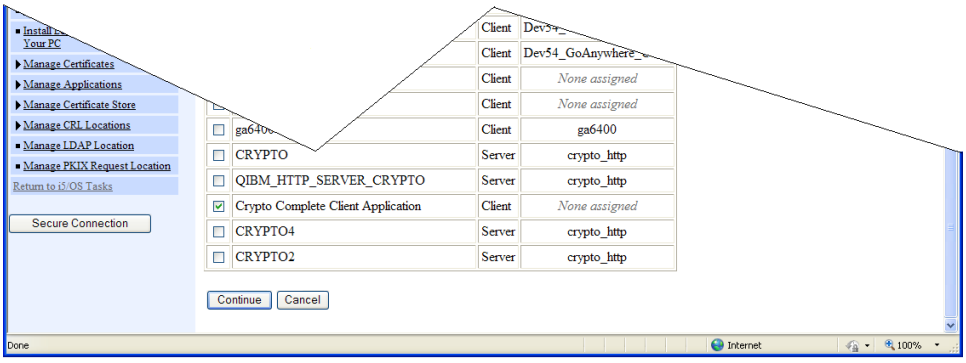


4. Enter a label for the client certificate. We have used ‘crypto_dev54_client’ because it is descriptive. Enter values appropriate for your organization, then click **Continue**.

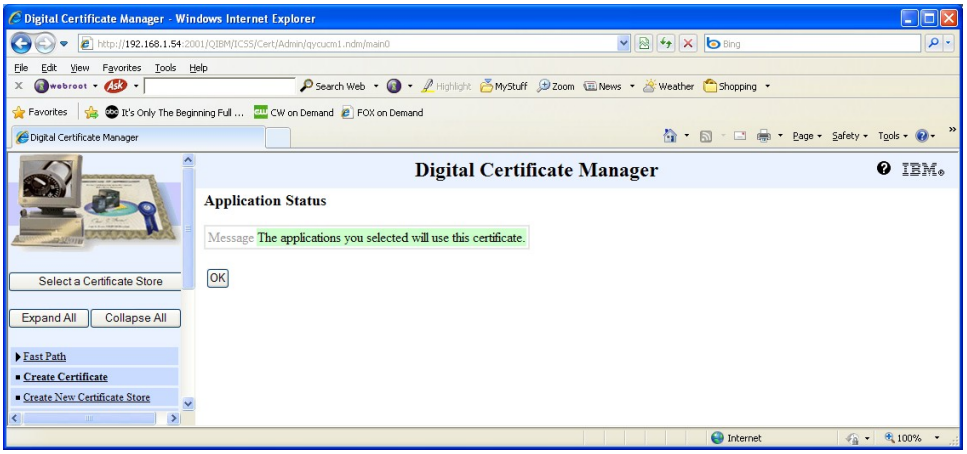


Associate the Client Certificate with the Client Application

1. When the certificate has been completed, Digital Certificate Manager provides an opportunity to select an application to associate the certificate to.



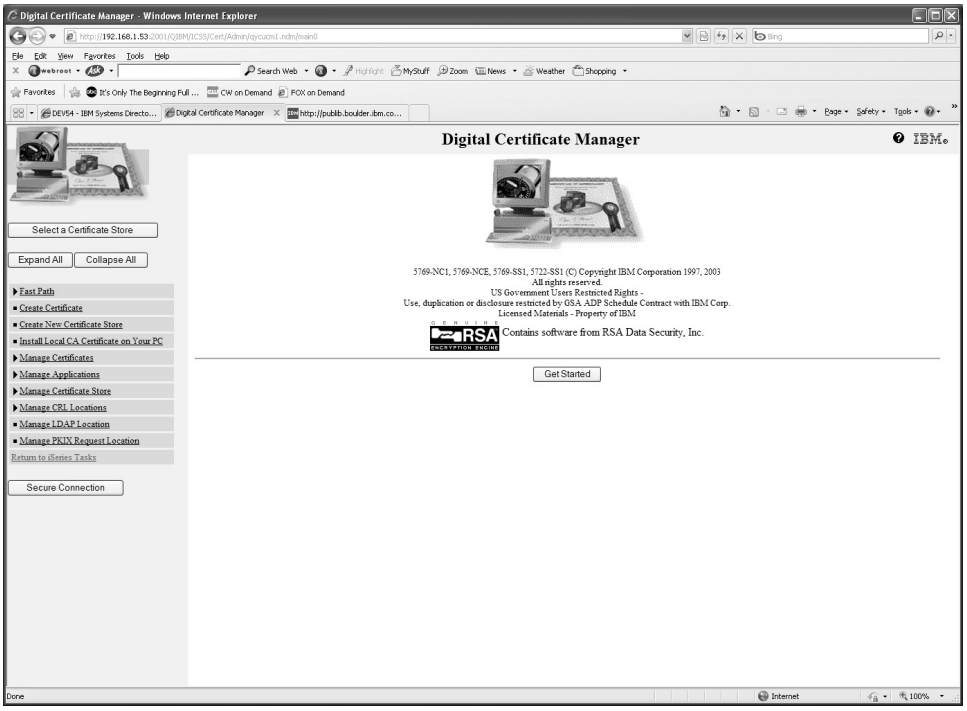
2. Scroll down in the list and locate the client application that you created earlier, select it.
3. Click **Continue**.



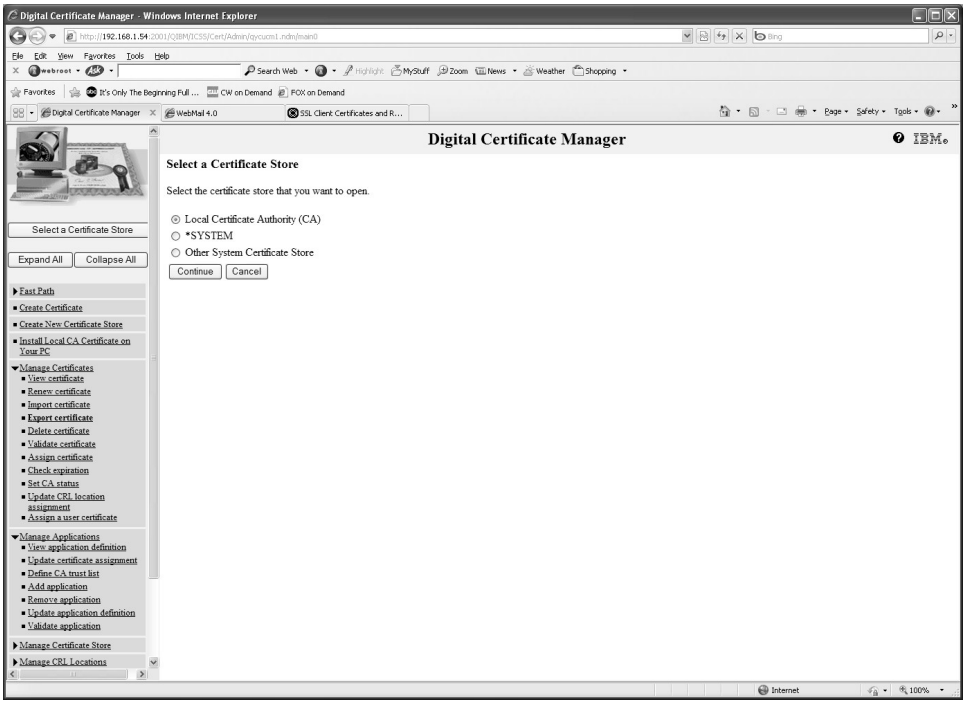
Export the Local CA Certificate from the Client System i

Next, we need to export the Local CA Certificate from the IBM i Client system and import it into the IBM i Server system.

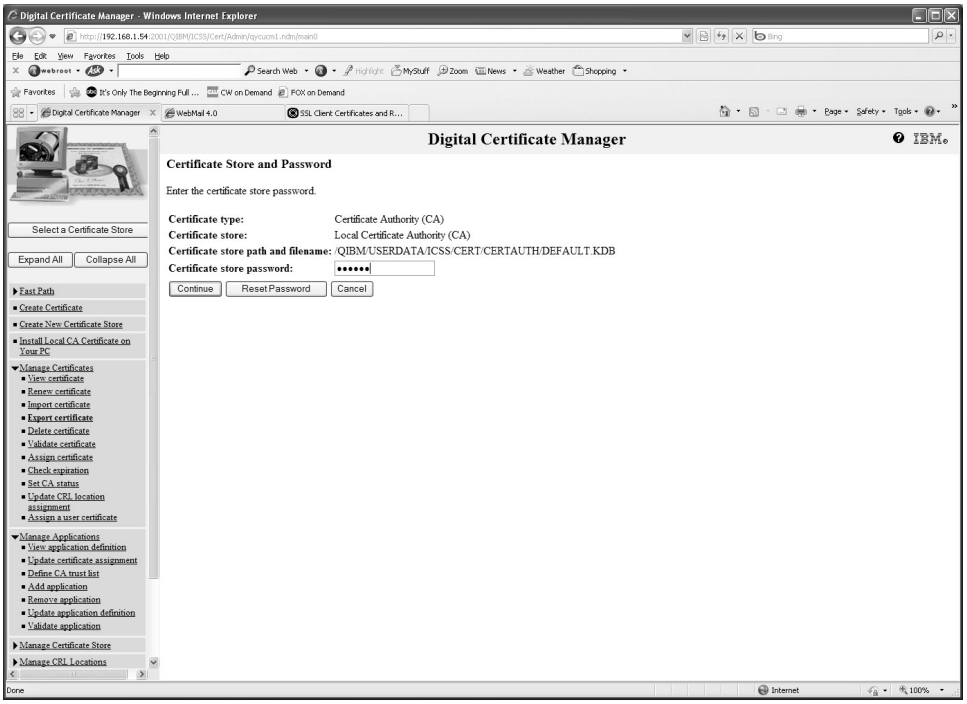
1. Upon signing into the client IBM i Administration server, the main screen is shown:



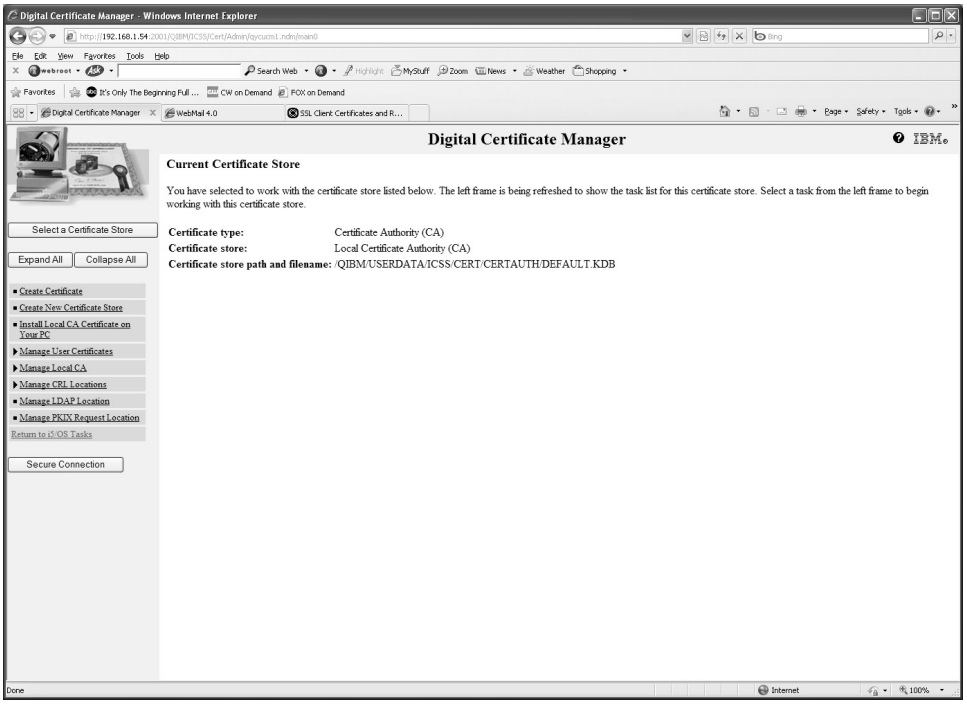
2. Click **Select a Certificate Store**.



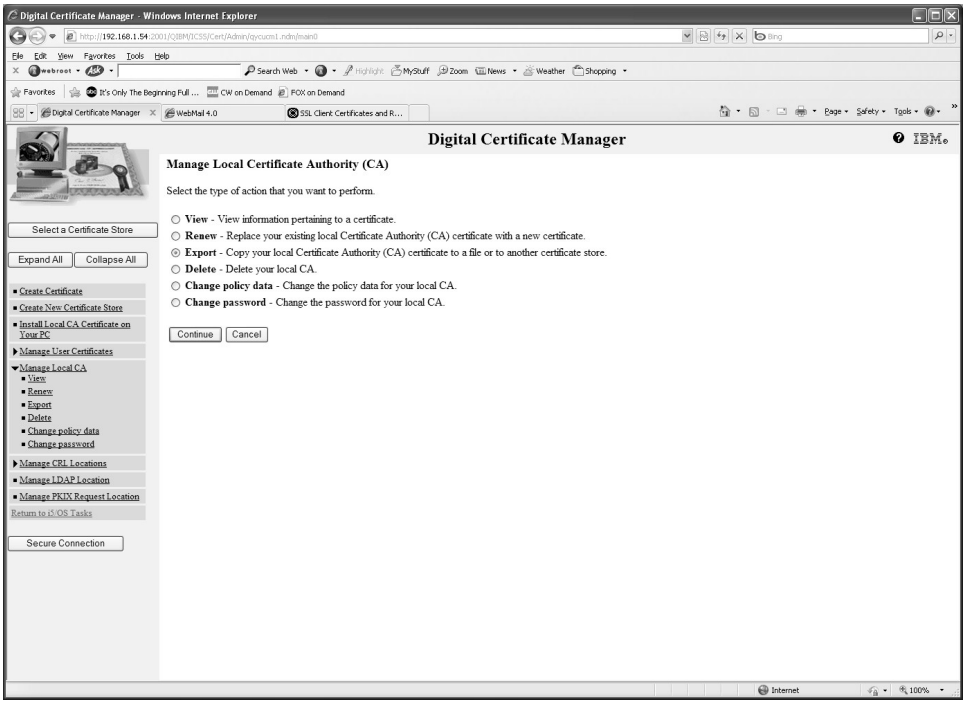
3. Select the Local Certificate Authority (CA) radio button and click Continue.



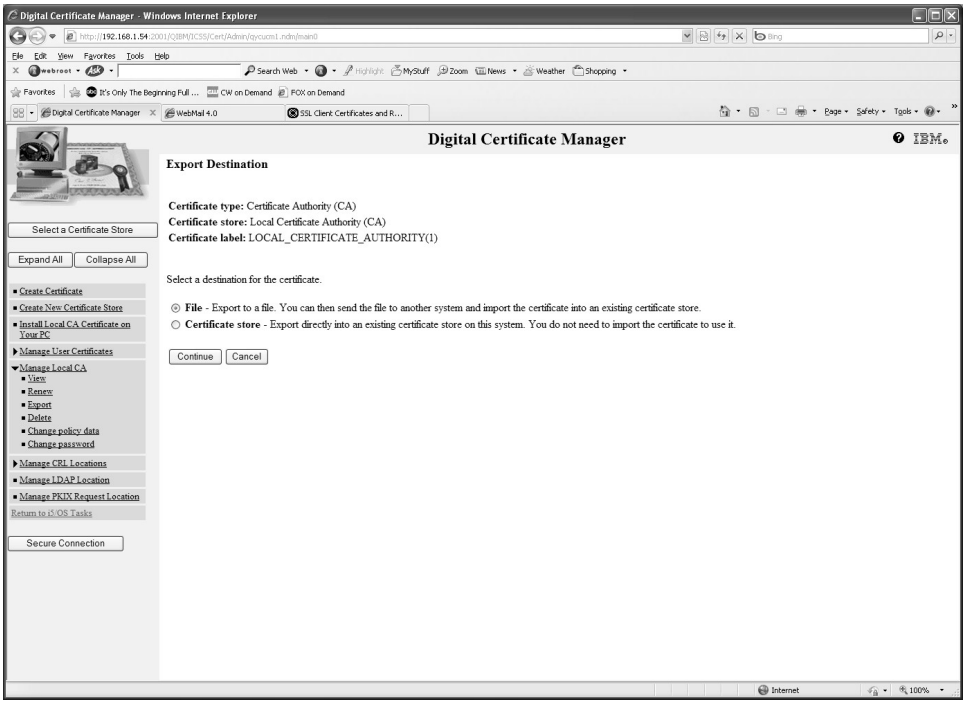
4. Enter the Certificate Store password and click **Continue**. If you do not know the password check with your system administrator.



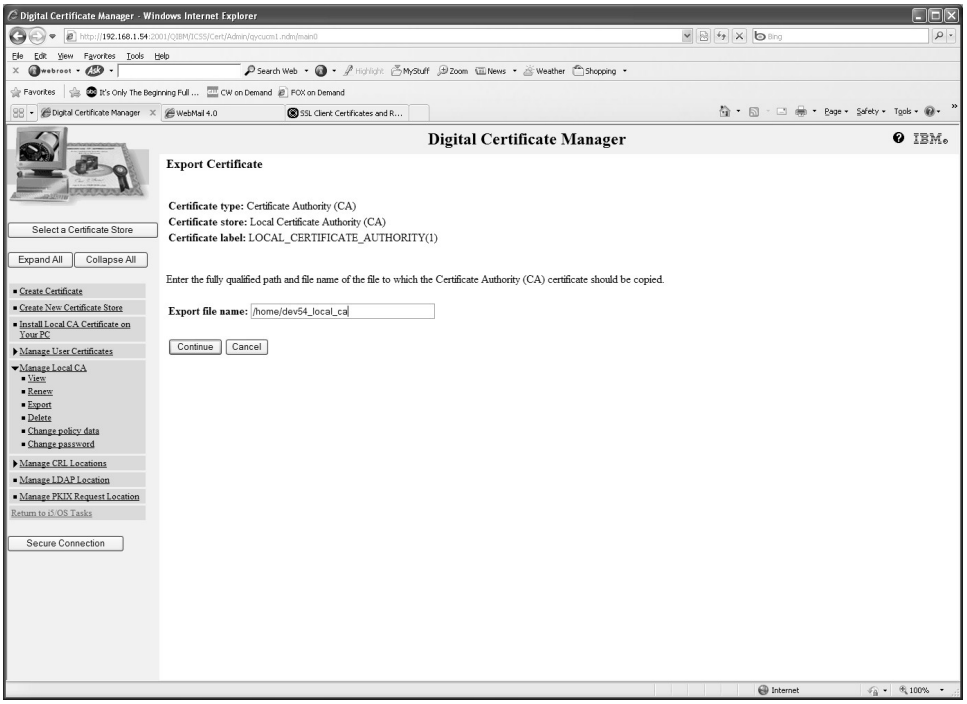
5. Click on the Manage Local CA link at the left side of the screen.



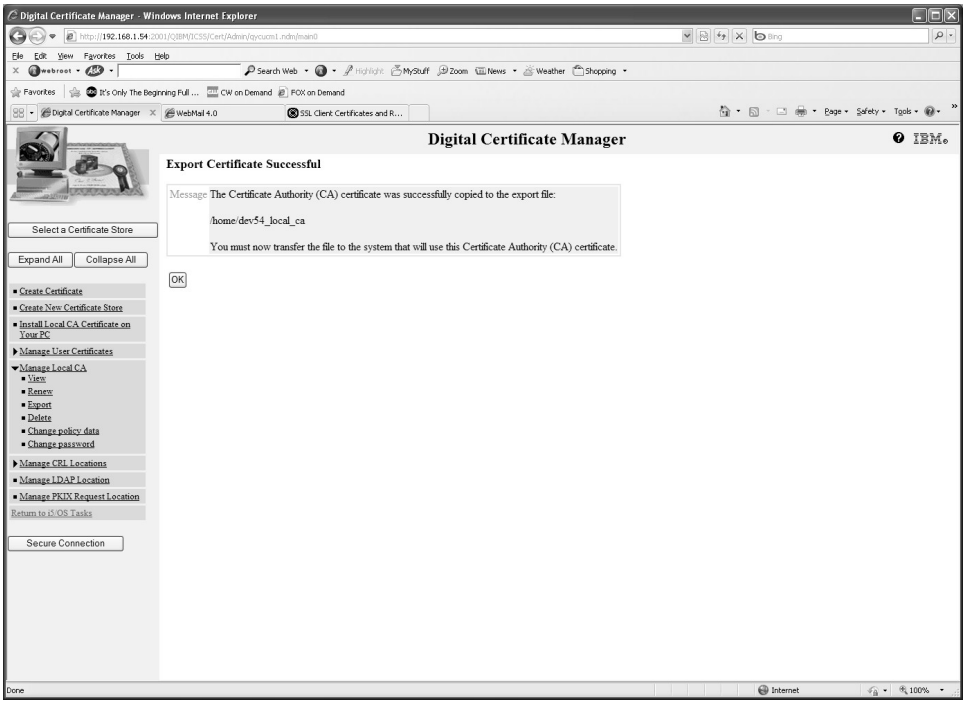
6. Click on the Export radio button, then click **Continue**.



7. Click on the File radio button, then click **Continue**.



8. Enter an IFS path and filename for the exported certificate, then click **Continue**. In our example, we use '/home/dev53_local_ca'.

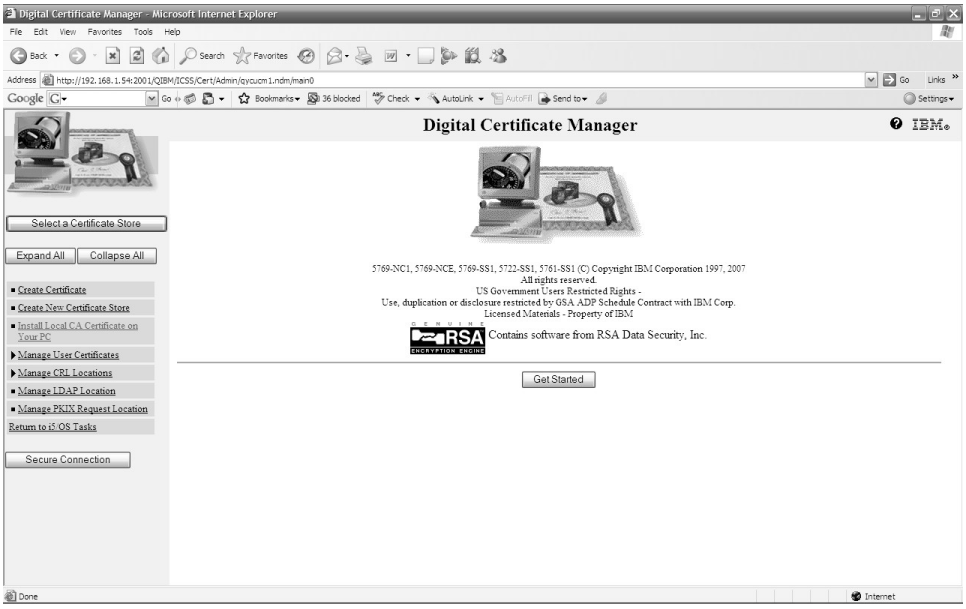


9. Export of the client IBM i Local Certificate Authority (CA) successful!

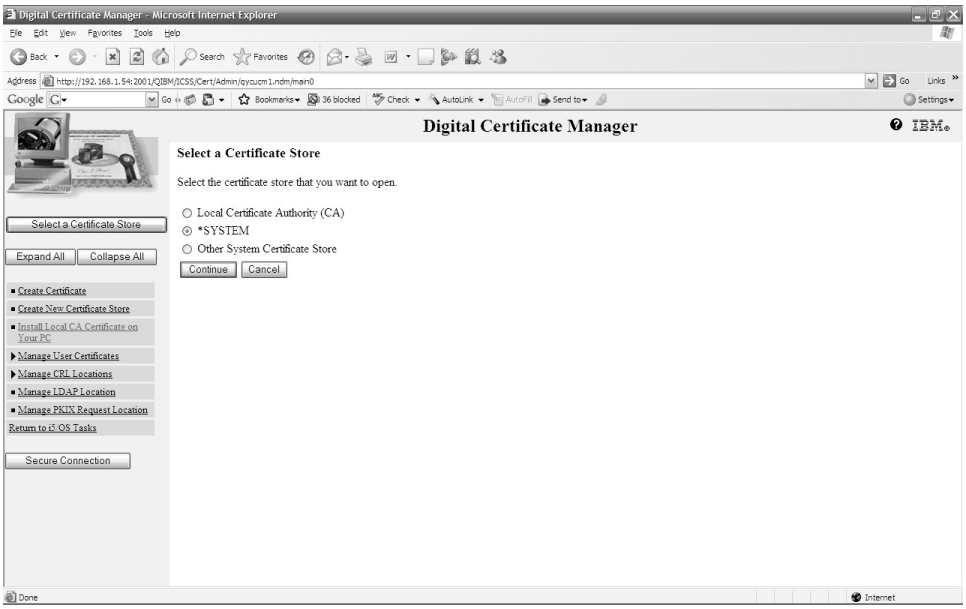
   The file will be a text representation of the certificate, so use IBM i FTP in ASCII mode to send the file from the client IBM i to the server System i.

Import the Client CA Certificate into the Server System i

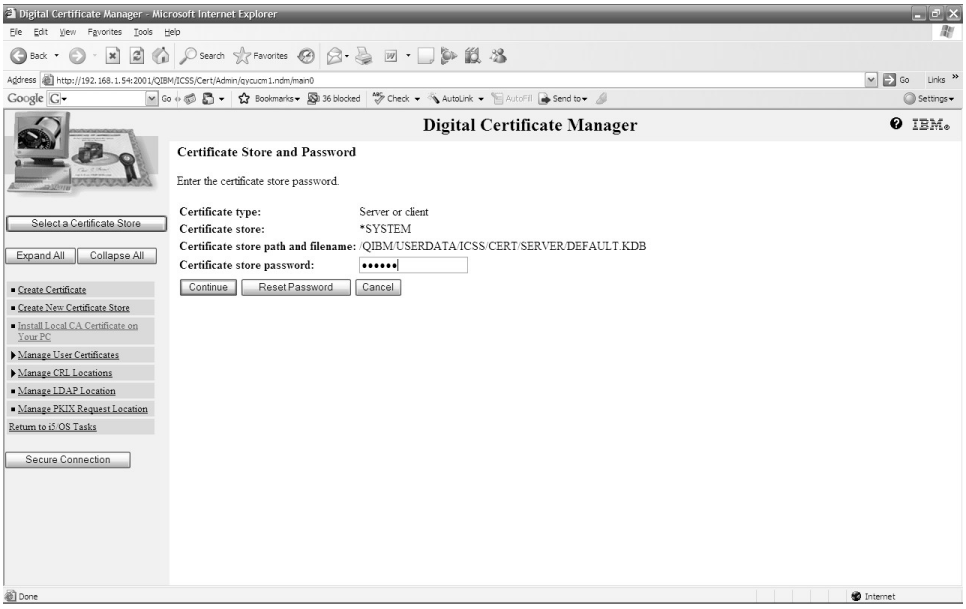
The next task is to import client CA into server *SYSTEM store on the server System i. Upon signing into the server IBM i Administration server, the main screen is shown:



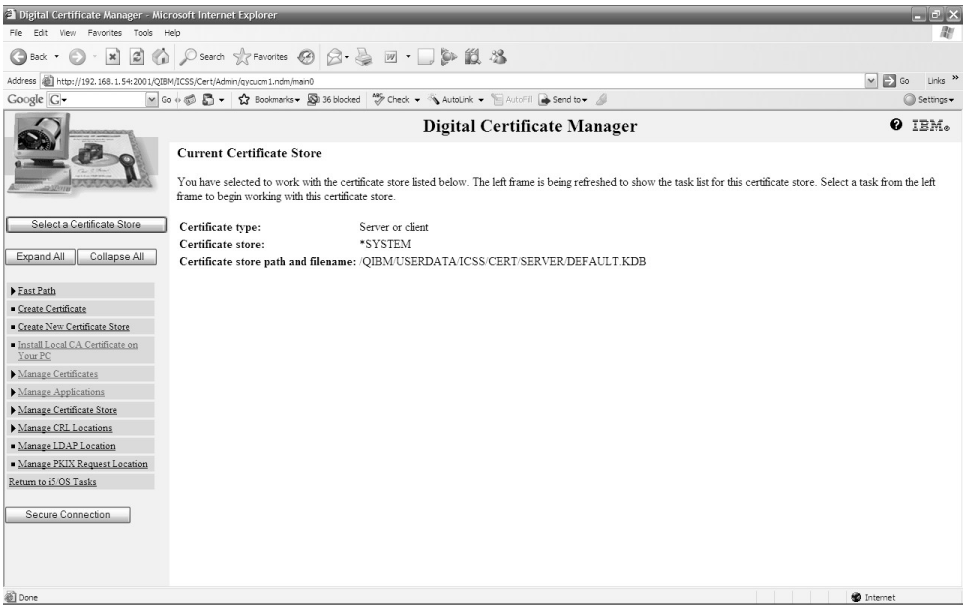
1. Click **Select a Certificate Store**.



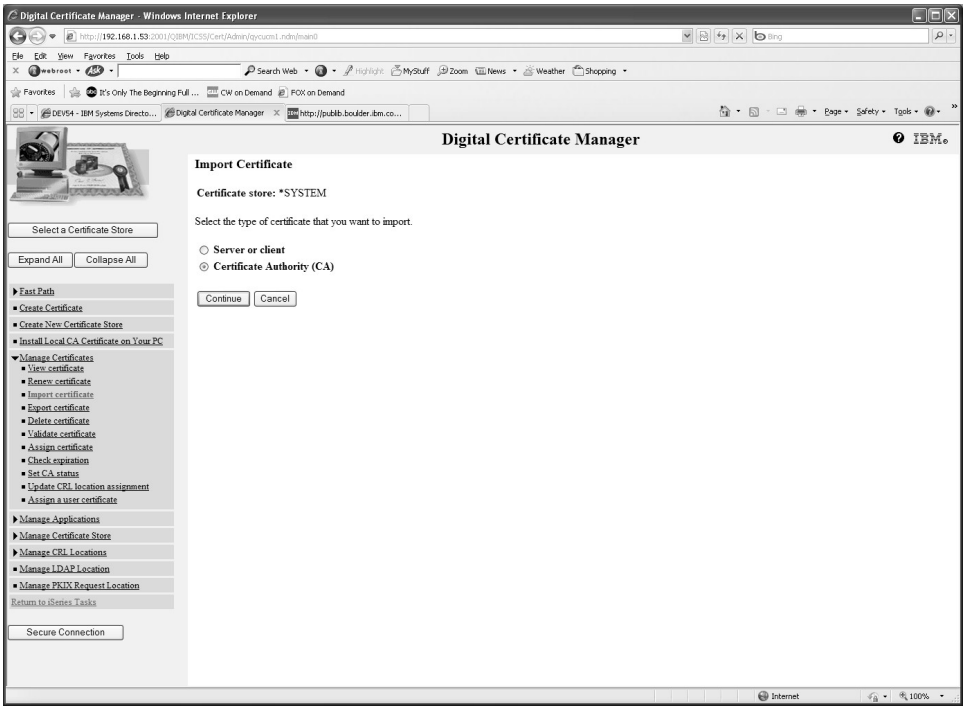
2. Click on the *SYSTEM radio button, then click **Continue**.



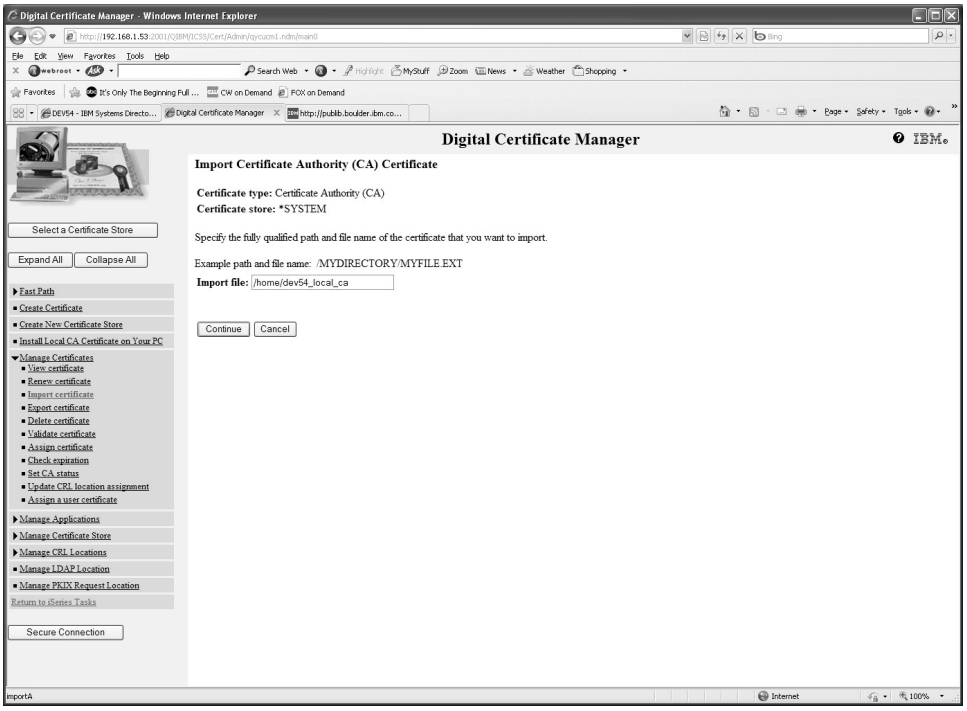
3. Enter the Certificate Store password and click **Continue**. If you do not know the password check with your system administrator.



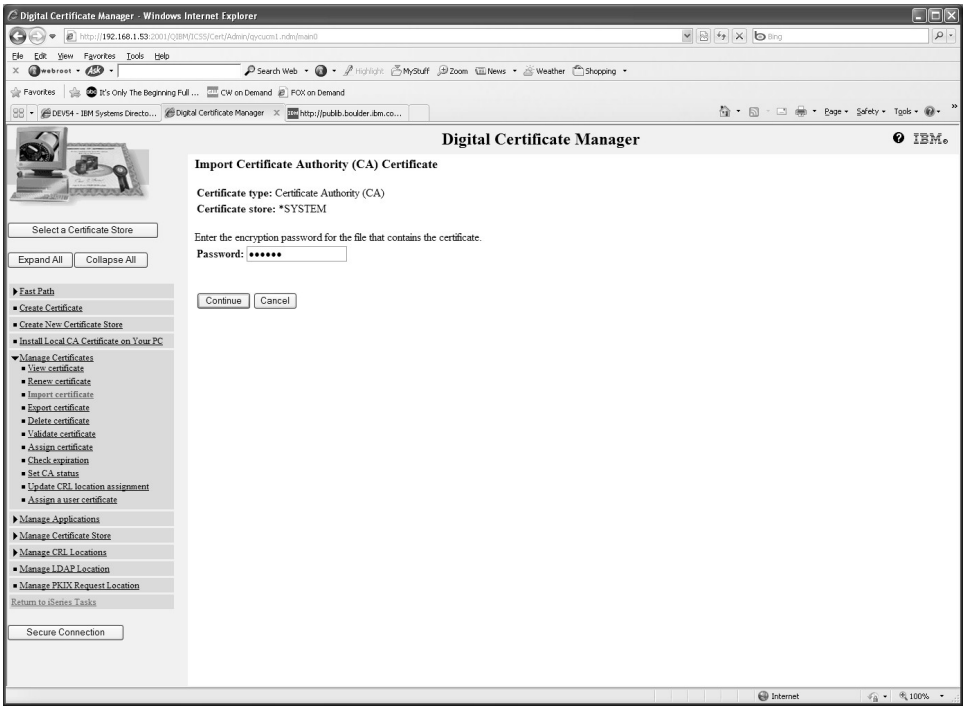
4. Click on the Manage Certificates link at the left side of the screen.



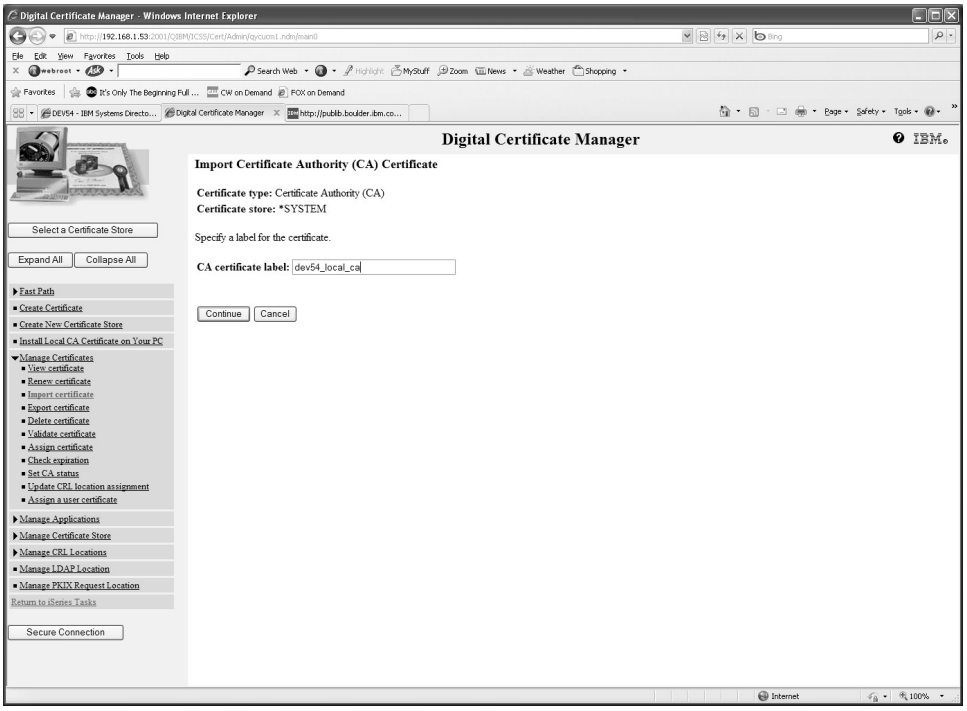
5. Click on the Import Certificate link at the left side of the screen. Select the Certificate Authority (CA) radio button, then click **Continue**.



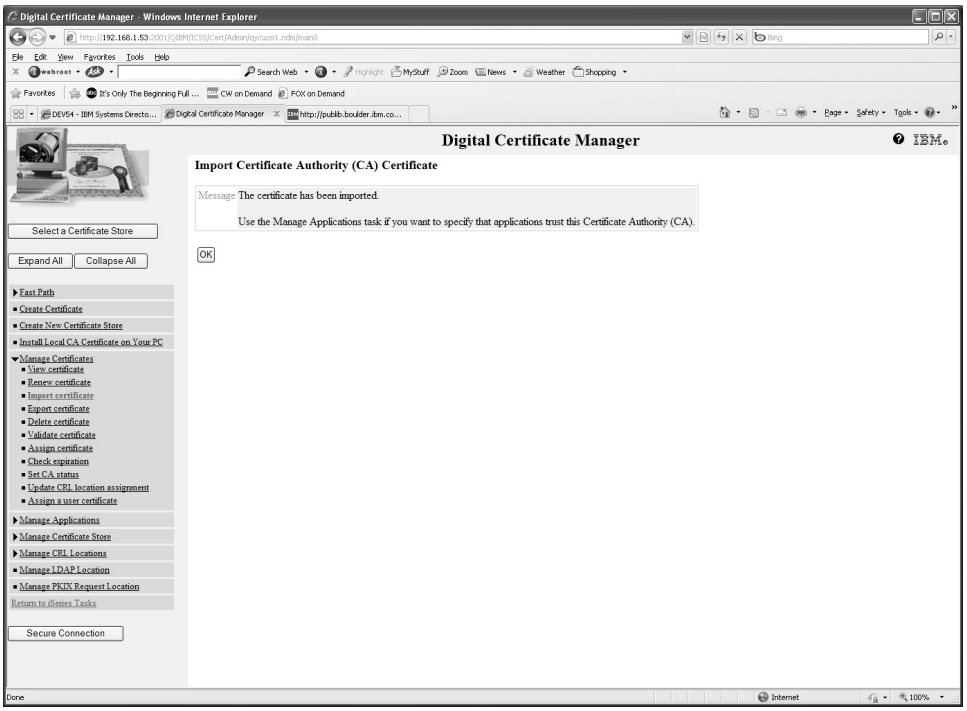
6. Enter the IFS path and filename of the certificate to import that you FTP'd from the client System i, then click **Continue**. In our example, we use '/home/dev53_local_ca' again.



7. Enter the encryption password that was specified during the export, then click **Continue**. If you do not know the password check with your system administrator.



8. Enter the name to be used to identify the imported client Local CA. We used ‘dev54_local_ca’ to identify the system that the certificate came from and to indicate that Default.PrintOnlyit was the Local Certificate Authority there. Click **Continue**.



You have successfully imported the certificate into the *SYSTEM store of the server IBM i.
This will allow the RPGLE and Java applications on the client IBM i to call the Powertech Encryption for IBM i HTTP APIs and to specify an Application ID when communicating over SSL to the server IBM i.

TIP: Remember to change the Powertech Encryption for IBM i HTTP server instance to “Require client certificate for connection” as shown below:

Welcome to Powertech Encryption for IBM i IFS Encryption

Powertech Encryption for IBM i is a comprehensive solution for protecting sensitive data through strong encryption technology, integrated key management and audit trails.

The design of Powertech Encryption for IBM i allows organizations to implement encryption quickly using intuitive screens and commands, while providing a high degree of protection. Every effort has been made in Powertech Encryption for IBM i to minimize the application

changes needed, allowing an organization to implement encryption successfully for less time and money.

IMPORTANT: The encryption algorithm of the symmetric key is used to determine the encryption algorithm with which the IFS files will be encrypted. For instance, if a symmetric key is created with AES-256, then the IFS files will also be encrypted with AES-256. See Symmetric Key Management for more information.

IFS Encryption

IFS directories can be set up for Automatic encryption.

Powertech Encryption for IBM i's innovative IFS Encryption Registry allows an organization to indicate (register) the directory(s) to encrypt. When IFS directory(s) are "activated" in the Registry, Powertech Encryption for IBM i will perform a mass encryption of the current Files in the directory and optionally include subdirectories. Powertech Encryption for IBM i can then automatically encrypt the Files in the directory and optionally subdirectories on an ongoing basis as new Files are added or changed.

The automated encryption function in Powertech Encryption for IBM i's IFS Encryption Registry will eliminate the need to make changes to your application programs for file encryption. When a user is authorized to view a file, it will be decrypted with no changes to existing programs. When a user is not authorized to view a file, the read will fail with the message "Object marked as a scan failure".

Getting Started with IFS Encryption

To get started with IFS encryption, you need to first configure Powertech Encryption for IBM i's Key Management settings.

Configure Settings and Keys

Use the commands (in the order listed) below to quickly configure Powertech Encryption for IBM i's Automatic IFS Encryption:

Step 1 – Change system values and IFS object settings.

1. QSCANFS set to *ROOTOPNUD.
2. QSCANFSCTL set to *NONE.
3. Create Object Scanning for the directory set to *YES.

   EXAMPLE:
   CHGATR OBJ('/home/lynn') ATR(*CRTOBJSCAN) VALUE(*YES)

4. Object Scanning set to *YES on the IFS File.

   EXAMPLE:
   CHGATR OBJ('/home/lynn/audit.trc') ATR(*SCAN) VALUE(*YES)

   Each file that is to be encrypted needs to have the attribute *CRTOBJSCAN set to *YES.

Step 2 – Call the ADDIFSEXTP (adds the Powertech Encryption Exit Point Programs) command. Find this command in the IFS Utility Menu (GO CRYPTO/CRYPTO14). This command adds the Fortra integrated exit programs to the QIBM_QPWFS_FILE_SERV, QIBM_QP0L_SCAN_CLOSE and QIBM_QP0L_SCAN_OPEN exit points on the system.

IMPORTANT: Fortra has integrated exit point sharing among select Fortra IBM i products. After you add the IFS exit points, a new EXITINT library and "Fortra Required Objects" will be created. The library and associated objects support the exit point sharing (for specific exit points).

Step 3 – Restart the QSERVER subsystem now or later.

After the exit programs have been added to the appropriate exit points, a pop-up window asks if you would like to restart the QSERVER subsystem now or later.

NOTE: The existing jobs under QSERVER will not recognize the newly added exit points until you restart QSERVER. You can press **F12** to ignore the QSERVER restart and prevent jobs from ending. Based on your internal operations, you might want to wait to restart until the next IPL.

Later execution of ADDIFSEXTP will not prompt you to restart QSERVER. Once you add the Fortra integrated exit program, and QSERVER restarts the first time, you do not need to restart QSERVER again. See Remove IFS Exit Point Programs (RMVIFSEXTP).

Step 4 – Run the STRIFSENCJ command. This command will submit the IFS server job to batch. Find this command on the IFS Utility Menu (GO CRYPTO/CRYPTO14).

This Job uses the CRYPTO Job Description shipped in the CRYPTO Library. Make any changes you want to this Job Description for your system before running the Command.

> **NOTE:** If you have not yet set up Powertech Encryption for IBM i on your system, at this point, configure Powertech Encryption for IBM i's Symmetric Key Management settings and establish your first Data Encryption Key. To do so:
>
> 1. CHGKEYPCY (Review and/or change the Key Policy settings. Prompt command with F4)
> 2. WRKKEYOFR (Indicate which users can create and manage Keys)
> 3. LODMSTKEY (Prepare a Master Encryption Key (MEK) by loading the passphrase parts)
> 4. CRYPTO/SETMSTKEY (Generate (set) the MEK using the loaded passphrase parts)
>
> See Getting Started in the Powertech Encryption for IBM i User Guide for additional details.

Step 5 – CRTKEYSTR (create a key store to contain the Data Encryption Keys (DEK))

Step 6 – CRTSYMKEY (create a Data Encryption Key (DEK) and save it into the Key Store)

Step 7 – Create an authorization list for determining who is authorized to decrypt.

Step 8 – WRKIFSENC (create an entry to set up which directory(s) to encrypt)

The documentation for these commands (and all other Powertech Encryption for IBM i commands) is contained within this Powertech Encryption for IBM i Manual. All Powertech Encryption for IBM i commands also have online help text which can be accessed with the F1 key when a command is prompted.

> **NOTE:** It is likely you will want a smaller group of users who can access (decrypt) sensitive data, compared to a larger group of users who can enter (encrypt) this data. This can be accomplished by using two Key Stores with their own respective authorities. In the first Key Store, you can store the Keys needed for encryption and give that Key Store a broader set of authorities. In the second Key Store, you can place the Keys needed for decryption and give that Key Store a much smaller set of authorities. For more information, see Controlling Access to Decrypted Values.

IFS Encryption Registry

Powertech Encryption for IBM i's IFS Encryption Registry allows an organization to specify (register) the directory(s) that require encryption. There are several configurable options that you can specify for each directory added to the registry.

One option is to have Powertech Encryption for IBM i encrypt all files in the subdirectories of the directory entered.

The registry also provides a target directory where the encrypted files should be stored.

Audit Trails

Entry Type	Description	Command Issued
60	IFS Encryption Registry – Entry added	ADDIFSENC
61	IFS Encryption Registry – Encryption Key changed	CHGIFSKEY
62	IFS Encryption Registry – Entry removed	RMVIFSENC
63	IFS Encryption Registry – Entry activated	ACTIFSENC
64	IFS Encryption Registry – Entry changed	CHGIFSENC
65	IFS Encryption Registry – Entry deactivated	DCTIFSENC
66	IFS Encryption Registry – Unable to Activate Entry	
67	IFS Encryption Registry – Unable to Deactivate Entry	
68	IFS Encryption Failed	
69	IFS Decryption Failed	
70	IFS Error	
71	Add Exit Point Program	
72	Remove Exit Point Program	
73	IFS Server Program Started	
74	IFS Server Program Stopped	
75	Debug Mode was changed	
76	Debug File was cleared	
77	Config File record was added	
78	Config File record was changed	

IFS Encryption Processes and Notes

The IFS Encryption process works in the following way:

When an IFS Entry is activated, the following processes will occur for all files in the directory (s):

1. When SAVDTA is set to *YES a backup of the directory and optionally subdirectories is created.
2. Journaling is started for the directory(s) and all files in the directory(s).
3. If the file is not zero bytes,
    a. the file is Encrypted into the Target directory
    b. The file is cleared and set to zero bytes.
    c. A record is added in the CRPFIFS File.

> **NOTE:** A record is added to the CRPFIFS file for every directory encrypted.

When an IFS Entry is Deactivated, the process will do the following to every file found in the directory(s):

1. When SAVDTA is set to *YES a backup of the directory and optionally subdirectories is created and a backup of the target directory is created.
2. Journaling is stopped for the file.
3. If the file has an entry in the CRPFIFS file and is zero bytes the file is Decrypted
4. The Target Encrypted File is deleted.
5. The record is removed from the CRPFIFS File.

> **NOTE:** The directory records are removed from the CRPFIFS.

For an Activated IFS Entry, when a user attempts to Open a file and the QIBM_QP0L_SCAN_OPEN exit point program is called then the following processes occur:

1. The User Authority is checked for the Decrypt Key Store
2. If an Authorization List has been entered the User Authority is checked on the Authorization List.
3. If the User is Authorized to read the file the file will be Decrypted back into the directory and the process will be allowed to continue.
4. If the User is NOT authorized to view the file, then the file is locked and the Open Process will fail. The IBM Message put out when the Open Fails is “Object marked as a scan failure”.
5. The file will remain locked until the IFS Server Job unlocks the file. If the Server Job is not running the File will remain locked.

For an Activated IFS Entry, when a File is Closed and the QIBM_QP0L_SCAN_CLOSE exit point program is called then the following processes occur:

1. If the file is not zero bytes then the following occurs:
    a. Check if a record exists in the CRPFIFS file.
        i. If a record does not exist.
            1. Check if the Target directory exists. If not then create it.
            2. Encrypt the file.
            3. Start Journaling
            4. Add a record to the CRPFIFS File
        ii. If the record exists
            1. Encrypt the file.
            2. Update the record in the CRPFIFS

> NOTE: When a file or directory is copied or moved, a File Open or Close may not be called. When this occurs then the QIBM_QP0L_SCAN_OPEN or QIBM_QP0L_SCAN_CLOSE exit point programs would not be called. When these operations are performed a journal entry may be created that allows the IFSENCJOB Server program to be alerted that the file or directory was copied or moved.

There are some processes that you need to be aware of when using the IFS Encryption Process.

1. When an unauthorized user tries to open a file to read, the file is locked on the system. The file will remain locked until the Server Program (IFSENCJOB) Opens and Clears the file. This process will unlock the file. Depending on how backed up the Server program (IFSENCJOB) is, this may not happen immediately.
2. When Encrypting or Decrypting large files, files 10 MB or larger, the process of opening or closing a file may take a little longer than you are used to. The process has to decrypt the file before you access the file and encrypt the file when the file closes.
3. If a user has a file decrypted and in edit mode, the file will stay decrypted until the file is closed by that user.
    a. If an unauthorized user tries to open the file to read it, the file will be locked.
    b. Also if another user tries to move that file out of the directory to another unencrypted directory, they will get the decrypted version of the file.
4. If a user moves a file into or out of an Encrypted directory, our processes may not know about it until the IFS Server Program retrieves the Journal after the process has occurred. No immediate Encrypting or Decrypting may take place. If the QIBM_QP0L_SCAN_OPEN or QIBM_QP0L_SCAN_CLOSE exit programs are not called then no Authority checking will take place.
    a. When the server program (IFSENCJOB) encounters a Journal record showing a file or directory was copied or moved out of an Encrypted directory, the following process is ran.

i. Check to see if a record exists in the CRPFIFS file for the Source file or directory. If the record does not exist we ignore the file or directory.

ii. If the source record still exists in the CRPFIFS file we

1. Check to see if the user was authorized to Copy or Move the file.
2. If they were not authorized and the process was a move we recreate the directory structure and zero byte files for the source.
3. If the user was authorized to move the files we decrypt the files into the destination directories and then remove the source Encrypted files and directories and then remove the CRPFIFS records.

5. When a user tries to Decrypt a file thru a mapped drive, the default user that is used is QUSER. Powertech Encryption for IBM i will use the exit point QIBM_QPWFS_FILE_SERV to retrieve the User Id that the user signed in as. This user Id will be used to check for authority to Decrypt the File. When the exit point QIBM_QP0L_SCAN_OPEN is called to Decrypt the file, the user QUSER must be authorized to the Decrypt Key Store to be able to Decrypt the file.
   a. If you want all operations to be authorized from a mapped drive you can also give QUSER authority to the Authorization List.
   b. If you want certain users to have authority to Decrypt only when using a mapped drive, you need to give those users or groups *USE Authority to the Authorization List.

Remove IFS Encryption from the system

Use the commands (in the order listed) below to remove Powertech Encryption for IBM i's IFS Encryption:

Step 1 – DCTIFSENCJ (Deactivate all files in the encrypted Directory(s)).

Run this for all Activated IFS IDs

Step 2 – ENDIFSENCJ (This will end the IFS Server Job)

Step 3 – RMVEXTPGMS (Remove the Powertech Encryption for IBM i Exit Point Programs)

Step 4 – You must stop and restart every Job that will Access the IFS Files.

You can do one of two things

1. IPL the system
2. Or end any restart any Job that will be using the exit programs. The instructions below will stop and restart many of the servers. There may be others that are not listed.
    a. End Processes
        i. ENDTCPSVR *NETSVR
        ii. ENDTCPSVR SERVER(*FTP)
        iii. ENDTCPSVR SERVER(*HTTP) HTTPSVR(*ALL)
        iv. ENDHOSTSVR *FILE
        v. ENDHOSTSVR *DATABASE
        vi. ENDSBS QSERVER
        vii. End any Batch Jobs that access the IFS Data
    b. Restart Processes
        i. STRSBS QSERVER
        ii. STRTCPSVR *NETSVR
        iii. ENDTCPSVR SERVER(*FTP)
        iv. ENDTCPSVR SERVER(*HTTP) HTTPSVR(*ALL)
        v. STRHOSTSVR *FILE
        vi. STRHOSTSVR *DATABASE
        vii. Restart any Batch Jobs that access the IFS Data

Work with Misc Settings

When using the new CRCONFIG file to set up the IFS encryption, you need to use the following steps to setup and run the IFS Encryption processes.

In all cases the CRCONFIG file must be in the CRYPTO library. When the file is not found or a setting is not found then the default settings are used.

The WRKCONFIG command allows you to change the settings in the CRCONFIG file. In the new version the WRKCONFIG command will be located on the “Product information” menu.

Below are the default settings in the file when installed.

Name	Value
Default Settings	
EXTFILE_USECMTCTRL	YES
IFS_ASPNAME _____	

Name	Value
IASP Settings	
IFS_IASP_CRYPTO_OBJECTS_LIBRARY	CRYPTO
IFS_IASP_FILES_JOURNALLED_BY_THIRD_PARTY	NO
IFS_IASP_JOURNAL_LIBRARY	CRYPTO
IFS_IASP_JOURNAL_NAME	CRJNI001
LOC1 Settings	
IFS_LOC1_CRYPTO_OBJECTS_LIBRARY	CRYPTO
IFS_LOC1_FILES_JOURNALLED_BY_THIRD_PARTY	NO
IFS_LOC1_JOURNAL_LIBRARY	CRYPTO
IFS_LOC1_JOURNAL_NAME	CRJNI001
LOC2 Settings	
IFS_LOC2_CRYPTO_OBJECTS_LIBRARY	CRYPTO
IFS_LOC2_FILES_JOURNALLED_BY_THIRD_PARTY	NO
IFS_LOC2_JOURNAL_LIBRARY	CRYPTO
IFS_LOC2_JOURNAL_NAME	CRJNI001
LOC3 Settings	
IFS_LOC3_CRYPTO_OBJECTS_LIBRARY	CRYPTO
IFS_LOC3_FILES_JOURNALLED_BY_THIRD_PARTY	NO
IFS_LOC3_JOURNAL_LIBRARY	CRYPTO
IFS_LOC3_JOURNAL_NAME	CRJNI001
LOC4 Settings	
IFS_LOC4_CRYPTO_OBJECTS_LIBRARY	CRYPTO
IFS_LOC4_FILES_JOURNALLED_BY_THIRD_PARTY	NO
IFS_LOC4_JOURNAL_LIBRARY	CRYPTO
IFS_LOC4_JOURNAL_NAME	CRJNI001
LOC5 Settings	
IFS_LOC5_CRYPTO_OBJECTS_LIBRARY	CRYPTO
IFS_LOC5_FILES_JOURNALLED_BY_THIRD_PARTY	NO
IFS_LOC5_JOURNAL_LIBRARY	CRYPTO
IFS_LOC5_JOURNAL_NAME	CRJNI001

Configuration Settings when in the IFS Encryption Registry the Journal Location (JRNLOC) field is set to *DEFAULT

This is used for all normal IFS Encryption. *ASP is used for IASP replication. *LOCx are used when a third party Journal is being used to journal a directory you want to encrypt.

> When the Journal Location for a field is *DEFAULT then no records need to be added to the CRCONFIG file. Use the WRKCONFIG command to enter or change the objects.

The following objects will already exist in the CRYPTO library and must stay there.

1.	CRVL003	*VLDL	IFS Encryption Registry
2.	CRPFIFS	*PF	IFS Encryption Information
3.	CRPFIFSL1	*LF	
4.	CRPFIFSL2	*LF	
5.	CRPFIFSL3	*LF	
6.	CRPFIFSL4	*LF	
7.	CRPFIFS2	*PF	IFS Encryption Changes File
8.	CRPFIFSPRC	*PF	IFS Encryption Changes File
9.	CRPFIFSPR1	*LF	
10.	CRJNI001	*JRN	Journal
11.	CRJRI001	*JRNRCV	Journal Receiver
12.	CRLSTSEQ	*DTAARA	Keeps track of the Last Receiver and Seq Number
13.	CRSRVRUN	*DTAARA	Used to let the server program know to End.

Configuration Settings when in the IFS Encryption Registry the Journal Location (JRNLOC) field is set to *IASP

*IASP is used for IASP replication is used on a system. The default objects will still exist in the CRYPTO library.

When the Journal Location for a field is *IASP then the following setup needs to be done.

1. Create or designate an IASP library to hold the Powertech Encryption for IBM i objects created below that need to be copied from the CRYPTO library.

2. All of the objects below must exist in the IASP Library. These objects should only hold information about the IASP files.
    a. CRPFIFS
    b. CRPFIFSL1
    c. CRPFIFSL2
    d. CRPFIFSL3
    e. CRPFIFSL4
    f. CRPFIFSPRC
    g. CRPFIFSPR1
    h. Journal Receiver. Use the following command CRTJRNRCV JRNRCV (IASPLIB/CRJRI001)
    i. Journal. Use the following command CRTJRN JRN(IASPLIB/CRJNI001) JRNRCV(IASPLIB/CRJRI001)
    j. CRLSTSEQ
    k. CRSRVRUN
3. Create a DDM file in the CRYPTO library called CRPFIFSA over the CRPFIFS file in the CRASP library. For example if my IASP name is IASP1 and my library in the IASP I am using is CRASP then use the following command.
    a. CRTDDMF FILE(CRYPTO/CRPFIFSA) RMTFILE(CRASP/CRPFIFS) RMTLOCNAME(*RDB) RDB(IASP1)
4. The CRCONFIG file in library CRYPTO must have the following records added and set. Use the WRKCONFIG command to enter or change the records.
    a. IFS_IASP_CRYPTO_ OBJECTS_LIBRARY
        i. The value must be the library that holds the new objects created above. This must not be CRYPTO.
    b. IFS_IASP_FILES_JOURNALLED_BY_THIRD_PARTY →NO
    c. IFS_IASP_JOURNAL_LIBRARY
        i. The value must be the library that holds the journal. This must not be CRYPTO.
    d. IFS_IASP_JOURNAL_NAME
5. The authorities for these objects should be the same as for the ones in the CRYPTO Library.
6. The CRPFIFS file must be empty.
7. The CRLSTSEQ Data Area holds the current Journal receiver name in the first 10 characters. This should be changed to the current Journal Receiver for the journal that is journaling the directory.
    a. 'CRJRI001 000000000000001'

8. The CRLSTSEQ Data Area holds the last sequence number read in the current Journal receiver in the last 15 characters. This should be changed to the last sequence number in the Current Journal Receiver for the journal that is journaling the directory.
9. The CRSRVRUN Data Area should be set to “N”.
10. Create an IFS Encryption Registry Entry in the (CRVL003) located in the CRYPTO library and set the JRNLOC value to *IASP.
11. Start the IFSENCJOBA by using the following command.
    a. STRIFSENCJ JRNLOC(*IASP)
12. Activate the Entry. This process will make sure that the IASP Library is available to the job. If the library is not then the SETASPGRP command will be ran. If the command fails then the Activate will fail.

IFS_IASP_CRYPTO_OBJECTS_LIBRARY	The IASP Library that holds the copied objects from the CRYPTO Library. 1. CRPFIFS 2. CRPFIFSL1 3. CRPFIFSL2 4. CRPFIFSL3 5. CRPFIFSL4 6. CRPFIFSPRC 7. CRPFIFSPR1 8. Journal object. If you are using the same naming convention as Powertech Encryption for IBM i does, then the name would be CRJNI001. 9. Journal Receiver objects. If you are using the same naming convention as Powertech Encryption for IBM i does, then the name would be CRJRI001. 10. CRLSTSEQ 11. CRSRVRUN
IFS_IASP_FILES_JOURNALLED_BY_THIRD_PARTY	Value should be NO when encrypting an IASP directory
IFS_IASP_JOURNAL_LIBRARY	The IASP Library that holds the Journal to be used.
IFS_IASP_JOURNAL_NAME	The Journal name .

Configuration Settings when in the IFS Encryption Registry the Journal Location (JRNLOC) field is set to *LOC1 through *LOC5

*LOCx is used when a directory is already being journaled by a third party journal.

When the Journal Location (JRNLOC) for a field is *LOC1, *LOC2, *LOC3, *LOC4 or *LOC5, then the following setup needs to be done.

1. Check that the directory and all files in the directory are being journaled by using the following:
2. WRKLNK '/DirectoryName'
    a. Enter option 8 next to the directory and the files to make sure they are journaled. The Directory and the files must be journaled. The journal information will be on the 4th or 5th page. Take note of the journal Library and Name.
3. Once you know the Journal Library and Journal Name use the command WRKJRN to view the Journal Information.
    a. Take option 8 to get the Attached receiver. Make note of the receiver name.
    b. Press F17 from there to get the Last sequence number.
4. Create or designate a library to hold the Powertech Encryption for IBM i objects created in the next step below. These objects can be copied from the CRYPTO library. Make sure the CRPFIFS file is empty in the new library.
5. All of the objects below must be copied to the new Library designated above.
    a. CRPFIFS Physical File
    b. CRPFIFSL1 Logical File over CRPFIFS
    c. CRPFIFSL2 Logical File over CRPFIFS
    d. CRPFIFSL3 Logical File over CRPFIFS
    e. CRPFIFSL4 Logical File over CRPFIFS
    f. CRPFIFSPRC Physical File
    g. CRPFIFSPR1 Logical File over CRPFIFSPRC
    h. CRLSTSEQ Data Area
    i. CRSRVRUN Data Area
6. The authorities for these objects should be the same as for the ones in the CRYPTO Library.
7. The CRPFIFS file must be empty.
8. The CRLSTSEQ Data Area holds the current Journal receiver name in the first 10 characters and the last 15 characters holds the Last Sequence Number of the current Journal. For example. 'CRJRI010 000000000000235'

   a. Change the first 10 characters to hold the current Journal receiver name found above.
   b. Change the last 15 characters to hold Last Sequence Number of the current Journal found above. Be sure to include the leading zeros.
9. The CRSRVRUN Data Area should be set to “N”.
10. The CRCONFIG file must have the following records added. Use the WRKCONFIG command to enter or change the objects.
   a. IFS_LOCx_CRYPTO_ OBJECTS_LIBRARY
      i. The value must be the library that holds the new objects created above. This must not be CRYPTO.
   b. IFS_LOCx_FILES_JOURNALLED_BY_THIRD_PARTY
   c. IFS_LOCx_JOURNAL_LIBRARY
      i. The value must be the library that holds the journal. This must not be CRYPTO.
   d. IFS_LOCx_JOURNAL_NAME
11. Create an IFS Encryption Registry Entry in the (CRVL003) located in the CRYPTO library and set the JRNLOC value to the appropriate value. Either *LOC1, *LOC2, *LOC3, *LOC4 or *LOC5.
12. Start the IFSENCJOBx (where x is 1 thru 5) by using the following command.
   a. STRIFSENCJ JRNLOC(*LOCx)
13. Activate the Entry in the IFS Registry.

IFS_LOCx_CRYPTO_OBJECTS_ LIBRARY	The Library that holds the copied objects from the CRYPTO Library. 1. CRPFIFS 2. CRPFIFSL1 3. CRPFIFSL2 4. CRPFIFSL3 5. CRPFIFSL4 6. CRPFIFSPRC 7. CRPFIFSPR1 8. Journal object. 9. Journal Receiver object 10. CRLSTSEQ 11. CRSRVRUN
IFS_LOCx_FILES_JOURNALLED_BY_ THIRD_PARTY	Value should be YES when using these options.

IFS_LOCx_JOURNAL_LIBRARY	The Library that holds the Journal to be used.
IFS_LOCx_JOURNAL_NAME	The Journal name .

Overview of Exit Program Integration

Several Powertech products utilize IBM i exit points to keep track of access to the system. One in particular, the File Server exit point, has created a bottleneck for a few reasons. To assist with overcoming this issue, we have introduced Exit Program Integration in our Antivirus 8.09 and Encryption 4.0 product versions. The advantages of this integration are:

- Once registered, you will not need to recycle your QSERVER jobs or adjust in the future
- Fortra products can easily share the exit point and fire in the correct order

Additionally, we support other exit programs you may have that also monitor the File Server exit point, and can even fire programs that use either of the available formats, where IBM would natively only run the higher level program.

Exit program integration provided with Powertech Encryption for IBM i includes the following exit points:

- The File Server exit point
- The Integrated File System Scan on Open exit point
- The Integrated File System Scan on Close exit point

Exit Program Sequencing

Exit program integration allows you to control the order of how the exit programs are called. Exit programs can prevent the following two exit program groups from being called:

1. The exit programs of Fortra products that support exit program integration, which are called first. Between the products, the order of how the exit programs are called is determined programatically and cannot be customized.
2. Other exit programs, which are called next. You can specify the order between these programs.

> **NOTE:** Any of the exit programs on this chain can stop the processing of the other exit programs further down the chain.

Exit Program Integration and Other Solutions Using Exit Points

Powertech Antivirus for IBM i

Powertech Encryption for IBM i version 4.0 and higher will automatically configure exit point integration for co-use between Powertech Encryption for IBM i and Powertech Antivirus.

Powertech Exit Point Manager for IBM i

Powertech Exit Point Manager for IBM i is compatible with but does not currently participate in the exit program integration.

To enable both Powertech Exit Point Manager for IBM i and Powertech Encryption for IBM i to use the File Server exit point, configure these solutions as follows:

1. On the main menu, choose option **7** IFS Encryption Menu, then option **21**, IFS Utility menu and finally option **3**, Add IFS Exit Point Programs and register the IFS exit programs.
2. In Exit Point Manager:
    a. Ensure the file server is not activated.
    b. Choose "Activate with supplemental" for the file server.
    c. Perform activation when convenient.
3. See 'Using Exit Program Integration with Multiple Powertech Products' on the Fortra Community Portal.

Other Solutions' Configuration

If Exit Point manager or another program is already registered in the File Server Exit Point, please see the multiple product document 'Using Exit Program Integration with Multiple Powertech Products' located in the Powertech Knowledge Base on our Community Portal.

Other solutions from Fortra and third-party providers can also use exit point integration. To add the exit program for another solution into the integration, follow these steps:

1. Use the Work with Exit Program Integration (WRKEXTPGM) Command to add an exit program to a specific exit point.
2. Under "exit program for which product", select one of the "NON-FORTRA" entries.
3. Under "Product", specify the exit program library and name.

Reference

The topics in this section include descriptions of Powertech Encryption for IBM i's options and controls.

Glossary

The topics in this section include descriptions of Powertech Encryption for IBM i terms.

DEK - Data Encryption Key

A Data Encryption Key (DEK) is a Symmetric Key which is used to encrypt and decrypt data. An organization can create one or more DEKs using Powertech Encryption for IBM i. For instance, a DEK could be created to encrypt/decrypt credit card numbers and a second DEK could be created to encrypt/decrypt social security numbers.

A DEK should be randomly generated by Powertech Encryption for IBM i in order to provide the highest degree of protection. Depending on your organization’s key policy, you can additionally have Powertech Encryption for IBM i generate a DEK which is based on a passphrase entered by the user.

Key Store

Data Encryption Keys (DEK) are contained within Key Stores. You can create one or more Key Stores on the IBM i using Powertech Encryption for IBM i. For instance, one Key Store could be used to contain DEKs for protecting Order Entry data, and a second Key Store could be used to contain DEKs for protecting Payroll data.

A Key Store is created as a *VLDL (Validation List) object on the IBM i. You can control access to the Key Store *VLDL object using IBM i object security.

Master Encryption Key

A Master Encryption Key (MEK) is a special Symmetric Key used to protect (encrypt) the Data Encryption Keys (DEKs) contained in a Key Store. An organization can create up to 8 MEKs per environment on the IBM i. For instance, a MEK could be used to encrypt the Order Entry DEKs contained in a Key Store, and a second MEK used to encrypt the Payroll DEKs contained in another Key Store.

A MEK is generated by Powertech Encryption for IBM i using passphrases entered by designated users. Depending on the organization's key policy, up to 8 different passphrases can be required (by different users) in order to generate a MEK.

MEKs are stored in a *VLDL (Validation List) object on the IBM i called CRVL001.

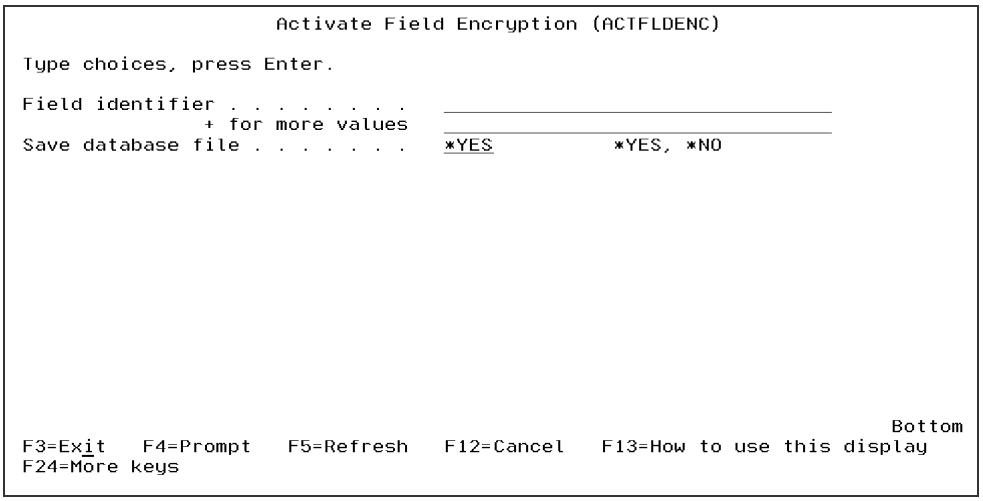
PEK – Product Encryption Key

A PEK is used by Powertech Encryption for IBM i to protect (encrypt) the Master Encryption Keys (MEKs) and user-defined settings (i.e. Key Policy, Key Officers, Security Alerts, etc).

Powertech Encryption for IBM i automatically generates the PEK using a combination of the IBM i serial number and a secret value. The PEK only resides in memory as-needed and is never stored.

Activate Field Encryption (ACTFLDENC)

The Activate File Fields Encryption (ACTFILFLDE) command will activate any *INACTIVE entries in the Field Encryption Registry for the file that use Field Procedures.



It is strongly recommended to submit this command to batch.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

IMPORTANT: Before using the ACTFLDENC command to encrypt production data, do the following steps:

1. Make sure you have *ALL authority to the database file containing the field to encrypt.
2. Within a test environment, you should have tested ACTFILFLDE, tested any API calls needed for encryption/decryption and tested your applications thoroughly with encrypted values.
3. No applications or users should be currently using the database file containing the field to encrypt.
4. The ACTFILFLDE command will perform a mass encryption of the current field values. You should allocate enough downtime for the ACTFILFLDE to execute. Execution times will vary depending on the processor speed of your system, the number of records in your database file, and other activity running on the system at the time. In order to estimate the execution time for ACTFILFLDE, you should run the ACTFILFLDE command over some test data first.
5. Check (and double check) the field entry settings using the DSPFLDENC command. Especially make sure the database file name, field name, type and length is correct.

   IMPORTANT: When activating a field using a DB2 Field Procedure, and if there are already other DB2 Field Procedures on the file, then you should have at least *USE authority to the 'Full' Authorization Lists assigned to those other fields, as well as at least *USE authority to the Key Stores that contain the encryption and decryption Keys used by those fields. This is because IBM's ALTER TABLE statement (used in the activation process) runs the decrypt/encrypt processes for all fields that have a DB2 Field Procedure. Failure to have proper authorities will cause loss of data.

The ACTFILFLDE command performs the following primary steps:

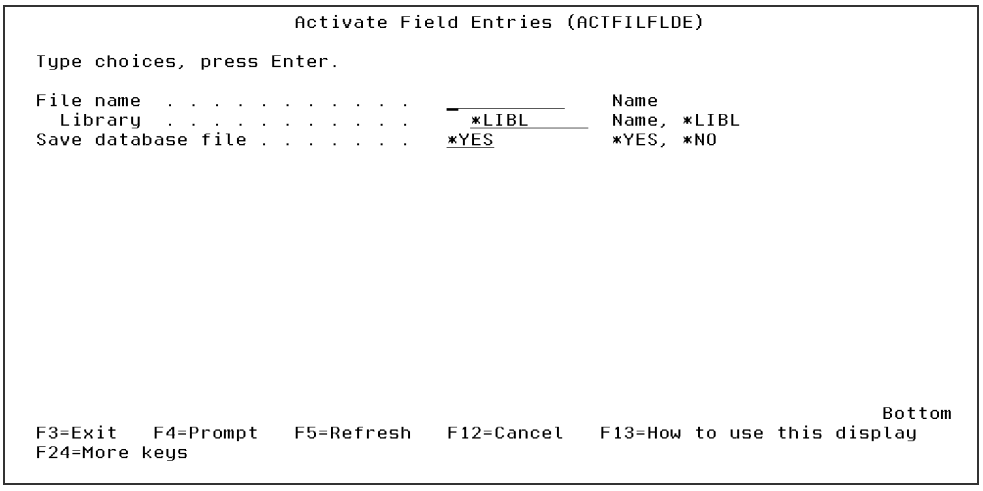
1. Obtains an exclusive (*EXCL) lock on the database file containing the field to encrypt.
2. Optional: Creates a backup of the database file (containing the field to encrypt) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
3. Performs a mass encryption of the current field values in the database file. If a DB2 Field Procedure is specified for the field, then it will be added to the field at that time.
4. The exclusive lock will be released on the database file containing the encrypted field.
5. The status of the field entries will be changed to *ACTIVE.

Notes on ACTFLDENC:

- After the ACTFLDENC command completes: Once you have determined that your applications are working properly with the encrypted values, you can remove the Save file (created in step 3 above) containing a backup of the database file.
- If an external file is specified for storing the encrypted values:
  - The external file will be created with the same authorities as the database file containing the field to encrypt. After the ACTFLDENC command completes, you can adjust the authorities on the external file (if needed) using the EDTOBJAUT command.
  - The external file will be created with the parameter of SIZE(*NOMAX), which will allow the external file to contain an unlimited number of records. After the ACTFLDENC command completes, you can adjust the SIZE limit on the external file (if needed) using the CHGPF command.
  - Any users that need to encrypt the field values will need *CHANGE authority to the CRVL002 *VLDL object, which holds the field registry information. Any users that need to decrypt the field values will need at least *USE authority to the CRVL002 *VLDL object.

Activate Field Entries (ACTFILFLDE)

The Activate File Fields Encryption (ACTFILFLDE) command will activate any *INACTIVE entries in the Field Encryption Registry for the file that use Field Procedures.



It is strongly recommended to submit this command to batch.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

IMPORTANT: Before using the ACTFLDENC command to encrypt production data, do the following steps:

1. Make sure you have *ALL authority to the database file containing the field to encrypt.
2. Within a test environment, you should have tested ACTFILFLDE, tested any API calls needed for encryption/decryption and tested your applications thoroughly with encrypted values.
3. No applications or users should be currently using the database file containing the field to encrypt.
4. The ACTFILFLDE command will perform a mass encryption of the current field values. You should allocate enough downtime for the ACTFILFLDE to execute. Execution times will vary depending on the processor speed of your system, the number of records in your database file, and other activity running on the system at the time. In order to estimate the execution time for ACTFILFLDE, you should run the ACTFILFLDE command over some test data first.
5. Check (and double check) the field entry settings using the DSPFLDENC command. Especially make sure the database file name, field name, type and length is correct.

   IMPORTANT: When activating a field using a DB2 Field Procedure, and if there are already other DB2 Field Procedures on the file, then you should have at least *USE authority to the 'Full' Authorization Lists assigned to those other fields, as well as at least *USE authority to the Key Stores that contain the encryption and decryption Keys used by those fields. This is because IBM's ALTER TABLE statement (used in the activation process) runs the decrypt/encrypt processes for all fields that have a DB2 Field Procedure. Failure to have proper authorities will cause loss of data.

The ACTFILFLDE command performs the following primary steps:

1. Obtains an exclusive (*EXCL) lock on the database file containing the field to encrypt.
2. Optional: Creates a backup of the database file (containing the field to encrypt) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.

3. Performs a mass encryption of the current field values in the database file. If a DB2 Field Procedure is specified for the field, then it will be added to the field at that time.
4. The exclusive lock will be released on the database file containing the encrypted field.
5. The status of the field entries will be changed to *ACTIVE.

How to Get There

On the File Field Encryption Menu, choose option **3**, Activate File Encryption Entry(s).

Options

Activate file name (ACTFILE)

Specify the name of the file that contains the field(s) to activate.

The possible values are:

file-name The name of the file that contains the field(s) to activate.

The possible library values are:

library-name Enter the name of the library where the file is located.
***LIBL** Locate the file within the library list.

Save database file (SAVDTA)

Indicate if the database file (containing the field to encrypt) should be saved (backed up) into a Save File before the activation process begins. It is highly recommended to save the database file for error recovery purposes.

The possible values are:

***YES** Save the database file into a Save File before activation begins.

> NOTE:
> - **The created Save File will be named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.**
> - **Before using this option, ensure that enough disk space is available for a saved copy of the database file.**

***NO** Do not save the database file before the activation process begins.

Activate IFS Encryption (ACTIFSENC)

The Activate IFS Encryption (ACTIFSENC) command will activate an *INACTIVE entry in the IFS Encryption Registry.

It is strongly recommended to submit this command to batch.

> **NOTE:** If Powertech Antivirus is installed on the system on which an IFS encryption entry is being activated, a scan will be performed as part of the activation step. Ensure the profile running the activation has an entry in the system directory.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

> **IMPORTANT:** The following specific authorities are required for users running this command:
>
> - *RWX data authority and *ALL object authority to the directories and files to encrypt.
> - *CHANGE authority to the CRVL003 (Validation List object which contains the IFS Encryption Registry), CRPFIFS, CRPFIFSL1, CRPFIFSL2, CRPFIFSL3, CRPFIFSL4, CRPFIFS2 and CRPFIFSLOG files, which will be updated during this process.
> - *USE authority to the Authorization List assigned to this entry.

> **WARNING:** Before using the ACTIFSENC command to encrypt production data, make sure you have performed the following steps:
>
> 1. Verified you have all the previously listed authorities.
> 2. Within a test environment, you should have tested ACTIFSENC, and tested your applications thoroughly with encrypted files.
> 3. No applications or users should be currently using the directory(s) and files to encrypt.

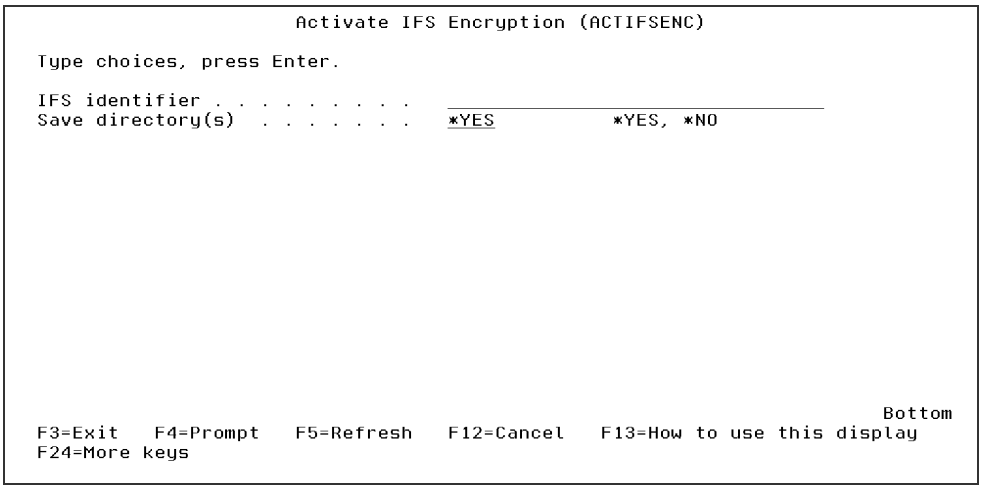
The ACTIFSENC command will perform a mass encryption of the files in the directory(s) to encrypt. You should allocate enough downtime for the ACTIFSENC to execute. Execution times will vary depending on the processor speed of your system, the number of files in the

directory(s), and other activity running on the system at the time. In order to estimate the execution time for ACTIFSENC, you should run the ACTIFSENC command over some test files first.

Check (and double check) the IFS entry settings using the DSPIFSENC command. Especially make sure the source and target directories are correct and that the include sub directories parameter is correct.

The ACTIFSENC command performs the following primary steps:

1. Optional: Creates a backup of the IFS directory and subdirectories if INCSUBDIR is *YES (containing the files to encrypt) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999. This backup file will be placed in the CRYPTO library.
2. Performs a mass encryption of the current files in the directory, as well as its subdirectories if INCSUBDIR is *YES.
3. Journaling will be started over the directory and if include subdirectories (INCSUBDIR) is *YES then the subdirectories will be journaled as well.
4. The status of the IFS entry will be changed to *ACTIVE.



How to Get There

From the IFS Encryption Menu, choose option **10**. Or, prompt (**F4**) the command **CRYPTO/ACTIFSENC**.

Options

IFS identifier (IFSID)

Enter the IFS identifier to activate.

Save IFS directory(s) and files (SAVDTA)

Indicate if directory(s) (containing the files to encrypt) should be saved (backed up) into a Save File before the activation process begins. It is highly recommended to save the directory(s) and files for error recovery purposes.

The possible values are:

***YES** Save the directory(s) and files into a Save File before activation begins.

> **NOTE:** The created Save File will be named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
> Before using this option, ensure that enough disk space is available for a saved copy of the directory(s) and files.

***NO** Do not save the directory(s) and files before the activation process begins.

Add a Field Encryption Pending Key (ADDPNDKEY)

When using Field Procedures, the Add a Field Encryption Pending Key (ADDPNDKEY) command allows authorized users to add a Pending key to a Field entry that can be used in the next key rotation process.

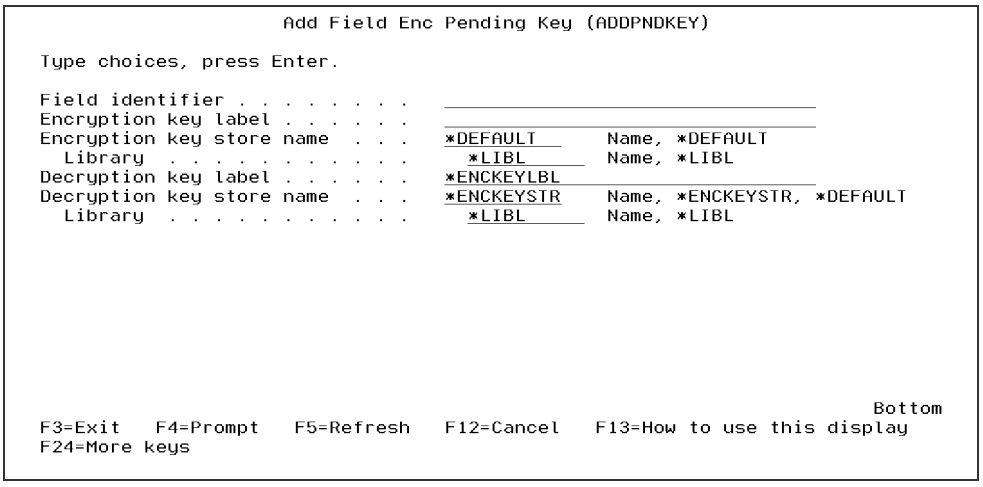
This command can be used for *ACTIVE field entries that use Field Procedures.

Up to 99,999 keys can be rotated for a field entry.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.



How to Get There

In the File Field Encryption Menu, choose option **6**.

Options

Field identifier (FLDID)

Indicate the unique name of the field entry to add the Pending key.

Encryption key label (ENCKEYLBL)

Indicate the label of the Symmetric Key to use in the Pending key.

Encryption key store name (ENCKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use in the Pending key.

The users (or user groups) which need to encrypt values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Decryption key label (DECKEYLBL)

Indicate the label of the Symmetric Key to use for decrypting the field values.

The possible values are:

decryption-key-label Indicate the label of the key to use for decryption.

> WARNING: If specifying a different key label than the label specified for encryption, then that decryption key should contain the same key value as the encryption key.

***ENCKEYLBL** Use the same label as specified on the ENCKEYLBL parameter.

Decryption key store name (DECKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the field.

The users (or user groups) that need access to the decrypted values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***ENCKEYSTR** Use the same Key Store as specified on the ENCKEYSTR parameter.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

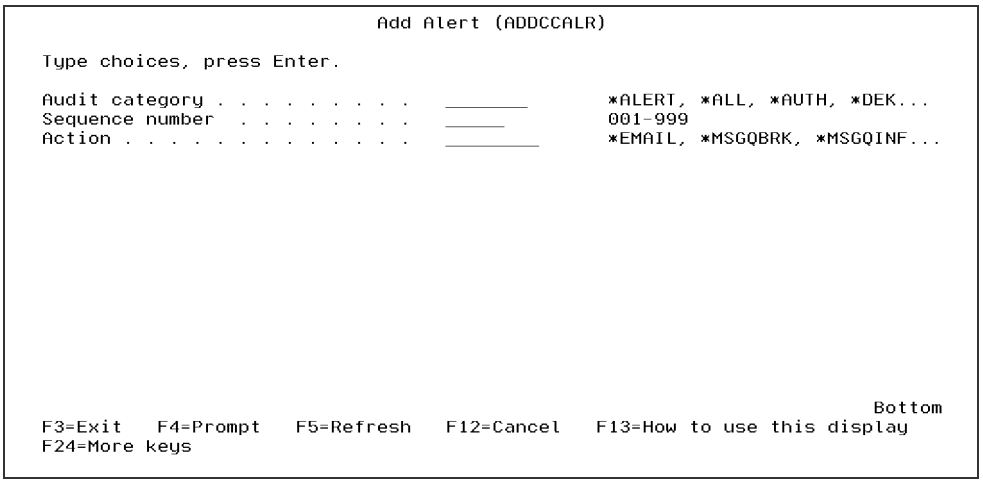
The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Add Alert (ADDCCALR)

The ADDCCALR command allows an authorized user to add a new Security Alert.

> NOTE: Any maintenance to the Security Alerts is logged into an audit file.



The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key policy and alerts” authority setting

Do the following steps to add an Alert:

1. Prompt (F4) the command of **CRYPTO/ADDCCALR**.
2. Press F1 on any parameter for complete online help text.
3. Press Enter after the parameter values are entered.

Field Descriptions

Audit category	Indicate the audit category to monitor. Valid categories are: *ALL All categories *ALERT Any maintenance activities for Security Alerts *AUTH Any authority errors encountered in Powertech Encryption for IBM i *DEK Any maintenance activities for Data Encryption Keys *EKMGR Any maintenance to the External Key Manager entries along with a any connection issues to the External Key Manager will trigger an alert, which includes the following audit types: • 42-Entry added • 43-Entry changed • 44-Entry removed • 45-Connection failed *FLDREG Any maintenance activities for Field Registry Entries *IFSREG Any maintenance activities for IFS Registry Entries *KEYOFR Any maintenance activities for Key Officer settings *KEYPCY Any maintenance activities for Key Policy settings *MEK Any maintenance activities for Master Encryption Keys
Sequence number	Indicate the sequence number from 1-999. This allows you to have multiple alerts sent out for each Audit Category.

Action	Indicate the action to perform. Valid actions are: *EMAIL Send email to one or more recipients using the SNDDST command. *MSGQBRK Send break messages to a specified message queue using the SNDBRKMSG command. *MSGQINF Send messages to a specified message queue using the SNDMSG command. *QAUDJRN Write journal entries into the QAUDJRN journal file. *PTGLOG Send log messages to the Protegrity Defiance Enterprise Security Administrator (ESA). *QHST Send messages to the QHST log using the SNDMSG command. *QSYSOPR Send messages to QSYSOPR using the SNDMSG command. *SYSLOG Send messages to an external log server using SYSLOG protocol. *USER Send messages to a User using the SNDMSG command.
To email address	If the Action is *EMAIL, then specify one or more email addresses to notify. Multiple email addresses should be separated by a comma. For instance: jsmith@abc.com,mlight@abc.com,kdodd@abc.com
To user profile	If the Action is *USER, then specify the name of the User Profile to send the message to.
To message queue name Library	If the Action is *MSGQBRK or *MSGQINF, then specify the name and library of the Message Queue to send the message to.
Log host	If the Action is *SYSLOG or *PTGLOG, then specify the host name or IP address of the log server.
Log source port	Valid for *SYSLOG action type. Specify the local port to use when connecting to the log server. The default syslog port is 514. **NOTE:** Note: When the local port is set to 0, the system will search for an available local port to use.
Log destination port	Valid for *SYSLOG and *PTGLOG action types. Specify the port for the log server. The default port for syslog servers is 514.

NOTE: If a Security Alert fails, then a message will be sent to QSYSOPR and an entry will be place in the audit log file.

Email Alerts

If you would like to send Alerts through email using the IBM i SMTP server, then you need to make sure that your system is configured properly. Example steps:

1. Run the following command:
   ADDDIRE USRID(INTERNET GATEWAY) USRD('Allow SNDDST to send INTERNET Mail') SYSNAME(INTERNET) MSFSRVLVL(*USRIDX) PREFADR (NETUSRID *IBM ATCONTXT)
2. Change the mail distribution attributes by running the following command:
   CHGDSTA SMTPRTE(INTERNET GATEWAY)
3. A directory entry is required for each user that may potentially send email (using the SNDDST command) as a Security Alert. Example:
   ADDDIRE USRID(USERNAME SYSTEMNAME) USRD('User name') USER (USERNAME)
4. Run the SNDDST command to send a test email, and then verify that it was received. Example:
   SNDDST TYPE(*LMSG) TOINTNET(username@abc.com) DSTD('Test Email Subject') LONGMSG('Test Message Text') SUBJECT(*DOCD)

> **NOTE:** Consult with your IBM i administrator before making any changes.

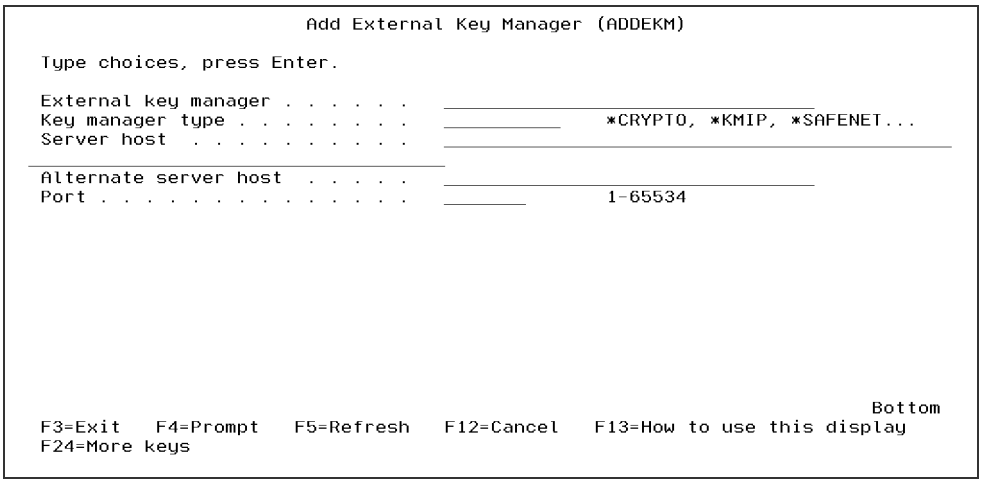
Add External Key Manager Entry (ADDEKM)

The Add External Key Manager (ADDEKM) command allows authorized users to define the properties for an External Key Manager (EKM). After the EKM is added, you can then use the CRTSYMKEY command to create an entry that refers to a remote key within that EKM.

> **NOTE:** Any maintenance to the External Key Managers is logged into an audit file.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the "Maintain key policy and alerts" authority setting



How to Get There

From the External Key Manager Menu, choose option **2**. Or, submit the command **ADDEKM**.

Options

Manager id (EKMGRID)

Indicate the unique name of the entry up to 30 characters.

Rules for key manager identifier:

- The key manager identifier is a descriptive name.
- The key manager identifier cannot contain spaces or certain special characters.
- The key manager identifier can contain underscore characters.
- The key manager identifier is not case sensitive. It will be stored in upper case.

Manager type (MGRTYPE)

Indicate the type of External Key Manager (EKM).

The possible values are:

***CRYPTO** Utilize the Powertech Encryption key management solution.
***KMIP** Utilize a Key Server that works with the Key Management Interoperability Protocol (KMIP) standard.
***SAFENET** Utilize the SafeNet key management solution.
***VORMETRIC** Utilize the Vormetric key managment solution.

Server host (SRVHOST)

Specify the host name or IP address of the External Key Manager.

Alternate Server host (ALTSRVHOST)

Specify the alternate host name or IP address of the External Key Manager.

The Server Host (SRVHOST) will be used first when connecting to the External Key Manager. If the connection attempt fails and the operation is a retrieve key or Verify host operation then the Alternate Server Host (ALTSRVHOST) will be used for the second connection attempt. If the second connection attempt fails then an error message will be returned to the caller.

A create key operation will not use the Alternate Server Host.

Server port (SRVPORT)

Specify the port to use to connect to the External Key Manager.

User profile (USER)

Specify the User profile for signing in to the External Key Manager.

User password (PASSWORD)

Specify the password to sign into the External Key Manager.

Domain (DOMAIN)

If the key manager type is *VORMETRIC, then specify the domain name of the external key server.

Use SSL (SSL)

Indicate if Secure Sockets Layer (SSL) encrpytion should be used to connect to the external key manager.

The possible values are:

***YES** Use SSL to connect to the external key manager.
***NO** Do not use SSL to connect to the external key manager.

Applcation Id (APPID)

When using an SSL connection, optionally specify the Application id from the IBM i Digital Key Manager that links to the certificate to use.

KMIP connection type (KMIPCONTYP)

Indicate the type of connection to use to connect to the KMIP server.

The possible values are:

***HTTP** HTTPS posts will be used when communicating with the KMIP server.
***TCP** SSL over TCP will be used when communicating with the KMIP server.

KMIP encoding method (KMIPENCMTH)

Indicate the encoding method to use when communicating with the KMIP server.

The possible values are:

***XML** XML encoding will be used to communicate with the KMIP server. XML can only be used with KMIP connection type of HTTPS.
***TTLV** TTLV encoding will be used to communicate with the KMIP server. TTLV can be used with either TCP or HTTP communication types.

Add Field Encryption Entry (ADDFLDENC)

The ADDFLDENC command allows authorized users to add a new entry into the Field Encryption Registry.

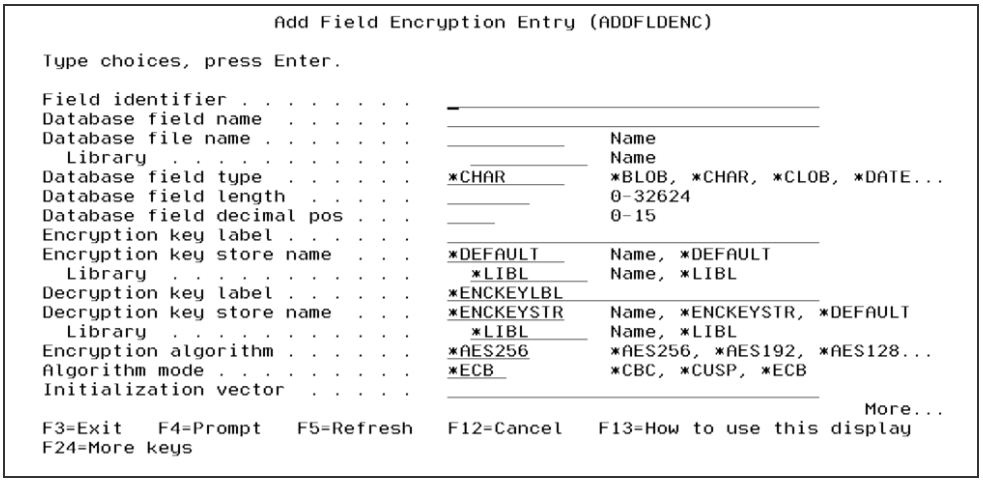
> **NOTE:** The ADDFLDENC command only adds the field entry settings into the registry. It will not cause any action to be performed on the actual database field in the file. The field will not be activated for encryption until the ACTFLDENC (Activate Field Encryption) command is executed.

The following users can use the ADDFLDENC command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)

- A Key Officer that has a *YES specified for the “Maintain Field Enc. Registry” authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object, which contains the Field Encryption Registry, and *USE authority to the library that contains the Key Store.



How to Get There

From the Work with Field Encryption Registry (WRKFLDENC) panel, press **F6**, Add. Or, prompt (**F4**) the command **CRYPTO/ADDFLDENC**.

Field Descriptions

Field identifier (FLDID)

Indicate the unique name of the entry up to 30 characters.

Rules for field identifier:

- The field identifier does not have to be the same name as the database field name to encrypt.
- The field identifier cannot contain spaces or certain special characters.
- The field identifier can contain underscore characters.
- The field identifier is not case sensitive. It will be stored in upper case.

Database field name (DBFLD)

Indicate the actual name of the database field to register for encryption. A field name up to 30 characters is supported.

The possible values are:

field-name The name of the field (or column name) in the database file to encrypt.
***REMOTE** The field is located in a remote database file (not local). Powertech Encryption will manage the encryption and storage of the field values through Tokenization. See Tokenization for Centralizing Sensitive Data for more information.

Database file name (DBFILE)

Indicate the name and library of the database file which contains the field to register.

Database field type (DBFLDTYP)

Indicate the data type of the database field.

The possible values are:

***BLOB** The field is a binary large object type.
***CHAR** The field is character type. For *hex fields, specify *CHAR in the encryption entry and use the 'actual' field length of the field definition in the file. For *varchar, *graphic, and *vargraphic fields, specify *CHAR in the encryption entry, and use the 'buffer length' for the field (in DSPFFD for the file) as your length.
***CLOB** The field is a character large object type.
***DATE** The field is a date data type. This option is only valid when using a DB2 Field Procedure to automatically encrypt/decrypt the field values.
***DEC** The field is decimal type. This includes packed decimal and zoned decimal types. If using a DB2 Field Procedure to automatically encrypt/decrypt the field values, then *DEC can also be used to indicate an Integer data type.
***TIME** The field is a time data type. This option is only valid when using a DB2 Field Procedure to automatically encrypt/decrypt the field values.
***TIMESTAMP** The field is a timestamp data type. This option is only valid when using a DB2 Field Procedure to automatically encrypt/decrypt the field values.

While defining a field of the type ‘Graphic’ or ‘VarGraphic’, choose *CHAR for Database Field Type. The Database Field Length should reflect the Buffer Length of the field (as shown in DSPFFD for this file/field) as opposed to the defined field length.

> **NOTE:** The field length is not edited until the entire list of parameters is entered for the field encryption entry. An incorrect field length value will likely result in a CRE0927 or CRE0387 error, which indicates the problem in your definition as well as the field length expected.

Database field length (DBFLDLEN)

Indicate the length of the values within the database field to encrypt.

For *CHAR type fields, the maximum allowable length is 32624.

For VARCHAR, the length parameter must be the length of the DB2 field plus 2 bytes (to accommodate the end-of-string delimiter).

For *DEC type fields, the maximum allowable length is 30.

If not using a DB2 Field Procedure and if you only want to encrypt a left portion of an alphanumeric field, then you can specify a length that is less than the actual field length.

> WARNING: If you specify a length that is less than the actual database field length, then any remaining bytes in the field will be cleared during the field activation process.

Database field decimal pos (DBFLDDEC)

For *DEC type database fields, indicate the number of decimal positions (scale) in the field. Allowable range is 0 to 4.

Encryption key label (ENCKEYLBL)

Indicate the label of the initial key to use for encrypting the field values.

Encryption key store name (ENCKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for encryption of the field.

The users (or user groups) which need to encrypt values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

***LIBL** Locate the Key Store within the library list.

Decryption key label (DECKEYLBL)

Indicate the label of the initial key to use for decrypting the field values.

The possible values are:

decryption-key-label Indicate the label of the key to use for decryption.

> WARNING: If specifying a different key label than the label specified for encryption, then that decryption key should contain the same key value as the encryption key.

***ENCKEYLBL** Use the same label as specified on the ENCKEYLBL parameter.

Decryption key store name (DECKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the field.

The users (or user groups) that need access to the decrypted values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***ENCKEYSTR** Use the same Key Store as specified on the ENCKEYSTR parameter.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Encryption algorithm (ALGORITHM)

Indicate the encryption algorithm to use for encrypting and decrypting the database field values.

This should match the algorithm used to create the Symmetric Key specified on the ENCKEYLBL parameter.

The possible values are:

***AES256** Use Advanced Encryption Standard (AES) algorithm and a 256 bit key.

***AES192** Use Advanced Encryption Standard (AES) algorithm and a 192 bit key.
***AES128** Use Advanced Encryption Standard (AES) algorithm and a 128 bit key.
***TDES** Use Triple Data Encryption Standard (TDES) algorithm.

Algorithm mode (MODE)

Indicate the mode to use within the encryption algorithm.

The possible values are:

***CBC** Mode is Cipher Block Chaining (CBC). This is a block-based mode. When using this mode with the AES algorithm, the length of the encrypted data will be a minimum of 16 bytes long. Its block-based length will be divisible by 16 or 24. When using this mode with the TDES algorithm, the length of the encrypted data will be a minimum of 8 bytes long. Its block-based length will be divisible by 8. With CBC mode, you can optionally specify an Initialization Vector (IV) to use as an additional input to the encryption algorithm.

***CUSP** Mode is Cryptographic Unit Support Program (CUSP). This is a stream-based mode, which means that the length of the encrypted data will equal the length of the input data. This mode is useful if the field data is not divisible by a block length of 16 or 24 (for AES), and if you want to store the encrypted values in your existing field (if not using DB2 Field Procedures). With CUSP mode, you can optionally specify an Initialization Vector (IV) to use as an additional input to the encryption algorithm.

***ECB** Mode is Electronic Code Book (ECB). This is a block-based mode. When using this mode with the AES algorithm, the length of the encrypted data will be a minimum of 16 bytes long. Its block-based length will be divisible by 16 or 24. When using this mode with the TDES algorithm, the length of the encrypted data will be a minimum of 8 bytes long. Its block-based length will be divisible by 8. With ECB mode, you cannot specify an Initialization Vector.

Initialization vector (INITVECTOR)

Optional. The Initialization Vector (IV) is an arbitrary value that you can enter, which will be used as an additional input to the encryption algorithm. Therefore, the encrypted output is dependent on the combination of the Initialization Vector, Encryption Key and the Plain Text (the data you want to encrypt).

The Initialization Vector is allowed for *CBC and *CUSP algorithm modes.

The length of the Initialization Vector must not exceed the block length, which is calculated as:

- 32 bytes for *AES algorithm and *CUSP mode
- 16 or 24 bytes for *AES algorithm and *CBC mode
- 8 bytes for *TDES algorithm and *CBC mode

Mask option (MASKOPT)

Indicate the mask option to use for the field when the masked value is requested on a decrypt operation.

The possible values are:

***NONE** No masking is performed.

***OPTION1** Exact positions within the field value can be shown or masked using the FLDMASK parameter.

***OPTION2** Only a specified number of digits or characters are shown on the left and right sides of the field using the DIGLEFT and DIGRIGHT parameters. Specify the masking character to use in the FLDMASKV parameter. When using field procedures, you must specify a masking character of 1-9, as only numbers are allowed in numeric fields. Examples when DIGLEFT(4) and DIGRIGHT(4) is specified with FLDMASKV(#) when using field procedures:

- String ' 1234567890123456' is masked as ' 1234999999993456'
- String '1234567890123456 ' is masked as '1234999999993456 '
- Numeric 001234567890123456 is masked as 001234999999993456
- Numeric 1234567890123456 is masked as 1234999999993456

If you are not using field procedures, a non-numeric character can be used, for example:

- String ' 1234567890123456' is masked as ' 1234########3456'
- String '1234567890123456 ' is masked as '1234########3456 '
- Numeric 001234567890123456 is masked as 001234########3456
- Numeric 1234567890123456 is masked as 1234########3456

***OPTION3** Used for Date, Time and Timestamp fields. The value entered will be used as the mask value. The mask value must be a proper Date, Time or Timestamp value for the field being masked. It is recommended to specify a mask value that does not conflict with an existing value in your database. For instance, for a date field, you may want to specify a mask value with a high value date of 9999-12-31. The format of the mask value must be in the format of the field format. For example if the "Date Format" of a field is *ISO, then the format of the masked value must be *ISO. Examples:

- Date '9999-12-31'
- Time '24.00.00'
- TimeStamp '9999-12-31-24.00.00.000000'

NOTE: When specifying *OPTION3, you should not specify a not-authorized fill value.

Field mask (FLDMASK)

Indicate the masking format to apply to the field when the masked value is requested on a decrypt operation. Valid for MASKOPT(*OPTION1) and MASKOPT(*OPTION3) masking.

- For MASKOPT(*OPTION1) masking

Specify the number 9 in a position to show the underlying value for that position. Specify any other character (including spaces) or number in a position to mask the underlying value for that position.

For example, if a mask of '**********9999' is specified for a credit card number, then a sample of a masked credit card number would be '**********1234'.

As another example, if a mask of '##99##999' is specified for an account number, then a sample of a masked account number would be '##76##541'.

When the field type is numeric the whole number is masked. The decimals values are not. The Mask must not be longer than the whole number length. When using field procedures the mask value must be numeric. For example, if a mask of '779977999' is specified for an account number, then a sample of a masked account number would be '774577541'.

- For MASKOPT(*OPTION3) masking

You must enter in a valid Date, Time or Timestamp value. This will be the value used as the mask.

Examples:

- Date '9999-12-31'
- Time '24.00.00'
- TimeStamp '9999-12-31-24.00.00.000000'

Used for Date, Time and Timestamp fields. The value entered will be used as the mask value. The mask value must be a proper Date, Time or Timestamp value for the field being masked.

It is recommended to specify a mask value that does not conflict with an existing value in your database. For instance, for a date field, you may want to specify a mask value with a high value date of 9999-12-31.

The format of the mask value must be in the format of the field format. For example if the "Date Format" of a field is *ISO, then the format of the masked value must be *ISO.

Examples:

- Date '9999-12-31'
- Time '24.00.00'
- TimeStamp '9999-12-31-24.00.00.000000'

> **NOTE:** When specifying *OPTION3, you should not specify a not-authorized fill value.

Char/Digits to show on left (DIGLEFT)

When a masked value is requested on a decrypt operation, indicate the number of characters or digits to show on the left side of the field value. Valid for MASKOPT(*OPTION2) masking.

For a character field, any leading blank characters will be ignored when performing the masking. For a decimal field, any leading zeros will be ignored.

Char/Digits to show on right (DIGRIGHT)

When a masked value is requested on a decrypt operation, indicate the number of characters or digits to show on the right side of the field value. Valid for MASKOPT(*OPTION2) masking.

For a character field, any trailing blank characters will be ignored when performing the masking.

Masking Value (FLDMASKV)

The value to be used as the masking character or number. Valid for MASKOPT(*OPTION2) masking. When masking a numeric field and using DB2 Field Procedures, the mask value must be a number between 0 and 9. When masking a character field, or when masking a numeric field and not using DB2 Field Procedures, the mask value can be any character or number.

Auth. list for full value (AUTLDEC)

Indicate the IBM i Authorization List that should be used to determine which users have authority to the full field values on decrypt operations.

This Authorization List will be used in field decryption APIs and DB2 Field Procedures.

The possible values are:

authorization-list-name Indicate the name of the Authorization List. An Authorization List can be created with the IBM i command CRTAUTL. The users (or user groups) which need access to the decrypted values will need at least (*USE) authority to the Authorization List. Additionally the users (or user groups) which need access to the decrypted values are required to have at least *USE authority to the Key Store object which holds the Decryption Key.

***NONE** An Authorization List should not be used by the decrypt operations. Therefore the user can gain access to the fully decrypted values as long as they have at least *USE authority to the Key Store which holds the Decryption Key.

Auth. list for masked value (AUTLMASK)

Indicate the IBM i Authorization List that should be used to determine which users have authority to the masked field values on decrypt operations.

This Authorization List will be used in field decryption APIs and DB2 Field Procedures.

The possible values are:

authorization-list-name Indicate the name of the Authorization List. An Authorization List can be created with the IBM i command CRTAUTL. The users (or user groups) which need access to the masked values will need at least (*USE) authority to the Authorization List. Additionally the users (or user groups) which need access to the masked values are required to have at least *USE authority to the Key Store object which holds the Decryption Key.

***NONE**
An Authorization List should not be used by the decrypt operations. Therefore the user can gain access to the masked values as long as they have at least *USE authority to the Key Store which holds the Decryption Key.

Auth. list caching (AUTLCACHE)

Indicate if the permissions for authorization lists are 'cached' in memory.

The possible values are:

***YES** Caching will occur. When a field decrypt operation is performed, the permissions for the authorization lists will be saved (in memory) and used in future authority checks [for decrypt operations] within the job. This caching option provides the best performance.

> **NOTE:** In order to recognize any permission changes to the authorization lists, the jobs [that are performing decrypt operations] will need to be restarted.

***INT** Caching occurs intermediately. The authorization lists are checked every 60 seconds between decryption operations. This option is a balanced approach between *YES and *NO. Jobs do not need to be restarted in order to identify permission changes. Expect a 60 second delay instead.

> **NOTE:** The 60 second window does not begin once the permissions are changed. The window occurs between decryption operations instead. For example, data is read and decrypted based on permissions. On the first row retrieval, the 60 second window starts. If the permissions are changed and data is retrieved again within that 60 second window, the new permission changes will not yet be reflected.

***NO** Caching will not occur. The permissions to the authorization lists will be checked each time a decrypt operation is performed. This option is useful when you want changes to the authorization lists to be immediately recognized by jobs that are performing decrypt operations, or if you want to take advantage of program adopted authority when determining permissions to an authorization list.

Not authorized fill value (NOTAUTHFV)

Indicate the 1-byte value to fill the returned value on a decryption request (from a DB2 Field Procedure or a Powertech Encryption 'auth' API) if the user is not authorized to either the full or masked authorization lists.

For instance, if the fill value is '9' and the field length is 7, then the value of '9999999' will be returned on an unauthorized decryption request.

NOTE:

- The fill value is required when a DB2 Field Procedure is utilized and the return value (FLDPROCOPT) is set to *AUTH.
- If the field type is *CHAR, then the fill value can be a number, letter or special character (e.g. #, *, %).
- If the field type is *DEC, then the fill value can be a number from 1 through 9 if a DB2 Field Procedure is being utilized, otherwise it can be number from 0 through 9.
- The fill value is not allowed for field types of *DATE, *TIME and *TIMESTAMP.

Store values in external file (STREXTFILE)

Indicate if the encrypted field values should be stored in a separate external file.

The encrypted values must be stored in an external file if not using a DB2 Field Procedure and if any of the following conditions are met:

- If the field type is *DEC.
- For *AES128, *AES192 and *AES256 algorithms with *CBC or *ECB modes: If the field type is *CHAR and the length specified for DBFLDLEN is not divisible by 16 or 24.
- For *TDES algorithm: If the field type is *CHAR and the length specified for DBFLDLEN is not divisible by 8.

The possible values are:

***YES** Store the encrypted field values in an external file.
***NO** Store the encrypted values in the existing database field.

External file name (EXTFILE)

Indicate the name and library of the external physical file which will be created to contain the encrypted field values.

The possible values are:

external-file-name Indicate the name of the external physical file object to store the encrypted values. This object name must NOT already exist.
***GEN** The process will automatically generate the external physical file object name, which uses the naming convention of CRXXnnnnn (where nnnnn is a sequential number from 1 to 99999).

The possible library values are:

library-name Indicate the name of the library to create the external physical file in.
***DBLIB** Specify *DBLIB for the library name to indicate that the external physical file should be created in the same library as where the database file (which contains the field to encrypt) resides.

The external physical file will be keyed by the Field Identifier

(XXFLDID) and the Index Number (XXINDEX).

External logical file name (EXTFILEL)

Indicate the name and library of the logical file which will be built over the external physical file, which will be keyed by the

Field Identifier (XXFLDID) and the Encrypted Value (XXVALUE).

The possible values are:

***NONE** The logical file will not be created.
external-logical-file-name Indicate the name of the logical file object over the external physical file.

This object name must NOT already exist.

***GEN** The process will automatically generate the logical file object name, which uses the naming convention of CRXXnnnnnL (where nnnnn is a sequential number from 1 to 99999).

The possible library values are:

library-name Indicate the name of the library to create the external logical file in.
***DBLIB** Specify *DBLIB for the library name to indicate that the external logical file should be created in the same library as where the database file (which contains the field to encrypt) resides.

Store hash for security check (EXTSTRHASH)

Indicate if a HASH value should be stored for each record in the external file.

The possible values are:

***YES** A HASH value will be stored for each record in the external file. The HASH value is an encrypted value which is calculated based on the Field identifier, Index number and Key id for each record. When a record is retrieved from the external file, it will recalculate the HASH and compare it to the stored HASH. A mismatch in the HASH values will indicate that an unauthorized change was made to the external file record's Field identifier, Index number and/or Key id.

***NO** A HASH value will not be stored in the external file. Store last retrieved user/time (EXTSTRRTV) Indicate if the user id and timestamp of when a field's value was last retrieved(decrypted) should be stored for each record in the external file.

> NOTE: You can additionally turn on audit logging for a Decryption key to log each time the key is used to decrypt data.

The possible values are:

***YES** Stores the last retrieved user/time in the external file.
***NO** Does not store the last retrieved user/time in the external file.

Align index number (INDEXALIGN)

When encrypting a character (alphanumeric) type field which is stored in an external file, then indicate how the external index number should be aligned in the field.

The possible values are:

***LEFT** The index number should be aligned on the left side of the database field.
***RIGHT** The index number should be aligned on the right side of the database field.

Index padding character (INDEXPAD)

When encrypting a character (alphanumeric) type field, indicate a padding character to place in the unused positions of the field.

For instance, if a padding character of "*" is specified with an alignment of *LEFT, then a 10 position field value with an index of 895 would appear as "895*******".

The padding character cannot be a number, a single quote (') or a dash (-).

Last index number storage (LSTINDSTG)

Indicate the object type to store the 'last index number used'. Each time a record is written (inserted) to the external file, the 'last index number used' is retrieved from the object, increased by 1, assigned to the new record and saved back to the object.

The possible values are:

***FLDREG** Store the last index number used in the field registry object, which is a validation list (*VLDL) with the name of CRVL002.

***PF** Store the last index number used in a physical file with the name of CRPF002. A physical file may be easier to replicate (than a *VLDL) with a High Availability tool. A physical file will also provide better performance (than a *VLDL) when a high volume of inserts occurs for the field, due to the file's ability to handle locks more efficiently.

> **NOTE:** If the CRPF002 physical file does not exist, it can be created using the CRTPF command. The source is located in the CRPF002 member in the source file of CRYPTO/QDDSSRC.

Use triggers to auto encrypt (USETRG)

Indicate if SQL triggers should be created over the database file, which will automatically encrypt the database field values without having to change your applications.

Note that SQL triggers are not allowed for encryption if a DB2 Field Procedure is already specified with USEFLDPROC(*YES).

The possible values are:

They are mutually exclusive.

***YES** SQL triggers will be created over the database file.
***NO** SQL triggers will not be created. You will need to use a DB2 Field Procedure or call Powertech Encryption encryption APIs from within your application programs to encrypt the field values.

Trigger name for inserts (INSTRG)

Indicate the name and library of the trigger to create. This trigger will be used to automatically encrypt the field value when records are inserted (added) into the database.

The possible values are:

trigger-name Specify the name of the trigger to create. Rules for trigger names:

- The trigger name cannot exceed 80 characters in length.
- The trigger name cannot be the same name as a trigger that is already on the database file.
- The trigger name cannot contain spaces or certain special characters.
- The trigger name can contain underscore characters.
- The trigger name is not case sensitive. It will be stored in upper case.

***GEN** Specify *GEN to have the process automatically generate the trigger name, which uses the naming convention of "FILENAME_FIELDNAME_CryptoInsert".

The possible library values are:

library-name Indicate the name of the library to create the trigger object in.
***DBLIB** The trigger will be created in in the same library as where the database file resides.

Trigger name for updates (UPDTRG)

Indicate the name and library of the trigger to create. This trigger will be used to automatically encrypt the field value when records are updated in the database.

This is a column trigger, so it is only called when the particular field value is changed.

The possible values are:

trigger-name **Specify the name of the trigger to create.** Rules for trigger names:

- The trigger name cannot exceed 80 characters in length.
- The trigger name cannot be the same name as a trigger that is already on the database file.
- The trigger name cannot contain spaces or certain special characters.
- The trigger name can contain underscore characters.
- The trigger name is not case sensitive. It will be stored in upper case.

***GEN** Specify *GEN to have the process automatically generate the trigger name, which uses the naming convention of "FILENAME_FIELDNAME_CryptoUpdate".

The possible library values are:

library-name Indicate the name of the library to create the trigger object in.
***DBLIB** The trigger will be created in in the same library as where the database file resides.

Trigger name for deletes (DLTTRG)

This is only valid if an external file is used to store the encrypted values: Indicate the name and library of the trigger to create.

This trigger will be used to automatically remove the encrypted field value (record) from the external file when a database record is deleted.

The possible values are:

trigger-name Specify the name of the trigger to create. Rules for trigger names:

- The trigger name cannot exceed 80 characters in length.
- The trigger name cannot be the same name as a trigger that is already on the database file.
- The trigger name cannot contain spaces or certain special characters.
- The trigger name can contain underscore characters.
- The trigger name is not case sensitive. It will be stored in upper case.

***GEN** Specify *GEN to have the process automatically generate the trigger name, which uses the naming convention of "FILENAME_FIELDNAME_CryptoDelete".

The possible library values are:

library-name Indicate the name of the library to create the trigger object in.
***DBLIB** The trigger will be created in in the same library as where the database file resides.

Trigger exit type (TRGEXITTYP)

Indicate if the triggers should call a custom exit program before performing any inserts, updates or deletes of the field value.

Listed below are some examples of how a Trigger exit program could be utilized:

- To write out additional audit entries in the audit journal file.
- To direct Powertech Encryption to not process (ignore) the requested insert/update/delete of the field value based on custom criteria, such as the user id or application performing the request.
- To perform additional custom logic.

A trigger exit program can be written in RPG, COBOL or C on the System i. Source examples of RPG trigger programs are provided in the members of TRGEXTPGM and TRGEXTSRV within the source file of CRYPTO/QRPGLESRC.

The possible values are:

***NONE** No Trigger exit program is used.
***PGM** The Trigger exit is a Program (*PGM) object.
***SRVPGM** The Trigger exit is a Service Program (*SRVPGM) object.

Trigger exit program (TRGEXITPGM)

Indicate the name and library of the trigger exit program to call.

program-name Enter the name of the program or service program.

The possible library values are:

library-name Enter the name of the library where the program is located.
***LIBL** Locate the program within the library list.

Trigger exit procedure (TRGEXITPRC)

Indicate the name of the procedure to call if the TRGEXITTYP is *SRVPGM.

Use DB2 field procedure (USEFLDPROC)

Indicate if a DB2 Field Procedure will be used to automatically encrypt/decrypt the field values, which is an alternative approach to using triggers and API calls. A DB2 Field Procedure also allows storing the 'encoded' encrypted values within the existing file, which is especially useful for numeric fields. [You will not need to create a separate external file to store the values for numeric fields.]

DB2 Field Procedures are available in IBM i version V7R1 and higher.

> **NOTE:** DB2 Field Procedures are not allowed if SQL triggers are already specified with USETRG(*YES). They are mutually exclusive.

Before using DB2 Field Procedures in a production environment, read the Installation Guide and User Guide to understand the potential performance issues and risks.

The possible values are:

***YES** A DB2 Field Procedure will be used to automatically encrypt and decrypt the field values.
***NO** A DB2 Field Procedure will not be used. You must use triggers or APIs to encrypt the values. APIs must be used to decrypt the values.

Field procedure return value (FLDPROCOPT)

Indicate which field value is returned (based on user permissions) from the DB2 Field Procedure to the application on a read operation.

The possible values are:

***FULL** Returns the fully decrypted value if the user has at least *USE rights to the authorization list specified on the AUTLDEC parameter (or if *NONE is specified on that parameter). Otherwise an error with the message id of CPF504D will be generated in the application performing the read. The *FULL option is available for *CHAR, *DEC, *DATE, *TIME and *TIMESTAMP data types.

***AUTH** For character (*CHAR) fields up to 30 bytes in length, this option returns either: 1) the full value if the user has at least *USE rights to the authorization list specified on the AUTLDEC parameter (or if *NONE is specified on that parameter) or 2) the masked value if the user has at least *USE rights to the authorization list specified on the AUTLMASK parameter (or if *NONE is specified on that parameter) or 3) the fill value if the user does not have at least *USE rights to either authorization list. For decimal/numeric (*DEC) fields or character fields over 30 bytes in length, this option returns either: 1) the full value if the user has at least *USE rights to the Authorization List specified on the AUTLDEC parameter (or if *NONE is specified on that parameter) or 2) the numeric masking character and the left/right digit (to show) logic (MASKOPT *OPTION2) if the user has at least *USE rights to the authorization list specified on the AUTLMASK parameter (or if *NONE is specified on that parameter). Otherwise (if not authorized) the fill value is returned. The fill value is defined in the Change Field Encryption Entry panel (CHGFLDENC).

> **NOTE:** A zero value in a numeric field is displayed in clear text and is not masked with the Mask character or Not Authorized character.

Decryption Accelerator (PERFACCEL)

Allows Powertech Encryption for IBM i to apply methods to attempt to improve performance gains when possible with native i/o.

The possible values are:

***YES** See Appendix G: Decryption Accelerator Prerequisites and Limitations for considerations and prerequisites. Apply methods to attempt to improve performance gains when possible.
***NO** Do not apply methods to attempt to improve performance gains when possible.

Add Field Triggers (ADDFLDTRG)

The Add Field Triggers (ADDFLDTRG) command will recreate any SQL triggers that were removed with the RMVFLDTRG command.

ADDFLDTRG can only be used for *ACTIVE field entries which were configured in the Registry to use SQL triggers.

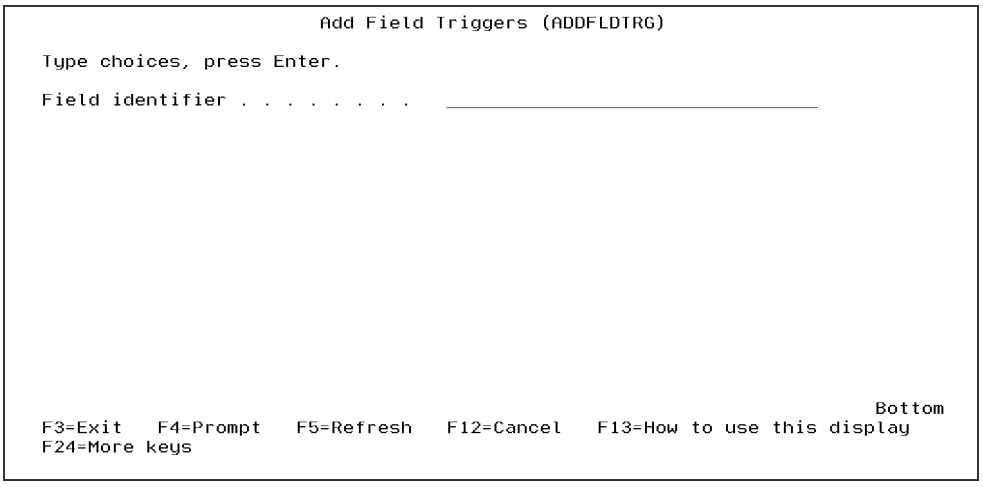
NOTE:
- ADDFLDTRG only recreates the SQL Triggers on the database file.
- Any field values, which existed before ADDFLDTRG was executed, will remain in their current form.
- The field entry will keep an *ACTIVE status in the Field Encryption Registry.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have the following object authorities:

- *USE authority for the CRVL002 *VLDL object which contains the Field Encryption Registry.
- Alter authority for the database file specified on the field entry.



How to Get There

From the Field Triggers Menu, choose option **2**.

Options

Field identifier (FLDID)

Enter the Field identifier to recreate the triggers for.

Add IFS Encryption Entry (ADDIFSENC)

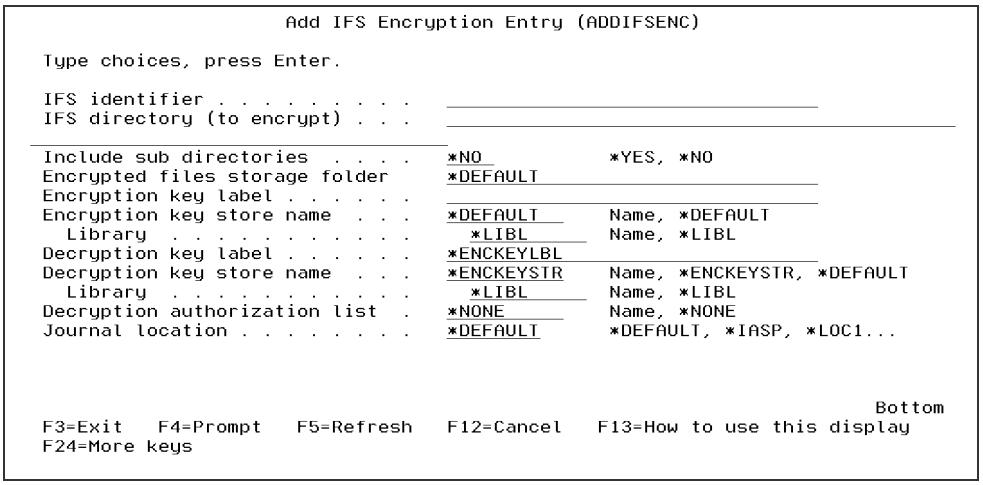
The ADDIFSENC command allows authorized users to add a new entry into the IFS Encryption Registry.

The following users can use the ADDIFSENC command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

> **NOTE:** The ADDIFSENC command only adds the IFS entry settings into the registry. It will not cause any action to be performed on the actual files in the directory(s). The IFS will not be activated for encryption until the ACTIFSENC (Activate IFS Encryption) command is executed.

This command requires that you have *CHANGE authority to the CRVL003 Validation List (*VLDL) object, which contains the IFS Encryption Registry.



How to Get There

From the IFS Encryption Menu, choose option **2**. Or, prompt (**F4**) the command **CRYPTO/ADDIFSENC**.

Options

IFS identifier (IFSID)

Indicate the unique name of the entry up to 30 characters.

Rules for IFS identifier:

- The IFS identifier does not have to be the same name as the directory or files to encrypt. It is simply used as a way to identify this entry within IFS registry.
- The IFS identifier cannot contain spaces or certain special characters.
- The IFS identifier can contain underscore characters.
- The IFS identifier is not case sensitive. It will be stored in upper case.

IFS directory to encrypt (SRCDIR)

Specify the path of the IFS directory containing the files to be encrypted.

The maximum size of the directory name is 256 bytes. The maximum size of any filename is 256 bytes.

For instance: '/HR/PayrollData'

Include subdirectories (INCSUBDIR)

Indicate if the files within the directory's subdirectories are to be encrypted.

The possible values are:

***YES** Files within the subdirectories will also be encrypted.
***NO** Files within the subdirectories will NOT be encrypted.

Encrypted files storage folder (TGTDIR)

Specify the path of the IFS directory to store the encrypted versions of the files.

If this directory does not exist, then it will be created.

If this is an existing directory, then it cannot contain existing files.

The maximum size of the directory name is 256 bytes.

The possible values are:

***DEFAULT** The encrypted versions of the files will be stored under the '/CryptoDisk' directory using the same directory name as specified for the SRCDIR parameter. For instance: '/CryptoDisk/HR/PayrollData'

ifs-directory-name Specify the full path to the IFS directory name to store the encrypted versions of the IFS files. For instance: '/Encrypted/HR/PayrollData'

Encryption key label (ENCKEYLBL)

Indicate the label of the symmetric key to use for encrypting the IFS files within the directory.

Encryption key store name (ENCKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for encrypting the IFS files.

The users which are writing or changing the IFS files will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Specify the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Decryption key label (DECKEYLBL)

Indicate the label of the symmetric key to use for decrypting the IFS files.

The possible values are:

decryption-key-label Indicate the label of the key to use for decryption.

> WARNING: If specifying a different key label than the label specified for encryption, then that decryption key should contain the same key value as the encryption key.

***ENCKEYLBL** Use the same label as specified on the ENCKEYLBL parameter.

Decryption key store name (DECKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the IFS files.

The users which are opening/reading the IFS files will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***ENCKEYSTR** Use the same Key Store as specified on the ENCKEYSTR parameter.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

***LIBL** Locate the Key Store within the library list.

Authorization list for decryption (AUTLDEC)

Indicate the IBM i Authorization List that should be used to determine which users have authority to decrypt the IFS files.

The possible values are:

authorization-list-name Indicate the name of the Authorization List. An Authorization List can be created with the IBM i command CRTAUTL. The users (or user groups) which need access to the decrypted IFS files will need at least (*USE) authority to the Authorization List.

***NONE** An Authorization List should not be used by the IFS decrypt operations. Therefore the user can gain access to the decrypted files as long as they have object authority to the IFS file and at least *USE authority to the Key Store which holds the Decryption Key.

Journal location (JRNLOC)

Indicate the location of the journal and related objects.

The possible values are:

***DEFAULT** The location for all related objects will be in the CRYPTO library. Also the name of the journal will be CRJNI001. No further changes will need to be made.

***IASP** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the IASP library designated. The following objects will need to be copied into the IASP library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRPFIFS2 PHYSICAL FILE
- CRVL003 VALIDATION LIST
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_IASP_CRPFIFS_LIBRARY
- IFS_IASP_CRPFIFS2_LIBRARY
- IFS_IASP_REGISTRY_LIBRARY
- IFS_IASP_JOURNAL_LIBRARY
- IFS_IASP_LAST_SEQ_DTAARA_LIBRARY
- IFS_IASP_SERVER_RUN_DTAARA_LIBRARY

***LOC1** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC1 library designated. The IFS Encryption Registry (CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC1 library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC1_CRPFIFS_LIBRARY
- IFS_LOC1_REGISTRY_LIBRARY
- IFS_LOC1_JOURNAL_LIBRARY
- IFS_LOC1_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC1_SERVER_RUN_DTAARA_LIBRARY

***LOC2** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC2 library designated. The IFS Encryption Registry (CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC2 library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC2_CRPFIFS_LIBRARY
- IFS_LOC2_REGISTRY_LIBRARY
- IFS_LOC2_JOURNAL_LIBRARY
- IFS_LOC2_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC2_SERVER_RUN_DTAARA_LIBRARY

***LOC3** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC3 library designated. The IFS Encryption Registry (CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC3 library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC3_CRPFIFS_LIBRARY
- IFS_LOC3_REGISTRY_LIBRARY
- IFS_LOC3_JOURNAL_LIBRARY
- IFS_LOC3_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC3_SERVER_RUN_DTAARA_LIBRARY

***LOC4** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC4 library designated. The IFS Encryption Registry (CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC4 library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC4_CRPFIFS_LIBRARY
- IFS_LOC4_REGISTRY_LIBRARY

- IFS_LOC4_JOURNAL_LIBRARY
- IFS_LOC4_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC4_SERVER_RUN_DTAARA_LIBRARY

***LOC5** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC5 library designated. The IFS Encryption Registry(CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC5 library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC5_CRPFIFS_LIBRARY
- IFS_LOC5_REGISTRY_LIBRARY
- IFS_LOC5_JOURNAL_LIBRARY
- IFS_LOC5_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC5_SERVER_RUN_DTAARA_LIBRARY

Add IFS Exit Point Programs (ADDIFSEXTP)

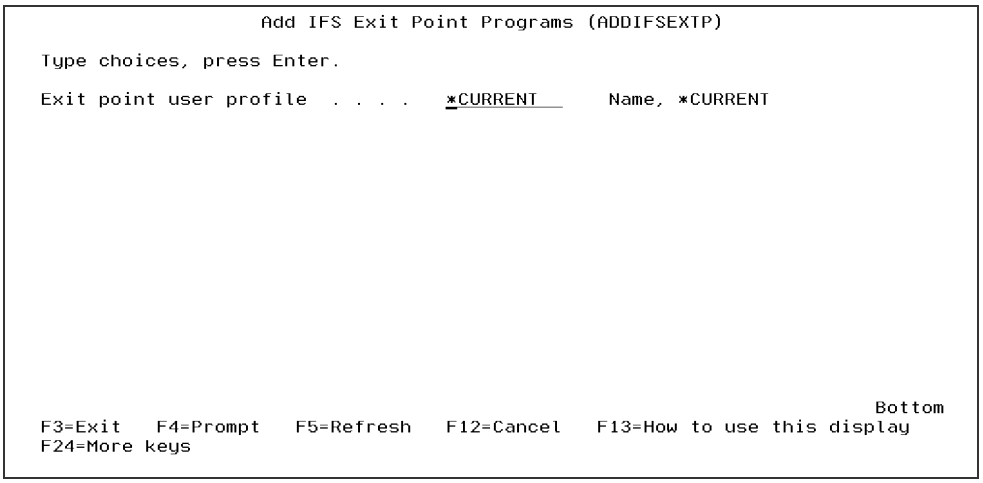
The Add IFS Exit Point Programs (ADDIFSEXTP) command will add the exit programs for the QIBM_QP0L_SCAN_CLOSE, QIBM_QP0L_SCAN_OPEN and QIBM_QPWFS_FILE_SERV exit points. These programs will capture IFS-related events in order to encrypt/decrypt the files as needed.

This command requires that you have *ALLOBJ (all object) and *SECADM (security administrator) special authorities to add exit programs to the registration facility.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

IMPORTANT: After adding the exit programs, they will be used for any new jobs. Any existing jobs must be restarted in order to use the exit programs.



How to Get There

From the IFS Utility Menu, choose option **3**, Add IFS Exit Point Programs. Or, prompt (**F4**) the command **CRYPTO/ADDIFSEXTP**.

Options

User profile (USRPRF)

The user profile under which the QIBM_QP0L_SCAN_CLOSE and QIBM_QP0L_SCAN_ OPEN exit programs will be called.

This user profile should have the following authorities.

- *USE authority to the exit program CRRP041, located in the CRYPTO library.
- *EXECUTE authority to the CRYPTO library.
- *ALL authority to the Directory(s) to encrypt and all the files in the directory(s).
- *ALL authority to the Directory(s) that hold the encrypted files and the files in the directory(s).

- *CHANGE authority to the files CRPFIFS, CRPFIFSL1, CRPFIFSL2, CRPFIFSL3, CRPFIFSL4, CRPFIFS2, CRPFIFSLOG Files which are located in the CRYPTO library.
- *CHANGE authority to the data areas CRDEBUG, CRLSTSEQ and CRSRVRUN, which are located in the CRYPTO library.
- *Use Authority to the CRJNI001 Journal and all Journal Receivers, which are located in the CRYPTO library.

If the user profile is not valid or accessible at the time the exit program is called, the action on the IFS file will be ignored, which may cause the file to NOT be encrypted or decrypted at the appropriate time.

The possible values are:

user-profile-name Specify a valid user profile name to run the exit programs under. This user profile must be enabled at the time the exit program is called.
***CURRENT** The current job's user profile is used to run the exit programs under.

Add Key Officer (ADDKEYOFR)

The Add Key Officer (ADDKEYOFR) command allows an authorized user to add a new Key Officer into the Symmetric Key Management System.

See Adding Key Officers in Getting Started.

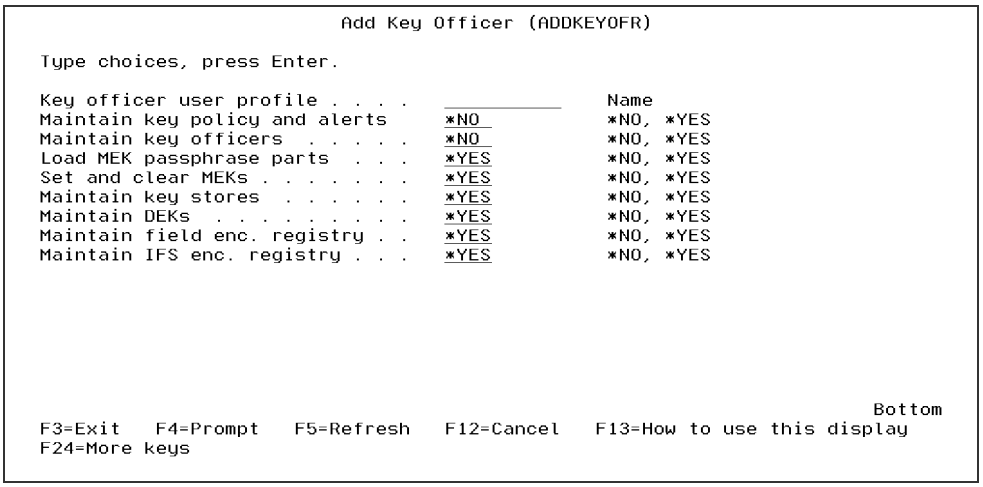
> **NOTE:** Any maintenance to the Key Officers is logged into an audit file.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key officers” authority setting

When a Key Officer is added the following occurs:

- If the User Profile is authorized to at least one option then the User Profile is added to the PCRADMIN Authorization List with *USE Authority.
- If the User is set to maintain any of the following: Key Policy, Key Officers, Load MEK Parts or Load MEK, then the User Profile is added to the CRVL001 object with *CHANGE Authority. The user profile running this command must have authority to run the ADDAUTLE command. The user profile running this command must have authority to run the GRTOBJAUT command.



How to Get There

From the Key Policy and Security Menu, choose option **11**, Add Key Officer. Or, prompt (**F4**) the command **CRYPTO/ADDKEYOFR**.

Field Descriptions

Key officer user profile (USRPRF)

Specify an existing user profile on the System i.

Maintain key policy and alerts (MNTPCYALR)

Indicate if the Key Officer can change the key policy settings and can add, change or delete Alerts.

The possible values are:

***YES** The Key Officer can change the key policy settings and can add, change or delete Alerts.
***NO** The Key Officer cannot change the key policy settings and cannot add, change or delete Alerts.

Maintain key officers (MNTKEYOFR)

Indicate if the Key Officer can add, change and remove other Key Officers.

The possible values are:

***YES** The Key Officer can add, change, and remove other Key Officers.
***NO** The Key Officer cannot add, change, and remove other Key Officers.

Load MEK passphrase parts (LODMEKPRT)

Indicate if the Key Officer can specify passphrase parts for a Master Encryption Key (MEK).

The possible values are:

***YES** The Key Officer can specify passphrase parts for a MEK.
***NO** The Key Officer cannot specify passphrase parts for a MEK.

Set and clear MEKs (MNTMEK)

Indicate if the Key Officer can set (generate) or clear a Master Encryption Key (MEK).

The possible values are:

***YES** The Key Officer can set or clear a MEK.
***NO** The Key Officer cannot set or clear a MEK.

Maintain key stores (MNTKEYSTR)

Indicate if the Key Officer can create Key Stores or translate Key Stores to other Master Encryption Keys (MEKs).

The possible values are:

***YES** The Key Officer can maintain Key Stores.
***NO** The Key Officer cannot maintain Key Stores.

Maintain DEKs (MNTDEK)

Indicate if the Key Officer can create, copy or delete Data Encryption Keys (DEKs)

The possible values are:

***YES** The Key Officer can maintain DEKs.
***NO** The Key Officer cannot maintain DEKs.

Maintain Field Enc. Registry (MNTFLDENC)

Indicate if the Key Officer can maintain the Field Encryption Registry.

The possible values are:

***YES** The Key Officer can maintain the Field Encryption Registry.
***NO** The Key Officer cannot maintain the Field Encryption Registry.

Maintain IFS Enc. Registry (MNTIFSENC)

Indicate if the Key Officer can maintain the IFS Encryption Registry and other automatic IFS Encryption settings. Reserved for future use.

The possible values are:

***YES** The Key Officer can maintain the IFS Encryption Registry and other automatic IFS Encryption settings.
***NO** The Key Officer cannot maintain the IFS Encryption Registry and other automatic IFS Encryption settings.

Change a Field Encryption Pending Key (CHGPNDKEY)

When using Field Procedures, the Change a Field Encryption Pending Key (CHGPNDKEY) command allows authorized users to change a Pending key in a Field entry that can be used in the next key rotation process.

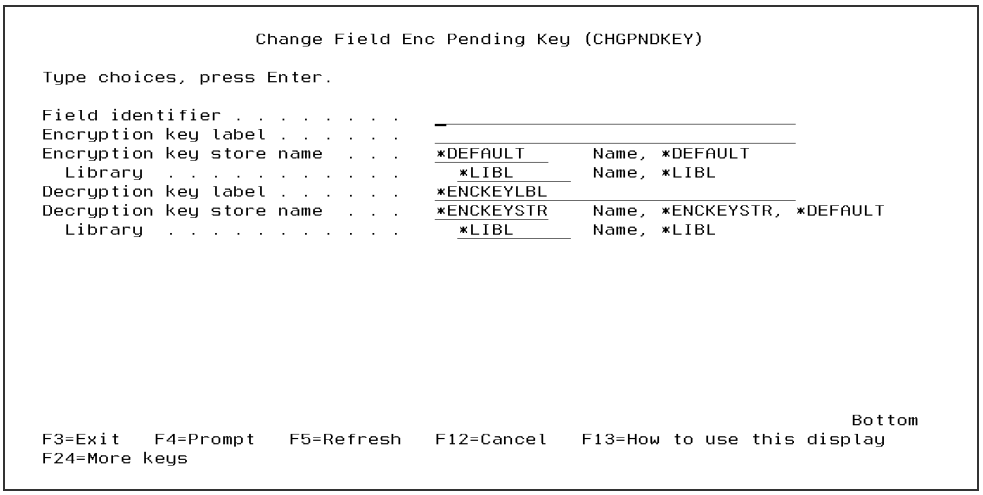
This command can be used for *ACTIVE field entries that use Field Procedures.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

The fields here are the same as those in the Add a Field Encryption Pending Key (ADDPNDKEY) panel.



How to Get There

On the File Field Encryption Menu, choose option **7**.

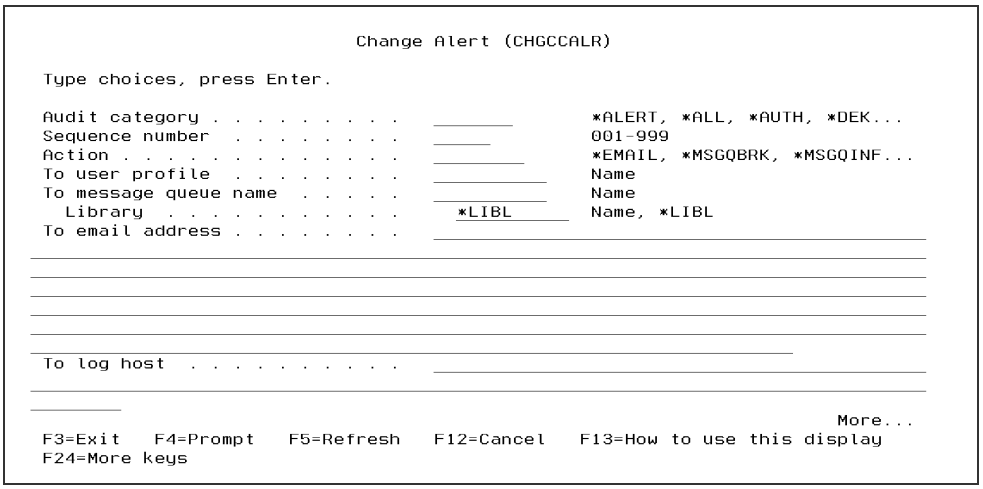
Change Alert (CHGCCALR)

The CHGCCALR command allows an authorized user to change the settings for a Security Alert entry.

> **NOTE:** Any maintenance to the Security Alerts is logged into an audit file.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key policy and alerts” authority setting



Do the following steps to change an alert:

1. Prompt (**F4**) the command **CRYPTO/CHGCCALR**.
2. Type in the audit category and sequence number, and then press enter.
3. The current Alert settings (parameter values) will be displayed.
4. Press **F1** on any parameter for complete online help text.
5. Press Enter after the parameter values are entered.

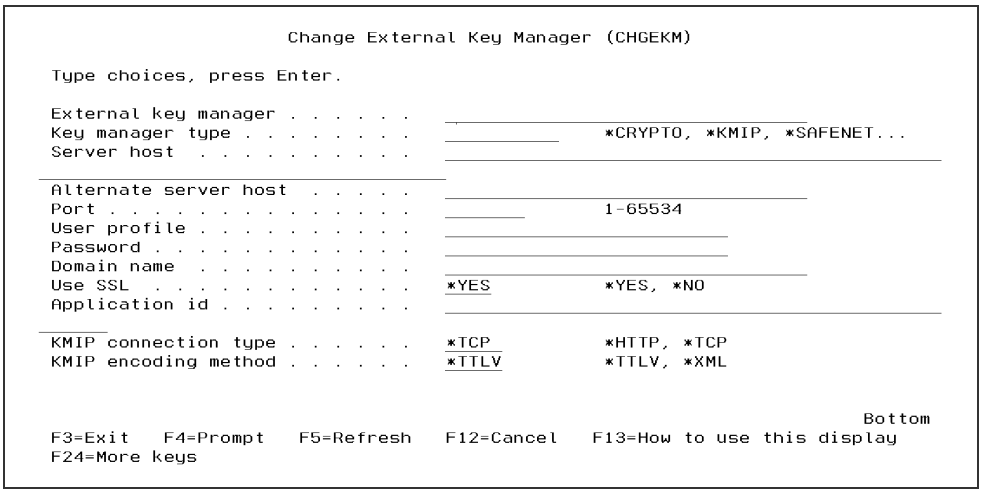
See the field descriptions on the Add Alert (ADDCCALR) for more information on the parameters.

Change External Key Manager (CHGEKM)

The Change External Key Manager (CHGEKM) command allows authorized users to change the properties for an EKM entry.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key policy and alerts” authority setting



How to Get There

From the External Key Manager Menu, choose option **3**. Or, submit the command **CHGEKM**.

Options

Manager id (EKMGRID)

Indicate the unique name of the entry up to 30 characters.

Rules for key manager identifier:

- The key manager identifier is a descriptive name.
- The key manager identifier cannot contain spaces or certain special characters.
- The key manager identifier can contain underscore characters.
- The key manager identifier is not case sensitive. It will be stored in upper case.

Manager type (MGRTYPE)

Indicate the type of External Key Manager (EKM).

The possible values are:

***CRYPTO** Utilize the Powertech Encryption key management solution.
***KMIP** Utilize a Key Server that works with the Key Management Interoperability Protocol (KMIP) standard.
***SAFENET** Utilize the SafeNet key management solution.
***VORMETRIC** Utilize the Vormetric key management solution.

Server host (SRVHost)

Specify the host name or IP address of the External Key Manager.

Alternate Server host (ALTSRVHOST)

Specify the alternate host name or IP address of the External Key Manager.

The Server Host (SRVHOST) will be used first when connecting to the External Key Manager. If the connection attempt fails and the operation is a retrieve key or verify host operation then the Alternate Server Host (ALTSRVHOST) will be used for the second connection attempt. If the second connection attempt fails then an error message will be returned to the caller.

A create key operation will not use the Alternate Server Host.

Server port (SRVPORT)

Specify the port to use to connect to the External Key Manager.

User profile (USER)

Specify the User profile for signing in to the External Key Manager.

User password (PASSWORD)

Specify the password to sign into the External Key Manager.

Domain (DOMAIN)

If the key manager type is *VORMETRIC, then specify the domain name of the external key server.

Use SSL (SSL)

Indicate if Secure Sockets Layer (SSL) encrpytion should be used to connect to the external key manager.

The possible values are:

***YES** Use SSL to connect to the external key manager.
***NO** Do not use SSL to connect to the external key manager.

Applcation Id (APPID)

When using an SSL connection, optionally specify the Application id from the IBM i Digital Key Manager that links to the certificate to use.

KMIP connection type (KMIPCONTYP)

Indicate the type of connection to use to connect to the KMIP server.

The possible values are:

***HTTP** HTTPS posts will be used when communicating with the KMIP server.
***TCP** SSL over TCP will be used when communicating with the KMIP server.

KMIP encoding method (KMIPENCMTH)

Indicate the encoding method to use when communicating with the KMIP server.

The possible values are:

***XML** XML encoding will be used to communicate with the KMIP server. XML can only be used with KMIP connection type of HTTPS.
***TTLV** TTLV encoding will be used to communicate with the KMIP server. TTLV can be used with either TCP or HTTP communication types.

Change Field Authorization Lists (CHGFLDAUTL)

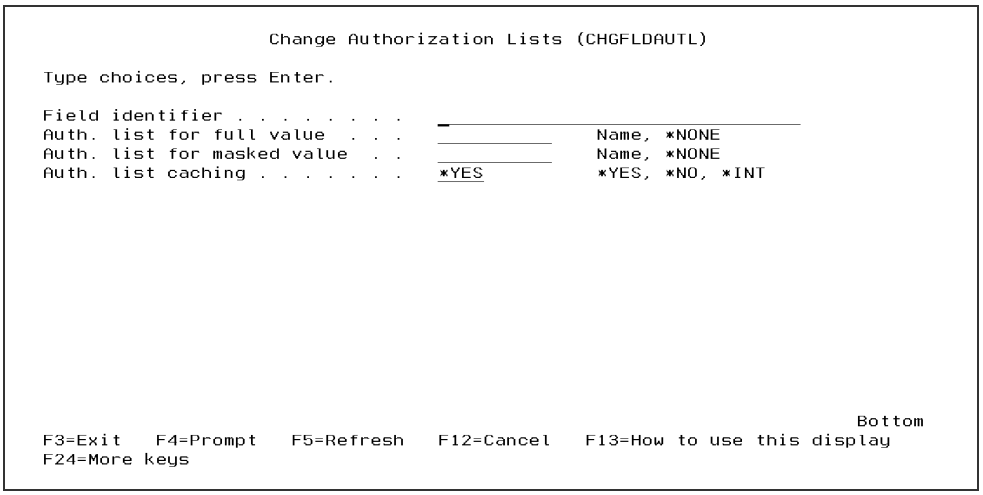
The Change Field Authorization Lists (CHGFLDAUTL) command allows authorized users to change the Authorization List settings for a field in the Field Encryption Registry.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires the user to have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

NOTE: The CHGFLDAUTL command only changes the field entry settings in the registry. It will not cause any action to be performed on the actual database field in the file.



How to Get There

From the Field Encryption Menu, choose option **5**. Or, prompt (**F4**) the command **CRYPTO/CHGFLDAUTL**.

Options

Field identifier (FLDID)

Indicate the Field identifier to change the Authorization Lists for.

Auth. list for full value (AUTLDEC)

Indicate the Authorization List that should be used by the field decryption APIs for checking the user's permissions to the full decrypted values for the field.

The possible values are:

***NONE** No Authorization List is used by the field decryption APIs to check the user's permissions. However, the user will still need at least *USE authority to the Key Store which holds the Decryption Key.

authorization-list-name Indicate the name of the Authorization List. An Authorization List can be created with the IBM i command CRTAUTL. The users (or user groups) which need access to the decrypted values will need at least (*USE) authority to the Authorization List. Additionally, the users (or user groups) which need access to the decrypted values are required to have at least *USE authority to the Key Store object which holds the Decryption Key.

Auth. list for masked value (AUTLMASK)

Indicate the Authorization List that should be used by the field decryption APIs for checking the user's permissions to the masked values for the field.

The possible values are:

***NONE** No authorization list is used by the field decryption APIs to check the user's permissions. However, the user will still need at least *USE authority to the Key Store which holds the Decryption Key.

authorization-list-name Indicate the name of the Authorization List. An Authorization List can be created with the IBM i command CRTAUTL. The users (or user groups) which need access to the masked values will need at least (*USE) authority to the Authorization List. Additionally, the users (or user groups) which need access to the masked values are required to have at least *USE authority to the Key Store object which holds the Decryption Key.

Auth. list caching (AUTLCACHE)

Indicate if the permissions for authorization lists are 'cached' in memory.

The possible values are:

***YES** Caching will occur. When a field decrypt operation is performed, the permissions for the authorization lists will be saved (in memory) and used in future authority checks [for decrypt operations] within the job. This caching option provides the best performance.

> **NOTE:** In order to recognize any permission changes to the authorization lists, the jobs [that are performing decrypt operations] will need to be restarted.

***NO** Caching will not occur. The permissions to the authorization lists will be checked each time a decrypt operation is performed. This option is useful when you want changes to the authorization lists to be immediately recognized by jobs that are performing decrypt operations, or if you want to take advantage of program adopted authority when determining permissions to an authorization list.

Change Field Encryption Entry (CHGFLDENC)

The Change Field Encryption Entry (CHGFLDENC) command allows authorized users to change an existing entry in the Field Encryption Registry.

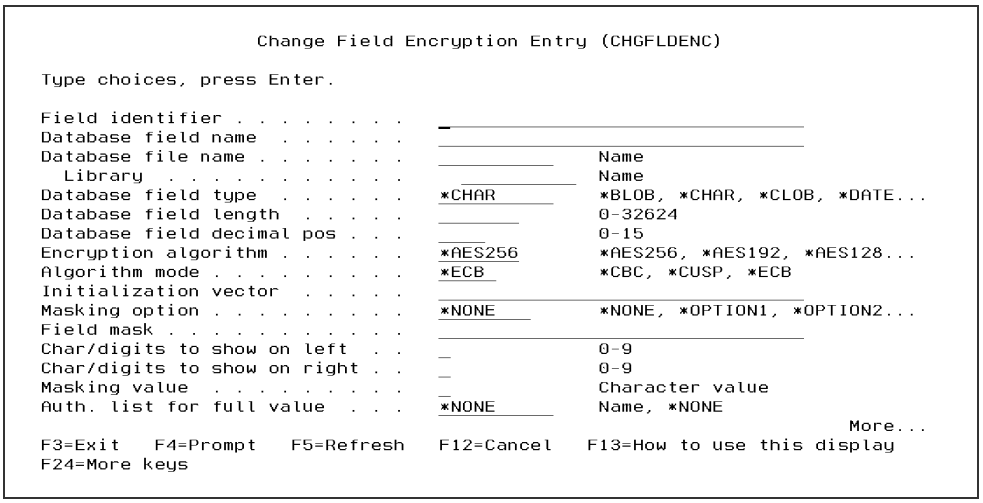
This command is only allowed for changing an entry with an *INACTIVE status.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires the user to have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

NOTE: The CHGFLDENC command only changes the field entry settings in the registry. It will not cause any action to be performed on the actual database field in the file. The field will not be activated for encryption until the ACTFLDENC (Activate Field Encryption) command is executed.



How to Get There

From the Field Encryption Menu, choose option **3**. Or, submit the command **CHGFLDENC**.

Options

Field identifier (FLDID)

Indicate the unique name of the entry up to 30 characters.

Rules for field identifier:

- The field identifier does not have to be the same name as the database field name to encrypt.
- The field identifier cannot contain spaces or certain special characters.
- The field identifier can contain underscore characters.
- The field identifier is not case sensitive. It will be stored in upper case.

Database field name (DBFLD)

Indicate the actual name of the database field to register for encryption. A field name up to 30 characters is supported.

The possible values are:

field-name The name of the field (or column name) in the database file to encrypt.
***REMOTE** The field is located in a remote database file (not local). Powertech Encryption will manage the encryption and storage of the field values through Tokenization. See Tokenization for Centralizing Sensitive Data for more information.

Database file name (DBFILE)

Indicate the name and library of the database file which contains the field to register.

Database field type (DBFLDTYP)

Indicate the data type of the database field.

The possible values are:

***BLOB** The field is a binary large object type.
***CHAR** The field is character type.
***CLOB** The field is a character large object type.
***DATE** The field is a date data type. This option is only valid when using a DB2 Field Procedure to automatically encrypt/decrypt the field values.
***DEC** The field is decimal type. This includes packed decimal and zoned decimal types.

If using a DB2 Field Procedure to automatically encrypt/decrypt the field values, then *DEC can also be used to indicate an Integer data type.

***TIME** The field is a time data type. This option is only valid when using a DB2 Field Procedure to automatically encrypt/decrypt the field values.
***TIMESTAMP** The field is a timestamp data type. This option is only valid when using a DB2 Field Procedure to automatically encrypt/decrypt the field values.

Database field length (DBFLDLEN)

Indicate the length of the values within the database field to encrypt.

For *CHAR type fields, the maximum allowable length is 32624.

For VARCHAR, the length parameter must be the length of the DB2 field plus 2 bytes (to accommodate the end-of-string delimiter).

For *DEC type fields, the maximum allowable length is 30.

If not using a DB2 Field Procedure and if you only want to encrypt a left portion of an alphanumeric field, then you can specify a length that is less than the actual field length.

> **NOTE:** If you specify a length that is less than the actual database field length, then any remaining bytes in the field will be cleared during the field activation process.

Database field decimal pos (DBFLDDEC)

For *DEC type database fields, indicate the number of decimal positions (scale) in the field.

Allowable range is 0 to 4.

Encryption algorithm (ALGORITHM)

Indicate the encryption algorithm to use for encrypting and decrypting the database field values.

This should match the algorithm used to create the Symmetric Key specified on the ENCKEYLBL parameter.

The possible values are:

***AES256** Use Advanced Encryption Standard (AES) algorithm and a 256 bit key.
***AES192** Use Advanced Encryption Standard (AES) algorithm and a 192 bit key.

***AES128** Use Advanced Encryption Standard (AES) algorithm and a 128 bit key.
***TDES** Use Triple Data Encryption Standard (TDES) algorithm.

Algorithm mode (MODE)

Indicate the mode to use within the encryption algorithm.

The possible values are:

***CBC** Mode is Cipher Block Chaining (CBC). This is a block-based mode. When using this mode with the AES algorithm, the length of the encrypted data will be a minimum of 16 bytes long. Its block-based length will be divisible by 16 or 24. When using this mode with the TDES algorithm, the length of the encrypted data will be a minimum of 8 bytes long. Its block-based length will be divisible by 8. With CBC mode, you can optionally specify an Initialization Vector (IV) to use as an additional input to the encryption algorithm.

***CUSP** Mode is Cryptographic Unit Support Program (CUSP). This is a stream-based mode, which means that the length of the encrypted data will equal the length of the input data. This mode is useful if the field data is not divisible by a block length of 16 or 24 (for AES), and if you want to store the encrypted values in your existing field (if not using DB2 Field Procedures). With CUSP mode, you can optionally specify an Initialization Vector (IV) to use as an additional input to the encryption algorithm.

***ECB** Mode is Electronic Code Book (ECB). This is a block-based mode. When using this mode with the AES algorithm, the length of the encrypted data will be a minimum of 16 bytes long. Its block-based length will be divisible by 16 or 24. block-based length will be divisible by 8. With ECB mode, you cannot specify an Initialization Vector. When using this mode with the TDES algorithm, the length of the encrypted data will be a minimum of 8 bytes long. Its block-based length will be divisible by 8. With ECB mode, you cannot specify an Initialization Vector.

Initialization vector (INITVECTOR)

Optional. The Initialization Vector (IV) is an arbitrary value that you can enter, which will be used as an additional input to the encryption algorithm. Therefore, the encrypted output is dependent on the combination of the Initialization Vector, Encryption Key and the Plain Text (the data you want to encrypt).

The Initialization Vector is allowed for *CBC and *CUSP algorithm modes.

The length of the Initialization Vector must not exceed the block length, which is calculated as:

- 32 bytes for *AES algorithm and *CUSP mode
- 16 or 24 bytes for *AES algorithm and *CBC mode
- 8 bytes for *TDES algorithm and *CBC mode

Mask option (MASKOPT)

Indicate the mask option to use for the field when the masked value is requested on a decrypt operation.

The possible values are:

***NONE** No masking is performed.
***OPTION1** Exact positions within the field value can be shown or masked using the FLDMASK parameter.
***OPTION2** Only a specified number of digits or characters are shown on the left and right sides of the field using the DIGLEFT and DIGRIGHT parameters. Specify the masking character to use in the FLDMASKV parameter. When using field procedures, you must specify a masking character of 1-9, as only numbers are allowed in numeric fields. Examples when DIGLEFT(4) and DIGRIGHT(4) is specified with FLDMASKV(#) when using field procedures:

- String ' 1234567890123456' is masked as ' 1234999999993456'
- String '1234567890123456 ' is masked as '1234999999993456 '
- Numeric 001234567890123456 is masked as 001234999999993456
- Numeric 1234567890123456 is masked as 1234999999993456

If you are not using field procedures, a non-numeric character can be used, for example:

- String ' 1234567890123456' is masked as ' 1234########3456'
- String '1234567890123456 ' is masked as '1234########3456 '
- Numeric 001234567890123456 is masked as 001234########3456
- Numeric 1234567890123456 is masked as 1234########3456

***OPTION3** Used for Date, Time and Timestamp fields. The value entered will be used as the mask value. The mask value must be a proper Date, Time or Timestamp value for the field being masked. It is recommended to specify a mask value that does not conflict with an existing value in your database. For instance, for a date field, you may want to specify a mask value with a high value date of 9999-12-31. The format of the mask value must be in the format of the field format. For example if the "Date Format" of a field is *ISO, then the format of the masked value must be *ISO.

Examples:

- Date '9999-12-31'
- Time '24.00.00'
- TimeStamp '9999-12-31-24.00.00.000000'

NOTE: When specifying *OPTION3, you should not specify a not-authorized fill value.

Field mask (FLDMASK)

Indicate the masking format to apply to the field when the masked value is requested on a decrypt operation. Valid for MASKOPT(*OPTION1) and MASKOPT(*OPTION3) masking.

- For MASKOPT(*OPTION1) masking

Specify the number 9 in a position to show the underlying value for that position. Specify any other character (including spaces) or number in a position to mask the underlying value for that position.

For example, if a mask of '**********9999' is specified for a credit card number, then a sample of a masked credit card number would be '**********1234'.

As another example, if a mask of '##99##999' is specified for an account number, then a sample of a masked account number would be '##76##541'.

When the field type is numeric the whole number is masked. The decimals values are not. The Mask must not be longer than the whole number length. When using field procedures the mask value must be numeric. For example, if a mask of '779977999' is specified for an account number, then a sample of a masked account number would be '774577541'.

- For MASKOPT(*OPTION3) masking

You must enter in a valid Date, Time or Timestamp value. This will be the value used as the mask.

Examples:

- Date '9999-12-31'
- Time '24.00.00'
- TimeStamp '9999-12-31-24.00.00.000000'

Used for Date, Time and Timestamp fields. The value entered will be used as the mask value. The mask value must be a proper Date, Time or Timestamp value for the field being masked.

It is recommended to specify a mask value that does not conflict with an existing value in your database. For instance, for a date field, you may want to specify a mask value with a high value date of 9999-12-31.

The format of the mask value must be in the format of the field format. For example if the "Date Format" of a field is *ISO, then the format of the masked value must be *ISO.

Examples:

- Date '9999-12-31'
- Time '24.00.00'
- TimeStamp '9999-12-31-24.00.00.000000'

> **NOTE:** When specifying *OPTION3, you should not specify a not-authorized fill value.

Char/Digits to show on left (DIGLEFT)

When a masked value is requested on a decrypt operation, indicate the number of characters or digits to show on the left side of the field value. Valid for MASKOPT(*OPTION2) masking.

For a character field, any leading blank characters will be ignored when performing the masking. For a decimal field, any leading zeros will be ignored.

Char/Digits to show on right (DIGRIGHT)

When a masked value is requested on a decrypt operation, indicate the number of characters or digits to show on the right side of the field value. Valid for MASKOPT(*OPTION2) masking.

For a character field, any trailing blank characters will be ignored when performing the masking.

Masking Value (FLDMASKV)

The value to be used as the masking character or number. Valid for MASKOPT(*OPTION2) masking.

When masking a numeric field and using DB2 Field Procedures, the mask value must be a number between 1 and 9.

When masking a character field, or when masking a numeric field and not using DB2 Field Procedures, the mask value can be any character or number.

Auth. list for full value (AUTLDEC)

Indicate the IBM i Authorization List that should be used to determine which users have authority to the full field values on decrypt operations.

This Authorization List will be used in field decryption APIs and DB2 Field Procedures.

The possible values are:

authorization-list-name Indicate the name of the Authorization List. An Authorization List can be created with the IBM i command CRTAUTL. The users (or user groups) which need access to the decrypted values will need at least (*USE) authority to the Authorization List. Additionally, the users (or user groups) which need access to the decrypted values are required to have at least *USE authority to the Key Store object which holds the Decryption Key.

***NONE** An Authorization List should not be used by the decrypt operations. Therefore the user can gain access to the fully decrypted values as long as they have at least *USE authority to the Key Store which holds the Decryption Key.

Auth. list for masked value (AUTLMASK)

Indicate the IBM i Authorization List that should be used to determine which users have authority to the masked field values on decrypt operations.

This Authorization List will be used in field decryption APIs and DB2 Field Procedures.

The possible values are:

authorization-list-name Indicate the name of the Authorization List. An Authorization List can be created with the IBM i command CRTAUTL. The users (or user groups) which need access to the masked values will need at least (*USE) authority to the Authorization List. Additionally, the users (or user groups) which need access to the masked values are required to have at least *USE authority to the Key Store object which holds the Decryption Key.

***NONE** An Authorization List should not be used by the decrypt operations. Therefore the user can gain access to the masked values as long as they have at least *USE authority to the Key Store which holds the Decryption Key.

Auth. list caching (AUTLCACHE)

Indicate if the permissions for authorization lists are 'cached' in memory.

The possible values are:

***YES** Caching will occur. When a field decrypt operation is performed, the permissions for the authorization lists will be saved (in memory) and used in future authority checks [for decrypt operations] within the job. This caching option provides the best performance.

> **NOTE:** To recognize any permission changes to the authorization lists, the jobs [that are performing decrypt operations] will need to be restarted.

***NO** Caching will not occur. The permissions to the authorization lists will be checked each time a decrypt operation is performed. This option is useful when you want changes to the authorization lists to be immediately recognized by jobs that are performing decrypt operations, or if you want to take advantage of program adopted authority when determining permissions to an authorization list.

Not authorized fill value (NOTAUTHFV)

Indicate the 1-byte value to fill the returned value on a decryption request (from a DB2 Field Procedure or a Powertech Encryption 'auth' API) if the user is not authorized to either the full or masked authorization lists.

For instance, if the fill value is '9' and the field length is 7, then the value of '9999999' will be returned on an unauthorized decryption request.

> **NOTE:**
>
> - The fill value is required when a DB2 Field Procedure is utilized and the return value (FLDPROCOPT) is set to *AUTH.
> - If the field type is *CHAR, then the fill value can be a number, letter or special character (e.g. #, *, %).
> - If the field type is *DEC, then the fill value can be a number from 1 through 9 if a DB2 Field Procedure is being utilized, otherwise it can be number from 0 through 9.
> - The fill value is not allowed for field types of *DATE, *TIME and *TIMESTAMP.

Store values in external file (STREXTFILE)

Indicate if the encrypted field values should be stored in a separate external file.

The encrypted values must be stored in an external file if not using a DB2 Field Procedure and if any of the following conditions are met:

- If the field type is *DEC.
- For *AES128, *AES192 and *AES256 algorithms with *CBC or *ECB modes: If the field type is *CHAR and the length specified for DBFLDLEN is not divisible by 16 or 24.
- For *TDES algorithm: If the field type is *CHAR and the length specified for DBFLDLEN is not divisible by 8.

The possible values are:

***YES** Store the encrypted field values in an external file.
***NO** Store the encrypted values in the existing database field.

External file name (EXTFILE)

Indicate the name and library of the external physical file which will be created to contain the encrypted field values.

The possible values are:

external-file-name Indicate the name of the external physical file object to store the encrypted values.

This object name must NOT already exist.

***GEN** Powertech Encryption will automatically generate the external physical file object name, which uses the naming convention of CRXXnnnnn (where nnnnn is a sequential number from 1 to 99999).

The possible library values are:

library-name Indicate the name of the library to create the external physical file in.
***DBLIB** Specify *DBLIB for the library name to indicate that the external physical file should be created in the same library as where the database file (which contains the field to encrypt) resides.

The external physical file will be keyed by the Field Identifier (XXFLDID) and the Index Number (XXINDEX).

External logical file name (EXTFILEL)

Indicate the name and library of the logical file which will be built over the external physical file, which will be keyed by the Field Identifier (XXFLDID) and the Encrypted Value (XXVALUE).

The possible values are:

***NONE** The logical file will not be created.
external-logical-file-name Indicate the name of the logical file object over the external physical file.

This object name must NOT already exist.

***GEN** Powertech Encryption will automatically generate the logical file object name, which uses the naming convention of CRXXnnnnnL (where nnnnn is a sequential number from 1 to 99999).

The possible library values are:

library-name Indicate the name of the library to create the external logical file in.
***DBLIB** Specify *DBLIB for the library name to indicate that the external logical file should be created in the same library as where the database file (which contains the field to encrypt) resides.

Store hash for security check (EXTSTRHASH)

Indicate if a HASH value should be stored for each record in the external file.

The possible values are:

***YES** A HASH value will be stored for each record in the external file. The HASH value is an encrypted value which is calculated based on the Field identifier, Index number and Key id for each record. When Powertech Encryption retrieves a record from the external file, it will recalculate the HASH and compare it to the stored HASH. A mismatch in the HASH values will indicate that an unauthorized change was made to the external file record's Field identifier, Index number and/or Key id.

***NO** A HASH value will not be stored in the external file.

Store last retrieved user/time (EXTSTRRTV)

Indicate if the user id and timestamp of when a field's value was last retrieved(decrypted) should be stored for each record in the external file.

> **NOTE:** You can additionally turn on audit logging for a Decryption key to log each time the key is used to decrypt data.

The possible values are:

***YES** Stores the last retrieved user/time in the external file.
***NO** Does not store the last retrieved user/time in the external file.

Align index number (INDEXALIGN)

When encrypting a character (alphanumeric) type field which is stored in an external file, then indicate how the external index number should be aligned in the field.

The possible values are:

***LEFT** The index number should be aligned on the left side of the database field.
***RIGHT** The index number should be aligned on the right side of the database field.

Index padding character (INDEXPAD)

When encrypting a character (alphanumeric) type field, indicate a padding character to place in the unused positions of the field.

For instance, if a padding character of "*" is specified with an alignment of *LEFT, then a 10 position field value with an index of 895 would appear as "895*******".

The padding character cannot be a number, a single quote (') or a dash (-).

Last index number storage (LSTINDSTG)

Indicate the object type to store the 'last index number used'.

Each time a record is written (inserted) to the external file, the 'last index number used' is retrieved from the object, increased by 1, assigned to the new record and saved back to the object.

The possible values are:

***FLDREG** Store the last index number used in the field registry object, which is a validation list (*VLDL) with the name of CRVL002.
***PF** Store the last index number used in a physical file with the name of CRPF002. A physical file may be easier to replicate (than a *VLDL) with a High Availability tool. A physical file will also provide better performance (than a *VLDL) when a high volume of inserts occurs for the field, due to the file's ability to handle locks more efficiently.

> **NOTE:** If the CRPF002 physical file does not exist, it can be created using the CRTPF command. The source is located in the CRPF002 member in the source file of CRYPTO/QDDSSRC.

Use triggers to auto encrypt (USETRG)

Indicate if SQL triggers should be created over the database file, which will automatically encrypt the database field values without having to change your applications.

> **NOTE:** SQL triggers are not allowed for encryption if a DB2 Field Procedure is already specified with USEFLDPROC(*YES).

They are mutually exclusive.

The possible values are:

***YES** SQL triggers will be created over the database file.
***NO** SQL triggers will not be created. You will need to use a DB2 Field Procedure or call Powertech Encryption encryption APIs from within your application programs to encrypt the field values.

Trigger name for inserts (INSTRG)

Indicate the name and library of the trigger to create. This trigger will be used to automatically encrypt the field value when records are inserted (added) into the database.

The possible values are:

trigger-name Specify the name of the trigger to create. Rules for trigger names:

- The trigger name cannot exceed 80 characters in length.
- The trigger name cannot be the same name as a trigger that is already on the database file.
- The trigger name cannot contain spaces or certain special characters.
- The trigger name can contain underscore characters.
- The trigger name is not case sensitive. It will be stored in upper case.

***GEN** Specify *GEN to automatically generate the trigger name, which uses the naming convention of "FILENAME_FIELDNAME_CryptoInsert".

The possible library values are:

library-name Indicate the name of the library to create the trigger object in.
***DBLIB** The trigger will be created in in the same library as where the database file resides.

Trigger name for updates (UPDTRG)

Indicate the name and library of the trigger to create. This trigger will be used to automatically encrypt the field value when records are updated in the database.

This is a column trigger, so it is only called when the particular field value is changed.

The possible values are:

trigger-name Specify the name of the trigger to create. Rules for trigger names:

- The trigger name cannot exceed 80 characters in length.
- The trigger name cannot be the same name as a trigger that is already on the database file.
- The trigger name cannot contain spaces or certain special characters.
- The trigger name can contain underscore characters.
- The trigger name is not case sensitive. It will be stored in upper case.

***GEN** Specify *GEN to automatically generate the trigger name, which uses the naming convention of "FILENAME_FIELDNAME_CryptoUpdate".

The possible library values are:

library-name Indicate the name of the library to create the trigger object in.
***DBLIB** The trigger will be created in in the same library as where the database file resides.

Trigger name for deletes (DLTTRG)

This is only valid if an external file is used to store the encrypted values: Indicate the name and library of the trigger to create. This trigger will be used to automatically remove the encrypted field value (record) from the external file when a database record is deleted.

The possible values are:

trigger-name Specify the name of the trigger to create. Rules for trigger names:

- The trigger name cannot exceed 80 characters in length.
- The trigger name cannot be the same name as a trigger that is already on the database file.
- The trigger name cannot contain spaces or certain special characters.
- The trigger name can contain underscore characters.
- The trigger name is not case sensitive. It will be stored in upper case.

***GEN** Specify *GEN to automatically generate the trigger name, which uses the naming convention of "FILENAME_FIELDNAME_CryptoDelete".

The possible library values are:

library-name Indicate the name of the library to create the trigger object in.
***DBLIB** The trigger will be created in in the same library as where the database file resides.

Trigger exit type (TRGEXITTYP)

Indicate if the triggers should call a custom exit program before performing any inserts, updates or deletes of the field value.

Listed below are some examples of how a Trigger exit program could be utilized:

- To write out additional audit entries in the audit journal file.
- To direct Powertech Encryption to not process (ignore) the requested insert/update/delete of the field value based on custom criteria, such as the user id or application performing the request.
- To perform additional custom logic.

A trigger exit program can be written in RPG, COBOL or C on the System i. Source examples of RPG trigger programs are provided in the members of TRGEXTPGM and TRGEXTSRV within the source file of CRYPTO/QRPGLESRC.

The possible values are:

***NONE** No Trigger exit program is used.
***PGM** The Trigger exit is a Program (*PGM) object.
***SRVPGM** The Trigger exit is a Service Program (*SRVPGM) object.

Trigger exit program (TRGEXITPGM)

Indicate the name and library of the trigger exit program to call.

program-name Enter the name of the program or service program.

The possible library values are:

library-name Enter the name of the library where the program is located.
***LIBL** Locate the program within the library list.

Trigger exit procedure (TRGEXITPRC)

Indicate the name of the procedure to call if the TRGEXITTYP is *SRVPGM.

Use DB2 field procedure (USEFLDPROC)

Indicate if a DB2 Field Procedure will be used to automatically encrypt/decrypt the field values, which is an alternative approach to using triggers and API calls. A DB2 Field Procedure also allows storing the 'encoded' encrypted values within the existing file, which is especially useful for numeric fields. [You will not need to create a separate external file to store the values for numeric fields.]

DB2 Field Procedures are available in IBM i version V7R1 and higher.

> **NOTE:** DB2 Field Procedures are not allowed if SQL triggers are already specified with USETRG(*YES). They are mutually exclusive.

Before using DB2 Field Procedures in a production environment, read the manual to understand the potential performance issues and risks.

The possible values are:

***YES** A DB2 Field Procedure will be used to automatically encrypt and decrypt the field values.
***NO** A DB2 Field Procedure will not be used. You must use triggers or APIs to encrypt the values. APIs must be used to decrypt the values.

Field procedure return value (FLDPROCOPT)

Indicate which field value is returned (based on user permissions) from the DB2 Field Procedure to the application on a read operation.

The possible values are:

***FULL** Returns the fully decrypted value if the user has at least *USE rights to the authorization list specified on the AUTLDEC parameter (or if *NONE is specified on that parameter). Otherwise an error with the message id of CPF504D will be generated in the application performing the read. The *FULL option is available for *CHAR, *DEC, *DATE, *TIME and *TIMESTAMP data types.

***AUTH** For character (*CHAR) fields up to 30 bytes in length, this option returns either: 1) the full value if the user has at least *USE rights to the authorization list specified on the AUTLDEC parameter (or if *NONE is specified on that parameter) or 2) the masked value if the user has at least *USE rights to the authorization list specified on the AUTLMASK parameter (or if *NONE is specified on that parameter) or 3) the fill value if the user does not have at least *USE rights to either authorization list. For decimal/numeric (*DEC) fields or character fields over 30 bytes in length, this option returns either: 1) the full value if the user has at least *USE rights to the Authorization List specified on the AUTLDEC parameter (or if *NONE is specified on that parameter) or 2) the numeric masking character and the left/right digit (to show) logic (MASKOPT *OPTION2) if the user has at least *USE rights to the authorization list specified on the AUTLMASK parameter (or if *NONE is specified on that parameter). The fill value is defined in the Change Field Encryption Entry panel (CHGFLDENC). Otherwise (if not authorized), the fill value is returned.

> **NOTE:** A zero value in a numeric field is displayed in clear text and is not masked with the Mask character or Not Authorized character.

The *AUTH option is valid for *DATE, *TIME and *TIMESTAMP data types only when the MASKOPT parameter is either *NONE or *OPTION3.

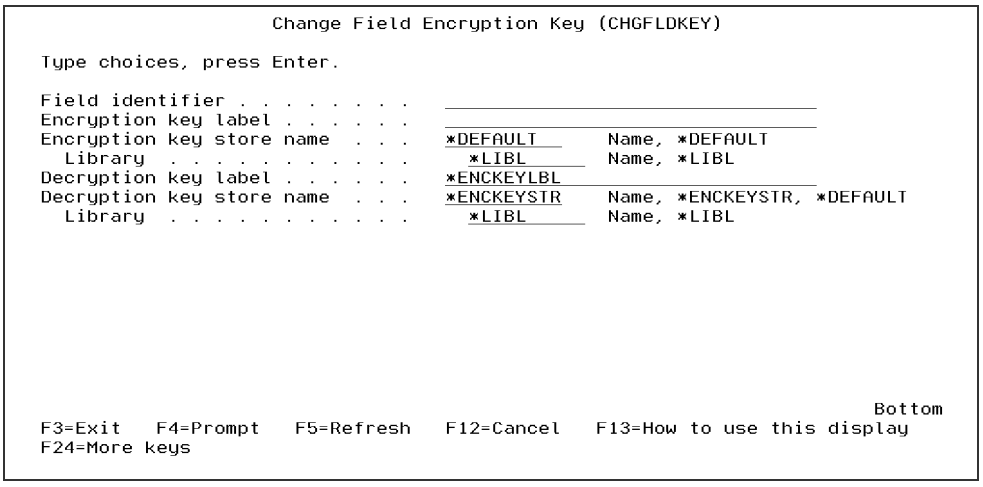
Decryption Accelerator (PERFACCEL)

Allows Powertech Encryption for IBM i to apply methods to attempt to improve performance gains when possible with native i/o.

The possible values are:

***YES** See Appendix G: Decryption Accelerator Prerequisites and Limitations for considerations and prerequisites. Apply methods to attempt to improve performance gains when possible.
***NO** Do not apply methods to attempt to improve performance gains when possible.

Change Field Encryption Key (CHGFLDKEY)



The CHGFLDKEY command allows authorized users to change (rotate) the keys used to encrypt and decrypt data for a field entry in the Encryption Registry. Up to 99,999 keys can be rotated for a field entry.

This command can be used for *INACTIVE field entries, as well as *ACTIVE field entries that store the encrypted field values in an external file.

The following users can use the CHGFLDKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain Field Enc. Registry” authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object, which contains the Field Encryption Registry.

Do the following steps to change the keys for a field entry in the Encryption Registry:

1. Prompt (**F4**) the command **CRYPTO/CHGFLDKEY**.
2. Press **F1** on any parameter for complete online help text.
3. Press Enter after the parameter values are specified.

How to Get There

From the Field Keys Menu, choose option **2**, Change Field Encryption Key. Or, in the Work with Field Encryption Registry (WRKFLDENC) panel, choose option **10** for an item. Or, execute the command **CHGFLDKEY**.

Options

Field identifier (FLDID)

Indicate the unique name of the field entry to change the keys for.

Encryption key label (ENCKEYLBL)

Indicate the label of the Symmetric Key to use for encrypting the field values.

Encryption key store name (ENCKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for encryption of the field.

The users (or user groups) which need to encrypt values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Decryption key label (DECKEYLBL)

Indicate the label of the Symmetric Key to use for decrypting the field values.

The possible values are:

decryption-key-label Indicate the label of the key to use for decryption.

WARNING: If specifying a different key label than the label specified for encryption, then that decryption key should contain the same key value as the encryption key.

***ENCKEYLBL** Use the same label as specified on the ENCKEYLBL parameter.

Decryption key store name (DECKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the field.

The users (or user groups) that need access to the decrypted values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***ENCKEYSTR** Use the same Key Store as specified on the ENCKEYSTR parameter.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Change Field Mask (CHGFLDMSK)

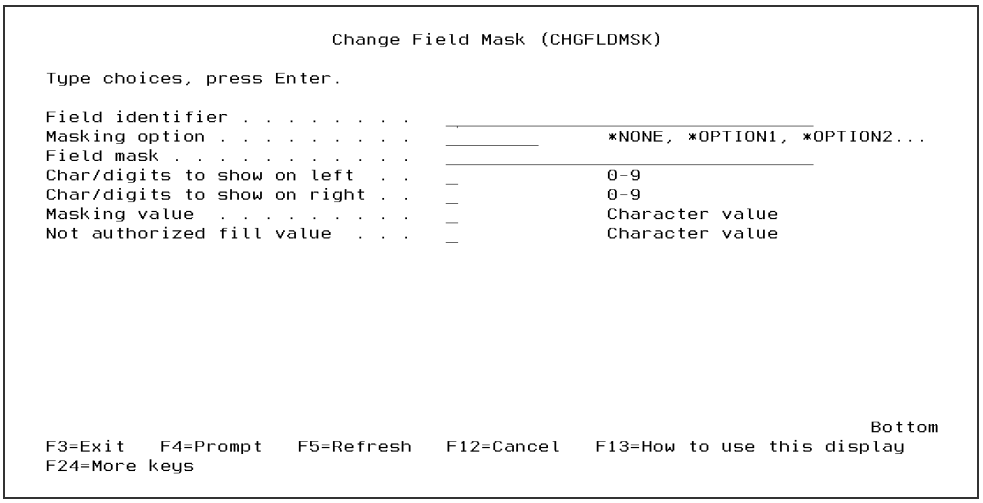
The Change Field Mask (CHGFLDMSK) command allows authorized users to change the mask for a field in the Field Encryption Registry.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires the user to have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

> **NOTE:** The CHGFLDMSK command only changes the field entry settings in the registry. It will not cause any action to be performed on the actual database field in the file.



How to Get There

From the Field Encryption Menu, choose option **4**. Or, submit the command **CHGFLDMSK**.

Options

Field identifier (FLDID)

Indicate the Field identifier to change the mask for.

Mask option (MASKOPT)

Indicate the mask option to use for the field when the masked value is requested on a decrypt operation.

The possible values are:

***NONE** No masking is performed.
***OPTION1** Exact positions within the field value can be shown or masked using the FLDMASK parameter.
***OPTION2** Only a specified number of digits or characters are shown on the left and right sides of the field using the DIGLEFT and DIGRIGHT parameters. Specify the masking character to use in the FLDMASKV parameter. When using field procedures, you must specify a masking character of 1-9, as only numbers are allowed in numeric fields. Examples when DIGLEFT(4) and DIGRIGHT(4) is specified with FLDMASKV(#) when using field procedures:

- String ' 1234567890123456' is masked as ' 1234999999993456'
- String '1234567890123456 ' is masked as '1234999999993456 '
- Numeric 001234567890123456 is masked as 001234999999993456
- Numeric 1234567890123456 is masked as 1234999999993456

If you are not using field procedures, a non-numeric character can be used, for example:

- String ' 1234567890123456' is masked as ' 1234########3456'
- String '1234567890123456 ' is masked as '1234########3456 '
- Numeric 001234567890123456 is masked as 001234########3456
- Numeric 1234567890123456 is masked as 1234########3456

***OPTION3** Used for Date, Time and Timestamp fields. The value entered will be used as the mask value. The mask value must be a proper Date, Time or Timestamp value for the field being masked. It is recommended to specify a mask value that does not conflict with an existing value in your database. For instance, for a date field, you may want to specify a mask value with a high value date of 9999-12-31.

Examples:

- Date '9999-12-31'
- Time '24.00.00'
- TimeStamp '9999-12-31-24.00.00.000000'

NOTE: When specifying *OPTION3, you should not specify a not-authorized fill value.

Field mask (FLDMASK)

Indicate the masking format to apply to the field when the masked value is requested on a decrypt operation. Valid for MASKOPT(*OPTION1) masking.

Specify the number 9 in a position to show the underlying value for that position. Specify any other character (including spaces) or number in a position to mask the underlying value for that position.

For example, if a mask of '**********9999' is specified for a credit card number, then a sample of a masked credit card number would be '**********1234'.

As another example, if a mask of '##99##999' is specified for an account number, then a sample of a masked account number would be '##76##541'.

When the field type is numeric the whole number is masked. The decimals values are not. The Mask must not be longer than the whole number length. When using field procedures the

mask value must be numeric. For example, if a mask of '779977999' is specified for an account number, then a sample of a masked account number would be '774577541'.

- For MASKOPT(*OPTION3) masking

You must enter in a valid Date, Time or Timestamp value. This will be the value used as the mask.

Examples:

- Date '9999-12-31'
- Time '24.00.00'
- TimeStamp '9999-12-31-24.00.00.000000'

Used for Date, Time and Timestamp fields. The value entered will be used as the mask value. The mask value must be a proper Date, Time or Timestamp value for the field being masked.

It is recommended to specify a mask value that does not conflict with an existing value in your database. For instance, for a date field, you may want to specify a mask value with a high value date of 9999-12-31.

The format of the mask value must be in the format of the field format. For example if the "Date Format" of a field is *ISO, then the format of the masked value must be *ISO.

Examples:

- Date '9999-12-31'
- Time '24.00.00'
- TimeStamp '9999-12-31-24.00.00.000000'

> **NOTE:**
> - When specifying *OPTION3, you should not specify a not-authorized fill value.
> - Char/Digits to show on left (DIGLEFT)
> - When a masked value is requested on a decrypt operation, indicate the number of characters or digits to show on the left side of the field value. Valid for MASKOPT (*OPTION2) masking.

For a character field, any leading blank characters will be ignored when performing the masking. For a decimal field, any leading zeros will be ignored.

Char/Digits to show on right (DIGRIGHT)

When a masked value is requested on a decrypt operation, indicate the number of characters or digits to show on the right side of the field value. Valid for MASKOPT(*OPTION2) masking.

For a character field, any trailing blank characters will be ignored when performing the masking.

Masking Value (FLDMASKV)

The value to be used as the masking character or number. Valid for MASKOPT(*OPTION2) masking.

When masking a numeric field and using DB2 Field Procedures, the mask value must be a number between 0 and 9.

When masking a character field, or when masking a numeric field and not using DB2 Field Procedures, the mask value can be any character or number.

Not authorized fill value (NOTAUTHFV)

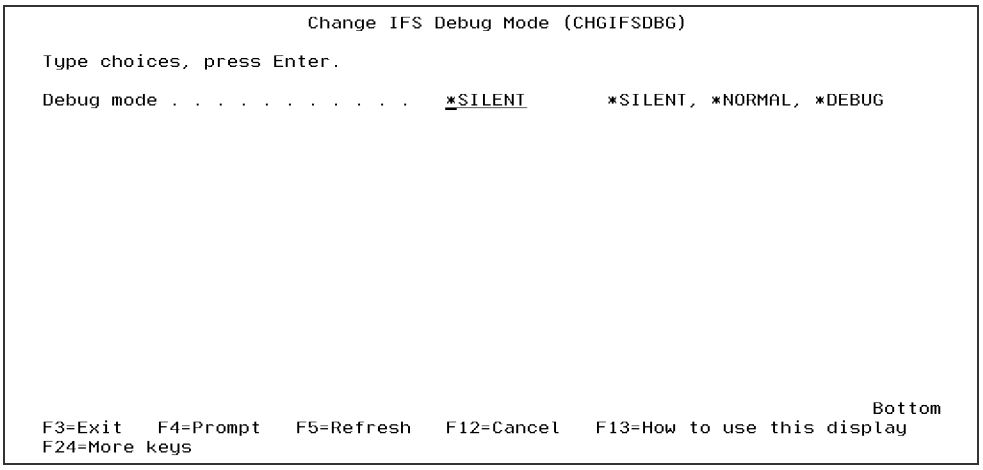
Indicate the 1-byte value to fill the returned value on a decryption request (from a DB2 Field Procedure or a Powertech Encryption 'auth' API) if the user is not authorized to either the full or masked authorization lists.

For instance, if the fill value is '9' and the field length is 7, then the value of '9999999' will be returned on an unauthorized decryption request.

NOTE:

- The fill value is required when a DB2 Field Procedure is utilized and the return value (FLDPROCOPT) is set to *AUTH.
- If the field type is *CHAR, then the fill value can be a number, letter or special character (e.g. #, *, %).
- If the field type is *DEC, then the fill value can be a number from 1 through 9 if a DB2 Field Procedure is being utilized, otherwise it can be number from 0 through 9.
- The fill value is not allowed for field types of *DATE, *TIME and *TIMESTAMP.

Change IFS Debug Mode (CHGIFSDBG)



The Change IFS Debug mode (CHGIFSDBG) command allows authorized users to change the IFS encryption debug mode.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRPFIFS2 file and CRSRVRUN data area, both of which are located in the CRYPTO library.

How to Get There

From the IFS Utility Menu, choose option **6**, Change IFS Debug Mode. Or, prompt (**F4**) the command **CRYPTO/CHGIFSDBG**.

Options

Debug Mode (DEBUG)

Indicate the Debug mode to use.

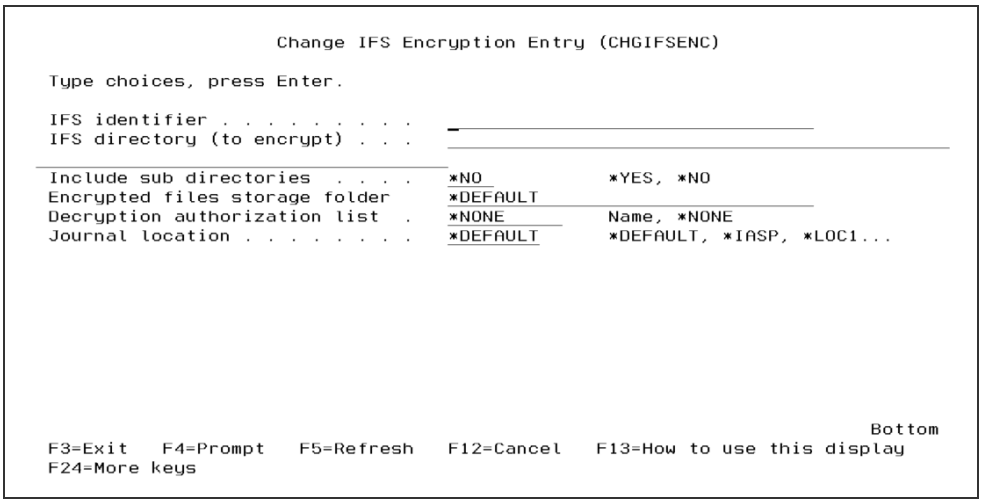
The possible values are:

***SILENT** Only error messages will be written to the CRPFIFSLOG File. This is the default setting.
***NORMAL** Encryption and Decryption activities, as well as any error messages, will be written to the CRPFIFSLOG File.

***DEBUG** Encryption and Decryption activities, error messages and program debug messages will be written to the CRPFIFSLOG File.

Change IFS Encryption Entry (CHGIFSENC)

The Change IFS Encryption Entry (CHGIFSENC) command allows authorized users to change an entry in the IFS Encryption Registry.



This command is only allowed for changing an entry with an *INACTIVE status.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

The CHGIFSENC command allows authorized users to change the settings for an *INACTIVE IFS entry in the Encryption Registry. This command requires the user to have *CHANGE authority to the CRVL003 Validation List (*VLDL) object which contains the IFS Encryption Registry.

> **NOTE:** The CHGIFSENC command only changes the settings in the IFS registry. It will not cause any action to be performed on the actual directory or files in the directory. The IFS entry will not be activated for encryption until the ACTIFSENC (Activate IFS Encryption) command is executed.

How to Get There

On the Work with IFS Encryption Registry (WRKIFSENC) panel, choose option **2** for an IFS identifier.

Options

IFS identifier (IFSID)

Indicate the unique name of the entry up to 30 characters.

Rules for IFS identifier:

- The IFS identifier does not have to be the same name as the directory or files to encrypt. It is simply used as a way to identify this entry within IFS registry.
- The IFS identifier cannot contain spaces or certain special characters.
- The IFS identifier can contain underscore characters.
- The IFS identifier is not case sensitive. It will be stored in upper case.

IFS directory to encrypt (SRCDIR)

The maximum size of the directory name is 256 bytes. The maximum size of any filename is 256 bytes. Specify the path of the IFS directory containing the files to be encrypted. For instance: '/HR/PayrollData'

Include subdirectories (INCSUBDIR)

Indicate if the files within the directory's subdirectories are to be encrypted.

The possible values are:

***YES** Files within the subdirectories will also be encrypted.
***NO** Files within the subdirectories will NOT be encrypted.

Encrypted files storage folder (SRCDIR)

The maximum size of the directory name is 256 bytes. Specify the path of the IFS directory to store the encrypted versions of the files. If this directory does not exist, then it will be created. If this is an existing directory, then it cannot contain existing files.

The possible values are:

***DEFAULT** The encrypted versions of the files will be stored under the '/CryptoDisk' directory using the same directory name as specified for the SRCDIR parameter. For instance: '/CryptoDisk/HR/PayrollData'

ifs-directory-name Specify the full path to the IFS directory name to store the encrypted versions of the IFS files. For instance: '/Encrypted/HR/PayrollData'

Authorization list for decryption (AUTLDEC)

Indicate the IBM i Authorization List that should be used to determine which users have authority to decrypt the IFS files.

The possible values are:

authorization-list-name Indicate the name of the Authorization List. An Authorization List can be created with the IBM i command CRTAUTL. The users (or user groups) which need access to the decrypted IFS files will need at least (*USE) authority to the Authorization List.

***NONE** An Authorization List should not be used by the IFS decrypt operations. Therefore the user can gain access to the decrypted files as long as they have object authority to the IFS file and at least *USE authority to the Key Store which holds the Decryption Key.

Journal location (JRNLOC)

Indicate the location of the journal and related objects.

The possible values are:

***DEFAULT** The location for all related objects will be in the CRYPTO library. Also the name of the journal will be CRJNI001. No further changes will need to be made.

***IASP** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the IASP library designated.

The following objects will need to be copied into the IASP library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRPFIFS2 PHYSICAL FILE
- CRVL003 VALIDATION LIST
- CRJNI001 JOURNAL CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_IASP_CRPFIFS_LIBRARY
- IFS_IASP_CRPFIFS2_LIBRARY
- IFS_IASP_REGISTRY_LIBRARY
- IFS_IASP_JOURNAL_LIBRARY
- IFS_IASP_LAST_SEQ_DTAARA_LIBRARY
- IFS_IASP_SERVER_RUN_DTAARA_LIBRARY

***LOC1** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC1 library designated. The IFS Encryption Registry(CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC1 library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC1_CRPFIFS_LIBRARY
- IFS_LOC1_REGISTRY_LIBRARY
- IFS_LOC1_JOURNAL_LIBRARY
- IFS_LOC1_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC1_SERVER_RUN_DTAARA_LIBRARY

***LOC2** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC2 library designated. The IFS Encryption Registry(CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC2 library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC2_CRPFIFS_LIBRARY
- IFS_LOC2_REGISTRY_LIBRARY
- IFS_LOC2_JOURNAL_LIBRARY
- IFS_LOC2_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC2_SERVER_RUN_DTAARA_LIBRARY

***LOC3** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC3 library designated. The IFS Encryption Registry(CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC3 library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC3_CRPFIFS_LIBRARY
- IFS_LOC3_REGISTRY_LIBRARY
- IFS_LOC3_JOURNAL_LIBRARY
- IFS_LOC3_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC3_SERVER_RUN_DTAARA_LIBRARY

***LOC4** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC4 library designated. The IFS Encryption Registry(CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC4 library:

- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC4_CRPFIFS_LIBRARY
- IFS_LOC4_REGISTRY_LIBRARY
- IFS_LOC4_JOURNAL_LIBRARY
- IFS_LOC4_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC4_SERVER_RUN_DTAARA_LIBRARY

***LOC5** The location of the objects will need to be entered into the CRCONFIG file located in the CRYPTO library and the objects will need to be copied into the LOC5 library designated. The IFS Encryption Registry(CRVL003) will need to be in the CRYPTO library.

The following objects will need to be copied into the LOC5 library:

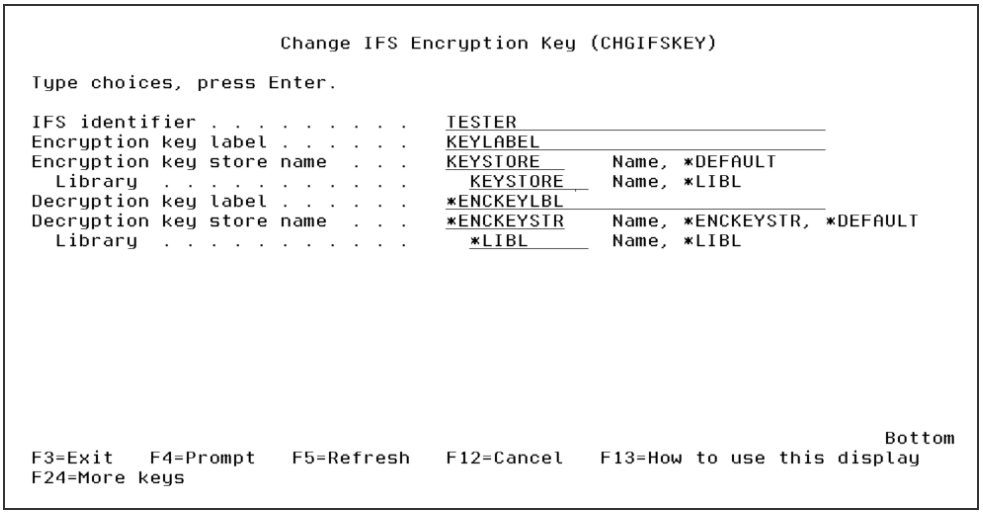
- CRPFIFS PHYSICAL FILE
- CRPFIFSL1 LOGICAL FILE
- CRPFIFSL2 LOGICAL FILE
- CRPFIFSL3 LOGICAL FILE
- CRPFIFSL4 LOGICAL FILE
- CRJNI001 JOURNAL
- CRJRI001 JOURNAL RECEIVER
- CRLSTSEQ DATA AREA
- CRVERSION DATA AREA

The following entries will need to be added into the CRCONFIG file:

- IFS_LOC5_CRPFIFS_LIBRARY
- IFS_LOC5_REGISTRY_LIBRARY
- IFS_LOC5_JOURNAL_LIBRARY
- IFS_LOC5_LAST_SEQ_DTAARA_LIBRARY
- IFS_LOC5_SERVER_RUN_DTAARA_LIBRARY

Change IFS Encryption Key (CHGIFSKEY)

The Change IFS Encryption Key (CHGIFSKEY) command allows authorized users to change (rotate) the keys used for an entry in the IFS Encryption Registry.



Up to 99,999 keys can be rotated for a registry entry.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL003 Validation List (*VLDL) object which contains the IFS Encryption Registry.

> **NOTE:** The new key will only be used for future encryption processes of IFS files. Existing encrypted files will not be affected. In other words, this command will not re-encrypt existing encrypted IFS files with the new key.

How to Get There

On the Work with IFS Encryption Registry (WRKIFSENC) panel, choose option **10** for an IFS identifier.

Options

IFS identifier (IFSID)

Indicate the unique name of the IFS registry entry to change the keys for.

Encryption key label (ENCKEYLBL)

Indicate the label of the Symmetric Key to use for encrypting the IFS files.

Encryption key store name (ENCKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for encryption of the IFS files.

The users (or user groups) which need to encrypt values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Decryption key label (DECKEYLBL)

Indicate the label of the Symmetric Key to use for decrypting the IFS files.

The possible values are:

decryption-key-label Indicate the label of the key to use for decryption.

> WARNING: If specifying a different key label than the label specified for encryption, then that decryption key should contain the same key value as the encryption key.

***ENCKEYLBL** Use the same label as specified on the ENCKEYLBL parameter.

Decryption key store name (DECKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the IFS files.

The users (or user groups) that need access to the decrypted values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***ENCKEYSTR** Use the same Key Store as specified on the ENCKEYSTR parameter.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Change Key Officer (CHGKEYOFR)

The Change Key Officer (CHGKEYOFR) command allows an authorized user to change a Key Officer within the Symmetric Key Management System.

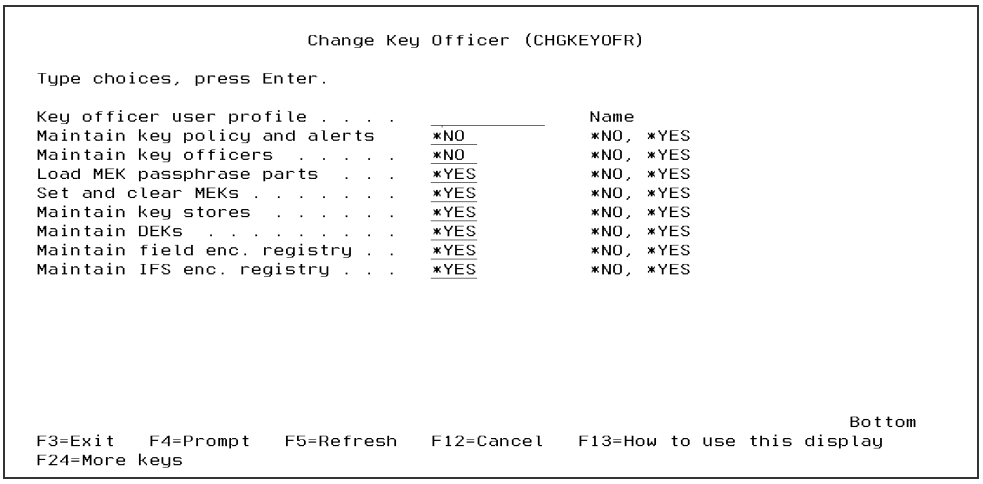
The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key officers” authority setting

When a Key Officer is changed the following will occur:

- If the User Profile is changed to be authorized to at least one option then the User Profile is added to the PCRADMIN Authorization List with *USE Authority.
- If the User Profile is changed to not be authorized to at least one option then the User Profile is removed from the PCRADMIN Authorization List.
- If the User is changed to maintain any of the following: Key Policy, Key Officers, Load MEK Parts or Load MEK, then the User Profile is added to the CRVL001 object with *CHANGE Authority.
- If the User is changed to not maintain any of the following: Key Policy, Key Officers, Load MEK Parts or Load MEK, then the User Profile is removed from the CRVL001 object.

The user profile running this command must have authority to run the ADDAUTLE command or RMVAUTLE command depending on how the entry is changed.



How to Get There

From the Key Policy and Security Menu, choose option **12**, Change Key Officer. Or, prompt (**F4**) the command **CRYPTO/CHGKEYOFR**.

Field Descriptions

Key officer user profile (USRPRF)

Specify the Key Officer's user profile on the System i.

Maintain key policy and alerts (MNTPCYALR)

Indicate if the Key Officer can change the key policy settings and can add, change or delete Alerts.

The possible values are:

***YES** The Key Officer can change the key policy settings and can add, change or delete Alerts.
***NO** The Key Officer cannot change the key policy settings and cannot add, change or delete Alerts.

Maintain key officers (MNTKEYOFR)

Indicate if the Key Officer can add, change and remove other Key Officers.

The possible values are:

***YES** The Key Officer can add, change, and remove other Key Officers.
***NO** The Key Officer cannot add, change, and remove other Key Officers.

Load MEK passphrase parts (LODMEKPRT)

Indicate if the Key Officer can specify passphrase parts for a Master Encryption Key (MEK).

The possible values are:

***YES** The Key Officer can specify passphrase parts for a MEK.
***NO** The Key Officer cannot specify passphrase parts for a MEK.

Set and clear MEKs (MNTMEK)

Indicate if the Key Officer can set (generate) or clear a Master Encryption Key (MEK).

The possible values are:

***YES** The Key Officer can set or clear a MEK.
***NO** The Key Officer cannot set or clear a MEK.

Maintain key stores (MNTKEYSTR)

Indicate if the Key Officer can create Key Stores or translate Key Stores to other Master Encryption Keys (MEKs).

The possible values are:

***YES** The Key Officer can maintain Key Stores.
***NO** The Key Officer cannot maintain Key Stores.

Maintain DEKs (MNTDEK)

Indicate if the Key Officer can create, copy or delete Data Encryption Keys (DEKs)

The possible values are:

***YES** The Key Officer can maintain DEKs.
***NO** The Key Officer cannot maintain DEKs.

Maintain Field Enc. Registry (MNTFLDENC)

Indicate if the Key Officer can maintain the Field Encryption Registry.

The possible values are:

***YES** The Key Officer can maintain the Field Encryption Registry.
***NO** The Key Officer cannot maintain the Field Encryption Registry.

Maintain IFS Enc. Registry (MNTIFSENC)

Indicate if the Key Officer can maintain the IFS Encryption Registry and other automatic IFS Encryption settings. Reserved for future use.

The possible values are:

***YES** The Key Officer can maintain the IFS Encryption Registry and other automatic IFS Encryption settings.
***NO** The Key Officer cannot maintain the IFS Encryption Registry and other automatic IFS Encryption settings.

Change Key Policy (CHGKEYPCY)

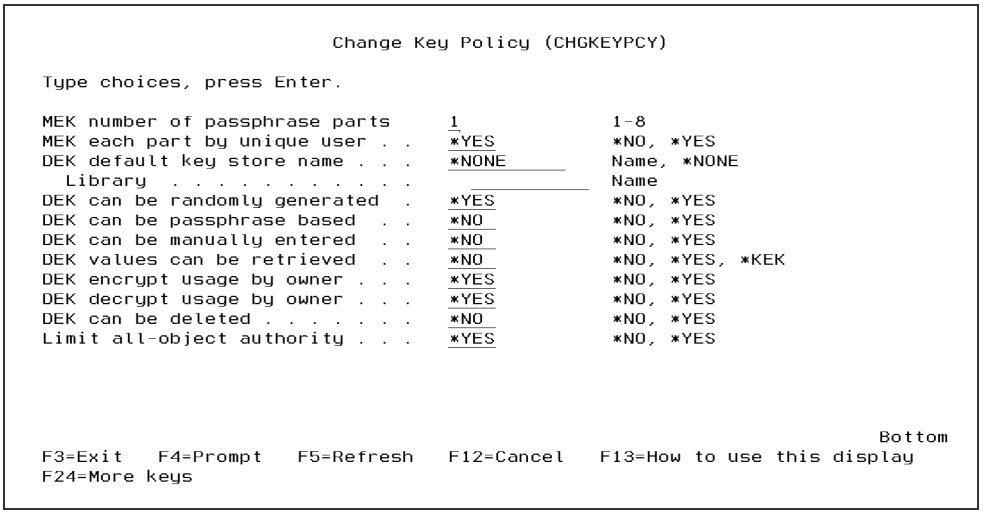
The Change Key Policy (CHGKEYPCY) command allows you to specify the policy settings for the Symmetric Key Management System.

For details and recommended settings, see Configuring Key Policy Settings in Getting Started.

> **NOTE:** Any changes to the Key Policy settings are logged into an audit file.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key policy and alerts” authority setting



How to Get There

From the Key Policy and Security Menu, choose option **1**, Change Key Policy. Or, prompt (**F4**) the command **CRYPTO/CHGKEYPCY**.

Field Descriptions

MEK number of passphrase parts (MEKPRT)

Indicates the number of passphrase parts that must be entered (loaded) before a Master Encryption Key (MEK) can be generated (set).

MEK each part by unique user (MEKUNQUSR)

Indicates whether or not each MEK (Master Encryption Key) passphrase part must be entered (loaded) by a different user profile.

The possible values are:

***YES** Each MEK passphrase part must be loaded by a different user profile.
***NO** Each MEK passphrase part can be loaded by the same user profile.

DEK default key store name (DEKKEYSTR)

Indicates the object name and library of the default Key Store which contains the Data Encryption Keys (DEK).

key-store-name Enter the name of the default Key Store.
***NONE** No default Key Store is specified.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

DEK can be randomly generated (DEKRNDGEN)

Indicates whether a Data Encryption Key (DEK) can be randomly generated with the CRTSYMKEY (Create Symmetric Key) command.

The possible values are:

***YES** A DEK can be randomly generated with the CRTSYMKEY command.
***NO** A DEK cannot be randomly generated with the CRTSYMKEY command.

DEK can be passphrase based (DEKPASBSD)

Indicates whether a Data Encryption Key (DEK) can be generated with a user-entered passphrase with the CRTSYMKEY (Create Symmetric Key) command.

The possible values are:

***YES** A passphrase-based DEK can be generated with the CRTSYMKEY command.
***NO** A passphrase-based DEK cannot be generated with the CRTSYMKEY command.

DEK can be manually entered (DEKMANENT)

Indicates whether a Data Encryption Key (DEK) value can be manually entered with the CRTSYMKEY (Create Symmetric Key) command.

The possible values are:

***YES** A manually entered DEK can be specified on the CRTSYMKEY command.
***NO** A manually entered DEK cannot be specified on the CRTSYMKEY command.

DEK values can be retrieved (DEKRTVVAL)

Indicates whether Data Encryption Key (DEK) values can be retrieved with the EXPSYMKEY (Export Symmetric Key) command.

The possible values are:

***YES** A DEK value can be retrieved.
***NO** A DEK value cannot be retrieved.
***KEK** A DEK value can be retrieved only if it is encrypted with a Key Encryption Key (KEK).

DEK encrypt usage by owner (DEKENCOWN)

Indicates whether the user profile which created a Data Encryption Key (DEK) can use that DEK to encrypt data.

The possible values are:

***YES** The user that created the DEK can use the DEK to encrypt data.
***NO** The user that created the DEK cannot use the DEK to encrypt data.

DEK Decrypt usage by owner (DEKDECOWN)

Indicates whether the user profile which created a Data Encryption Key (DEK) can use that DEK to decrypt data.

The possible values are:

***YES** The user that created the DEK can use the DEK to decrypt data.
***NO** The user that created the DEK cannot use the DEK to decrypt data.

DEK can be deleted (DEKDLTALW)

Indicates whether a Data Encryption Key (DEK) can be deleted from a Key Store.

The possible values are:

***YES** A DEK can be deleted from a Key Store.
***NO** A DEK cannot be deleted from a Key Store.

Limit all-object authority (LMTALLOBJ)

Indicates whether to limit authority for users with *ALLOBJ special authority to Key Stores and Authorization Lists used in the Field Registry.

The possible values are:

***NO** If the user has *ALLOBJ authority, then IBM's 'QSYCUSRA' API will be used to check if the user is authorized to any requested Key Store or Authorization List. Therefore users with *ALLOBJ authority will always be authorized. This is the default setting.

***YES** If the user has *ALLOBJ authority, then Powertech Encryption will perform its own authority check on any requested Key Store or Authorization List. IBM's 'QSYCUSRA' API will not be used. The user profile (or group profile which it belongs) must be specifically listed as an authority entry (with at least *USE authority) on the Key Store or Authorization List.

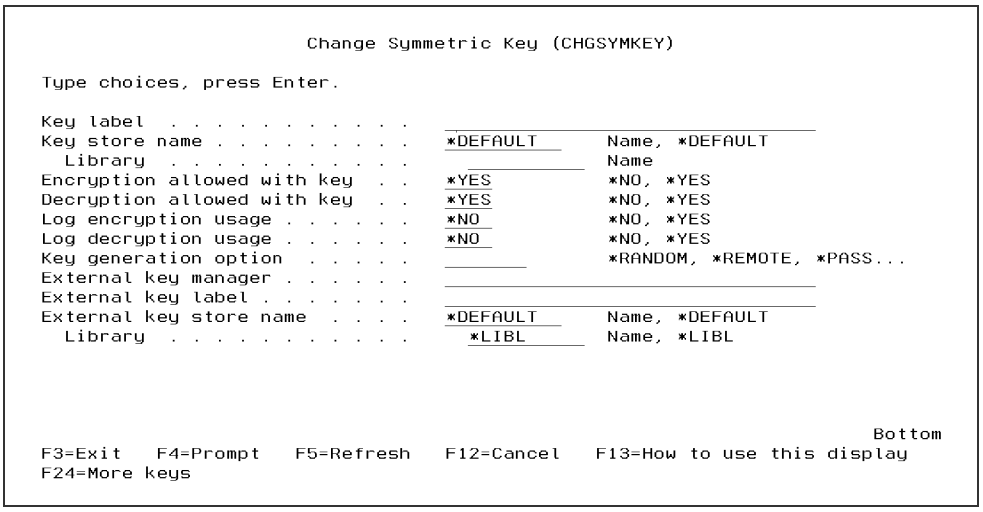
Change Symmetric Key (CHGSYMKEY)

The CHGSYMKEY command allows authorized users to change the attributes of an existing Data Encryption Key (Symmetric Key).

The following users can use the CHGSYMKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain DEKs” authority setting

The user must have *CHANGE authority to the Validation List (*VLDL) object containing the Key Store.



How to Get There

From the Symmetric Encryption Key Menu, choose option **12**.

Options

Key label (KEYLABEL)

Indicate the unique name (label) of the Key.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key.

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

Encryption allowed with key (ENCRYPTALW)

Indicate if this key can be used to encrypt data.

The possible values are:

***YES** This key can be used to encrypt data.

***NO** This key cannot be used to encrypt data.

Decryption allowed with key (DECRYPTALW)

Indicate if this key can be used to decrypt data.

The possible values are:

***YES** This key can be used to decrypt data.
***NO** This key cannot be used to decrypt data.

Log encryption usage (LOGENCRYPT)

Indicate if the usage of the Key for encryption purposes will be logged into the audit journal file.

The possible values are:

***YES** Usage of the key for encryption will be logged.
***NO** Usage of the key for encryption will not be logged.

Log decryption usage (LOGDECRYPT)

Indicate if the usage of the Key for decryption purposes will be logged into the audit journal file.

The possible values are:

***YES** Usage of the key for decryption will be logged.
***NO** Usage of the key for decryption will not be logged.

Key generation option (GENOPT)

Indicate the option used to generate the Symmetric Key.

The possible values are:

***RANDOM** The Key is randomly generated by Powertech Encryption. This is the preferred option.
***REMOTE** The key value is stored in an External Key Manager.
***PASS** The Key is generated based on a user-entered passphrase, iteration count and salt. Uses the PBKDF2 pseudorandom key function as detailed in RFC2898.
***MANUAL** The Key value is manually entered by the user.

External key manager (EXTKEYMGR)

Valid for GENOPT(*REMOTE). Indicate the name of the External Key Manager that contains the remote key. The properties for the External Key Manager must be predefined using the WRKEKM command.

External key label (EXTKEYLBL)

Valid for GENOPT(*REMOTE). Indicate the label (or name) of the remote key in the External Key Manager. The key label is case sensitive.

External key store name (EXTKEYSTR)

Valid for GENOPT(*REMOTE).

If the remote key is in the product, then specify the name of the remote key store that contains the key.

ext-key-store-name Specify the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level on the remote server.

The possible library values are:

library-name Specify the name of the library where the Key Store is located.

Clear IFS Log (CLRIFSLOG)

The Clear IFS Log (CLRIFSLOG) command will clear all records from the CRPFIFSLOG Log File.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

This command requires that you have *USE authority to the CRVL003 Validation List (*VLDL) object which contains the IFS Encryption Registry, and that you have *CHANGE authority to the CRPFIFSLOG File.

How to Get There

From the IFS Utility Menu, choose option **7**, Clear IFS Debug Log. Or, prompt (**F4**) the command **CRYPTO/CLRIFSLOG**.

This command has no parameters. Press Enter to run the command.

Clear Master Encryption Key (CLRMSTKEY)

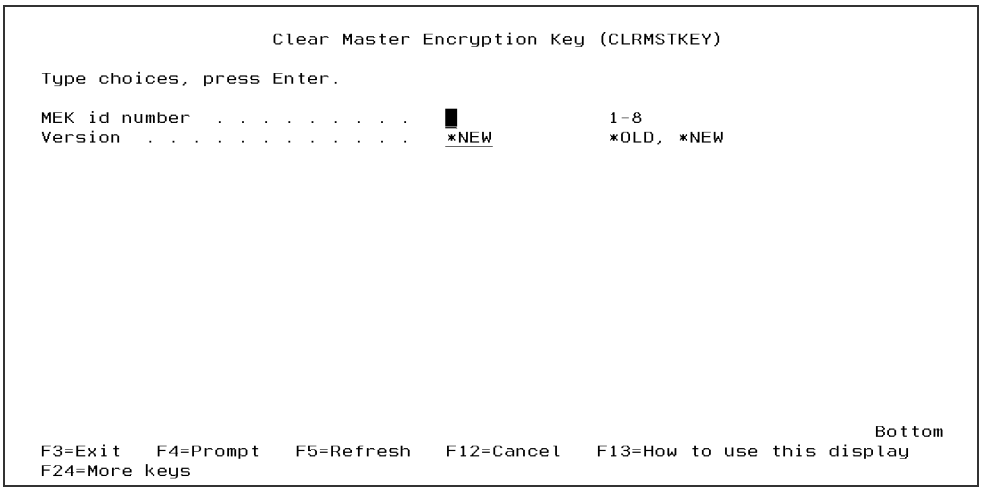
> WARNING: DO NOT clear an *OLD version of a MEK if there are Key Stores still encrypted with this *OLD version. You should first use the TRNKEYSTR command to translate (re-encrypt) the DEKs in the Key Stores which are still encrypted under the *OLD version of the MEK.

The CLRMSTKEY command allows authorized users to clear the *NEW or *OLD version of a Master Encryption Key (MEK).

Before the version of the MEK is cleared, the CRVL001 validation list (*VLDL) object that contains the encrypted Master Encryption Keys is backed up into a Save File object (sequentially named).

The following users can use the CLRMSTKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Set and clear MEKs” authority setting



How to Get There

From the Master Encryption Key Menu, choose option **4**, Clear Master Encryption Key. Or, prompt (**F4**) the command **CRYPTO/CLRMSTKEY**.

Field Descriptions

MEK id number

Indicate the id number of the Master Encryption Key (MEK) to clear.

Version

Specify either the *OLD or *NEW version to clear.

Copy Field Encryption Entry (CPYFLDENC)

The Copy Field Encryption Entry (CPYFLDENC) command allows authorized users to copy Field Encryption Registry entries into the same Registry or to other Registries. This command is especially useful for replicating entries if you are using different library lists (environments) to control which Field Encryption Registry is utilized by your applications.

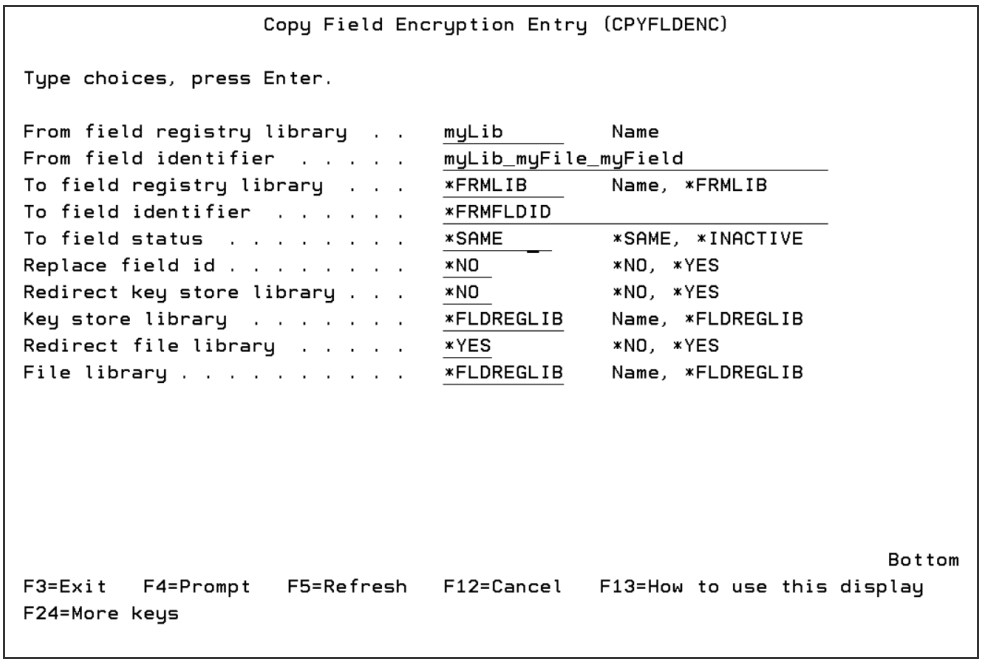
> **NOTE:** The status for the copied Field Identifier will be set by the TOFLDSTS value. If the current status is *ACTIVE and if the TOFLDSTS is left as *SAME, the new status will also be *ACTIVE. If the file is not encrypted in the TOLIB, then the status is invalid and unpredictable results my occur.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have the following object authorities:

- *USE authority to the "From" Field Encryption Registry CRVL002 Validation List (*VLDL) object
- *CHANGE authority to the "To" Field Encryption Registry CRVL002 Validation List (*VLDL) object



How to Get There

From the Field Encryption Menu, choose option **6**.

Options

From field registry library (FRMLIB)

Indicate the library name containing the Field Encryption Registry to copy the entry from.

The possible library values are:

> **library-name** Enter the library name of the Field Encryption Registry (CRVL002) to copy the entry from.

From field identifier (FRMFLDID)

Indicate the name of the entry to copy.

To field registry library (TOLIB)

Indicate the library name containing the Field Encryption Registry to copy the entry to.

The possible library values are:

***FRMLIB** Use the same name specified on the FRMLIB parameter.
library-name Enter the library name of the Field Encryption Registry (CRVL002) to copy the entry to.

NOTE: The library names on the new field entry will be automatically changed to use the library name specified on this parameter.

To field identifier (TOFLDID)

Indicate the name of the entry to create in the "To" Field Encryption Registry.

Rules for field identifier:

- The field identifier does not have to be the same name as the database field name to encrypt.
- The field identifier cannot contain spaces or certain special characters.
- The field identifier can contain underscore characters.
- The field identifier is not case sensitive. It will be stored in upper case.

The possible field values are:

field-id Enter the name of the Field Id (up to 30 characters).
***FRMFLDID** Use the same name specified on the FRMFLDID parameter.

To field status (TOFLDSTS)

Indicate the status to use in the "To" Field identifier.

The possible field values are:

***INACTIVE** The status in the "To" field will be set to *INACTIVE.
***SAME** The status in the "To" field will be the same as the "From" field.

Replace field entry (REPLACE)

Indicate if any existing field entry (with the same name) should be replaced.

The possible values are:

***YES** Replace any existing field entry in the "To" Field Registry which has the same name as the TOFLDID parameter.

> **NOTE:** The destination entry cannot be replaced if it is in *ACTIVE or *PROCESS status.

***NO** Do not replace any existing field entry.

Redirect key store library (REDKSLIB)

Indicate if the Key Store library names on the new field entry should be changed during the copy process.

The possible values are:

***YES** The Key Store library names on the new field entry will be changed to the library name specified on KEYSTRLIB parameter.
***NO** The Key Store library names on the new field entry will not be changed.

Key store library (KEYSTRLIB)

Specify the library name containing the Key Store to use for the new Field entry. Only valid if REDKSLIB(*YES) is specified.

The possible library values are:

***FLDREGLIB** Use the same name specified on the TOLIB parameter.
library-name Specify the library name of the Key Store to use for the field entry.

Redirect file library (REDFILLIB)

Indicate if the file library names to use on the new field entry should be changed during the copy process.

The possible values are:

***YES** The file library names on the new field entry will be changed to the library name specified on the FILLIB parameter.
The following libraries will be changed:

- Database File Library
- External Physical File library (when external file is used)
- External Logical File Library (when external file is used)
- Trigger program library (when triggers are used)

***NO** The library names on the new field entry will not be changed.

File library (FILLIB)

Indicate the library name containing the Files to use for the Field entry. Only valid if REDFILLIB(*YES) is specified.

The possible library values are:

***FLDREGLIB** Use the same name specified on the TOLIB parameter.
library-name Enter the library name of the database file to use for the new field entry.

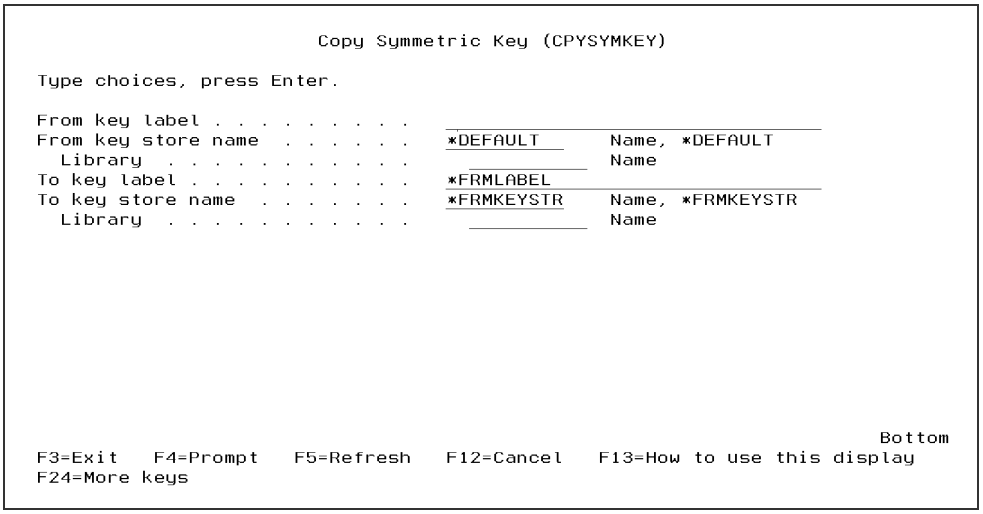
Copy Symmetric Key (CPYSYMKEY)

The CPYSYMKEY command allows authorized users to copy Data Encryption Keys (Symmetric Keys) between Key Stores.

The following users can use the CPYSYMKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain DEKs” authority setting

The user must have *CHANGE authority to the Key Store Validation List (*VLDL) object into which the Key(s) will be copied.



How to Get There

From the Symmetric Encryption Key Menu, choose option **14**.

Options

From key label (FRMLABEL)

Indicate the label of the Symmetric Key to copy.

The possible values are:

from-key-label Indicate the label name of the Symmetric Key to copy.
***ALL** Copy all Symmetric Keys from the Key Store indicated.

From key store name (FRMKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key(s) to copy.

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

To key label (TOLABEL)

The possible values are:

to-key-label Indicate the label of the new Symmetric Key to create.
***FRMLABEL** Use the same label name specified on the FRMLABEL parameter.

To key store name (TOKEYSTR)

Indicate the object name and library of the Key Store to copy the Symmetric Key(s) into.

key-store-name Enter the name of the Key Store.
***FRMKEYSTR** Copy the Symmetric Key(s) into the same Key Store specified in the FRMKEYSTR parameter.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

Create Key Store (CRTKEYSTR)

The Create Key Store (CRTKEYSTR) command allows authorized users to create a Key Store for containing Symmetric Keys.

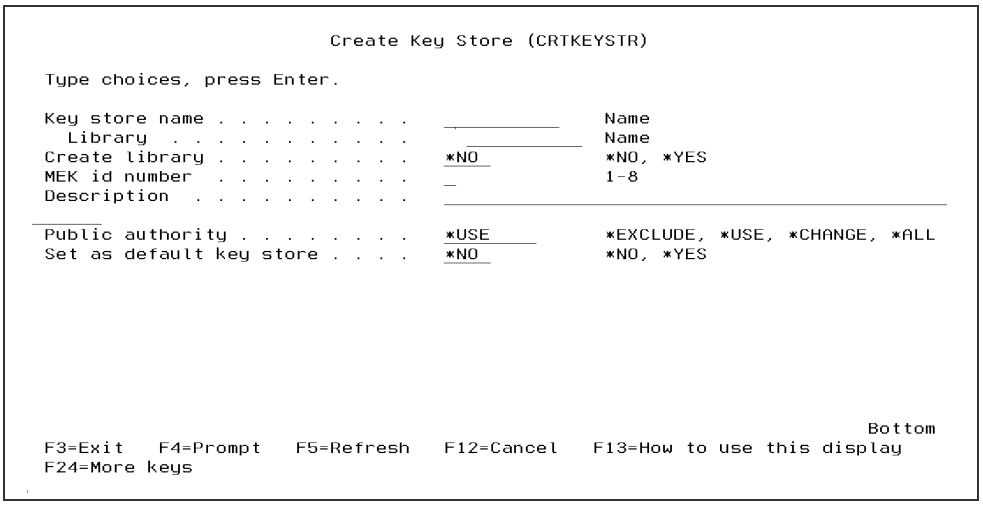
The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain key stores" authority setting.
- When Set as default key store (SETDFT) is *YES A Key Officer who has a *YES specified for the "Maintain key policy" authority setting.

The Key Store is created as a Validation List (*VLDL) object on the System i.

IMPORTANT: This command requires all of the following:

- At least *CHANGE authority to the library you are creating this key store in;
- Authority to IBM's CRTVLDL (Create Validation List) command; and
- Authority to IBM's CRTLIB (Create Library) command when CRTLIB (Create Library) is set to *YES.



How to Get There

From the Symmetric Encryption Key Menu, choose option **1**.

Options

Key store name (KEYSTR)

Indicate the name and library of the Key Store, which is created as a Validation List (*VLDL) object on the System i.

Create library (CRTLIB)

Indicate to create the key store library if it does not exist.

The possible values are:

***YES** Create the library if it does not exist. If the library already exists, then ignore the parameter.
***NO** This is the Default. Do not attempt to create the library if it does not exist.

MEK id number (MEKID)

Indicate the id number of the Master Encryption Key (MEK) which will be used to encrypt any Symmetric Keys which are added (created) to the Key Store.

The possible values are:

mek-id-number Indicate a number from 1-8. A *CURRENT version of the MEK must exist.

Description (TEXT)

Indicate the description for the Key Store object.

Public authority (AUT)

Indicate the public authority for the Key Store *VLDL object.

TIP: Specify *USE to allow the public to use the Key Store.

The possible values are:

***USE** Grants *PUBLIC *USE authority for the Key Store *VLDL object.
***CHANGE** Grants *PUBLIC *CHANGE authority for the Key Store *VLDL object.
***ALL** Grants *PUBLIC *ALL authority for the Key Store *VLDL object.
***EXCLUDE** Grants *PUBLIC the *EXCLUDE authority for the Key Store *VLDL object. If this option is selected and the job's current user does not have private authority to the key store or the *ALLOBJ special authority, the job will encounter a "hard" SQL or Db2 error and may abort when it tries to read or modify encrypted data.

Set as default key store (SETDFT)

Indicate to set the new key store as the default key store in the key policy.

The possible values are:

***YES** Set the new key store as the default in the key policy.
***NO** Do not set the new key store as the default key store in the key policy.

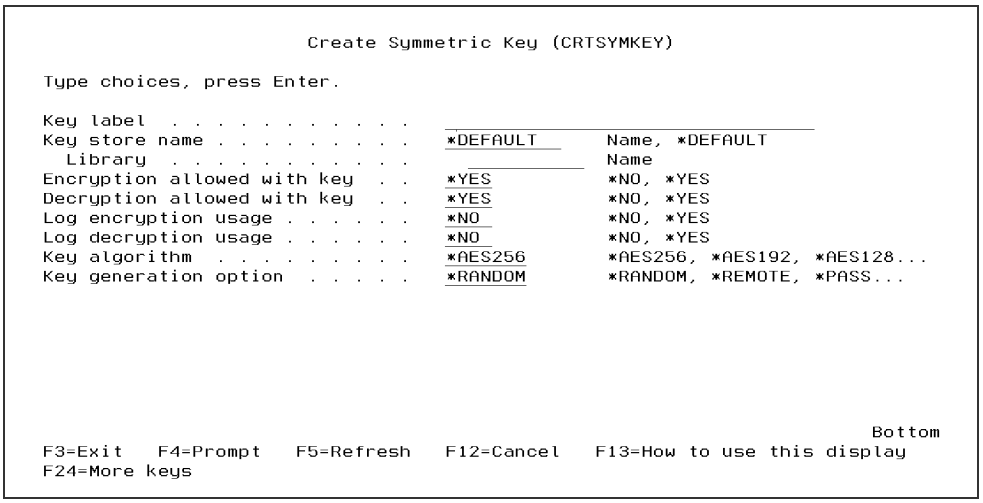
Create Symmetric Key (CRTSYMKEY)

The CRTSYMKEY command allows authorized users to create a new Data Encryption Key (Symmetric Key) and place it into a Key Store.

The following users can use the CRTSYMKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain DEKs” authority setting

The user must have *CHANGE authority to the Validation List (*VLDL) object containing the Key Store, into which the Key will be created.



How to Get There

From the Symmetric Encryption Key Menu, choose option **11**.

Options

Key label (KEYLABEL)

Indicate the unique name (label) of the Key up to 30 characters.

Rules for label name:

- The label cannot contain spaces or certain special characters.
- The label can contain underscore characters.
- The label is not case sensitive. It will be stored in upper case.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store to save the Symmetric Key into.

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level. You must have *CHANGE authority to the Key Store *VLDL object.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

Encryption allowed with key (ENCRYPTALW)

Indicate whether the Key can be used for encryption purposes.

The possible values are:

***YES** This key can be used to encrypt data.
***NO** This key cannot be used to encrypt data.

Decryption allowed with key (DECRYPTALW)

Indicate whether the key can be used for decryption purposes.

The possible values are:

***YES** This key can be used to decrypt data.
***NO** This key cannot be used to decrypt data.

Log encryption usage (LOGENCRYPT)

Indicate if the usage of the Key for encryption purposes will be logged into the audit journal file.

The possible values are:

***YES** Usage of the key for encryption will be logged.
***NO** Usage of the key for encryption will not be logged.

> NOTE: Auditing will have an additional impact on application performance and will consume disk space.

Log decryption usage (LOGDECRYPT)

Indicate whether the usage of the Key for decryption purposes will be logged into the audit journal file.

The possible values are:

***YES** Usage of the key for decryption will be logged.
***NO** Usage of the key for decryption will not be logged.

> NOTE: Auditing will have an additional impact on application performance and will consume disk space.

Key algorithm (ALGORITHM)

Indicate the encryption algorithm to use for creating the Symmetric Key.

The possible values are:

***AES256** Use Advanced Encryption Standard (AES) algorithm and a 256 bit key.
***AES192** Use Advanced Encryption Standard (AES) algorithm and a 192 bit key.
***AES128** Use Advanced Encryption Standard (AES) algorithm and a 128 bit key.
***TDES** Use Triple Data Encryption Standard (TDES) algorithm.

Key generation option (GENOPT)

Indicate the option used to generate the Symmetric Key.

The possible values are:

***RANDOM** The Key is randomly generated by Powertech Encryption. This is the preferred option.
***REMOTE** The key value is stored in an External Key Manager.

***PASS** The Key is generated based on a user-entered passphrase, iteration count and salt. Uses the PBKDF2 pseudorandom key function as detailed in RFC2898.
***MANUAL** The Key value is manually entered by the user.

Passphrase (PASSPHRASE)

The Passphrase to use for generating the Symmetric Key. Valid for GENOPT(*PASS). The passphrase can be between 1 and 256 characters in length.

Salt (SALT)

The salt value to use for generating the Symmetric Key. Valid for GENOPT(*PASS). The salt value can be between 1 and 32 characters in length.

Iteration count (ITER)

The iteration count to use for generating the Symmetric Key. Valid for GENOPT(*PASS). The iteration count indicates the number of times this function loops through the key creation process. The higher the iteration count, the more difficult it will be for an unauthorized party to reverse-engineer the Symmetric Key. The iteration count can be a number from 1 to 50000.

ASCII input format (ASCII)

Indicate the character set of the passphrase and salt. Valid for GENOPT(*PASS).

The possible values are:

***YES** The passphrase and salt uses the ASCII character set.
***NO** The passphrase and salt uses the EBCDIC character set.

Character format used (KEYVALFMT)

Indicate the format of the symmetric key value to enter. Valid for GENOPT(*MANUAL).

The possible values are:

***HEX** The key value will be entered in hexadecimal format.
***CHAR** The key value will be entered in character format.
***BASE64** The key value will be entered in base64 format.

Key value (KEYVAL)

Indicate the actual value of the key. For AES algorithms. Valid for GENOPT(*MANUAL).

Key value (TDKEYVAL)

Indicate the actual value of the key. For TDES algorithm. Valid for GENOPT(*MANUAL).

KEK key label (KKEYLABEL)

Valid for GENOPT(*MANUAL).

Indicate the label of the Key Encryption Key (KEK) which the Symmetric key is encrypted with.

The possible values are:

key-label Enter the label of the KEK.
***NONE** The Symmetric Key is not encrypted with a KEK.

KEK key store name (KKEYSTR)

Indicate the object name and library of the Key Store which contains the Key Encryption Key (KEK).

kek-key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

External key manager (EXTKEYMGR)

Valid for GENOPT(*REMOTE). Indicate the name of the External Key Manager that contains the remote key. The properties for the External Key Manager must be predefined using the WRKEKM command.

External key label (EXTKEYLBL)

Valid for GENOPT(*REMOTE). Indicate the label (or name) of the remote key in the External Key Manager. The key label is case sensitive.

External key store name (EXTKEYSTR)

Valid for GENOPT(*REMOTE). If the remote key is in the product, then specify the name of the remote key store that contains the key.

ext-key-store-name Specify the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level on the remote server.

The possible library values are:

library-name Specify the name of the library where the Key Store is located.

Create external key (CRTEXTKEY)

Indicate if you would like to create the key on the remote (external) key manager.

The possible values are:

***YES** When creating a key type of *VORMETRIC, a random key will be generated and inserted into the Vormetric Vault. This key entry will point to actual key on the external (remote) key manager. When creating a key type of *KMIP, a key will be generated on the external key manager. This key entry will point to actual key on the external (remote) key manager.

***NO** The key will not be created on the remote key manager. A key with the External key label (EXTKEYLBL) should already exist on the remote key manager.

Deactivate Field Encryption (DCTFLDENC)

WARNING: The DCTFLDENC command will lock the database file and will perform a mass-decryption of the current field values. You should only run this command when no applications are utilizing the database file.

The DCTFLDENC command allows authorized users to deactivate a field entry in the Encryption Registry.

The following users can use the DCTFLDENC command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)

- A Key Officer that has a *YES specified for the “Maintain Field Enc. Registry” authority setting

This command can only be used for field entries that have an *ACTIVE status.

It is strongly recommended to submit this command to batch.

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object, which contains the Field Encryption Registry.

IMPORTANT: Before using the DCTFLDENC command to decrypt production data, do the following steps:

1. Make sure you have *ALL authority to the database file containing the field to decrypt.
2. Make sure you have at least *USE authority to the Key Store(s) which hold the Data Encryption Keys (DEKs) that will be used to decrypt the data. You can use the WRKFLDKEY command to find out which Key Store(s) and DEKs are used to decrypt the field values. If you created any of these DEKs yourself, in which you are considered the owner of these DEK(s), then the “DEK decrypt usage by owner” setting (viewable in the DSPKEYPCY command) must be a *YES.
3. If SQL triggers or DB2 Field Procedures are not used to auto-encrypt the field values: All programs that maintain (add/change/delete) records in the database file should be modified to not call Powertech Encryption for IBM i’s APIs for encrypting the field values.
4. If a DB2 Field Procedure is not used to auto-decrypt the field values: All programs that required access to the decrypted field values should be modified to not call Powertech Encryption for IBM i’s decryption APIs.
5. Within a test environment, you should have tested DCTFLDENC and tested your applications thoroughly with decrypted values.
6. No applications or users should be currently using the database file containing the field to decrypt.
7. The DCTFLDENC command will perform a mass decryption of the current field values. You should allocate enough downtime for the DCTFLDENC to execute. Execution times will vary depending on the processor speed of your system, the number of records in your database file, and other activity running on the system at the time. In order to estimate the execution time for DCTFLDENC, you should run the DCTFLDENC command over some test data first.

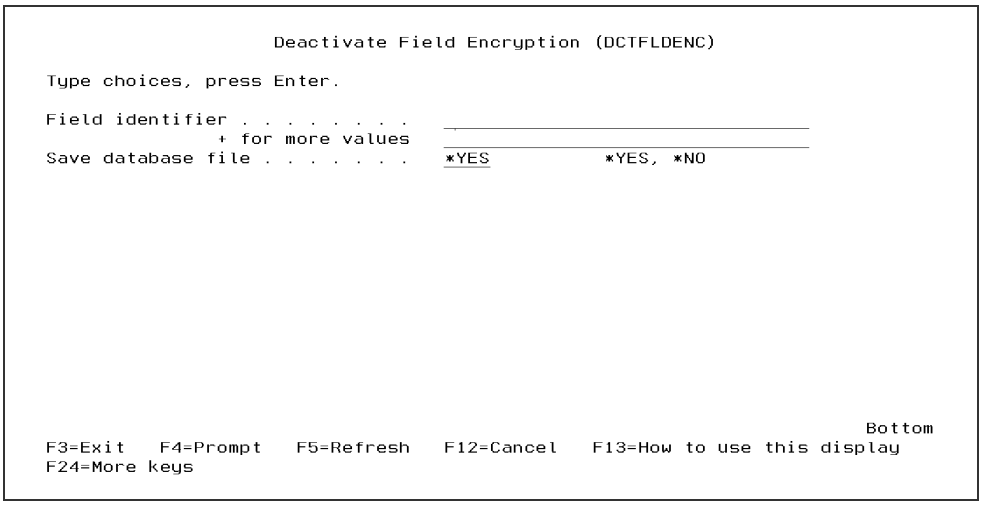
WARNING: If deactivating a field with a DB2 Field Procedure, and if there are already other DB2 Field Procedures on the file, then you should have at least *USE authority to the 'Full' Authorization Lists assigned to those other fields, as well as at least *USE authority to the Key Stores that contain the encryption and decryption Keys used by those fields, and *USE authority to the library that contains the Key Store. This is because IBM's ALTER TABLE statement (used in the deactivate process) runs the decrypt/encrypt processes for all fields that have a DB2 Field Procedure. **Failure to have proper authorities will cause loss of data.**

Recommendations for DCTFLDENC command:

- Run DCTFLDENC in batch using the SBMJOB command.
- Specify *YES for the "Save database file" parameter to save a copy of the database file into a Save File before the deactivation process. This option is important for error recovery purposes.
- Ensure that enough disk space is available for a saved copy of the database file.

Do the following steps to deactivate a field entry in the Encryption Registry:

1. Prompt (**F4**) the command **CRYPTO/DCTFLDENC**.
2. Enter the Field identifier to deactivate, then press Enter.



The DCTFLDENC command performs the following steps:

1. Obtains an exclusive (*EXCL) lock on the database file containing the field to decrypt.
2. If an external file was used for storing the encrypted values, then that external file will be backed up into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.

3. Optional: Creates a backup of the database file (containing the field to decrypt) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
4. Performs a mass decryption of the current field values. If a DB2 Field Procedure was specified for the field, then it will be removed from the field at that time.
5. If SQL triggers were used for automatically encrypting the field values, then the SQL triggers will be removed from the file.
6. If an external file was used for storing the encrypted values, then that external file will be deleted.
7. The exclusive lock will be released on the database file containing the decrypted field.
8. The status of the field entry will be changed to *INACTIVE.

Notes on DCTFLDENC:

- After the DCTFLDENC command completes: Once you have determined that your applications are working properly with the decrypted values, you can remove the Save files (created in steps 2 and 3 above) containing the backup of the external and database files.

Deactivate Field Entries (DCTFILFLDE)

The Deactivate File Fields Encryption (DCTFILFLDE) command will deactivate any *ACTIVE entries in the Field Encryption Registry for the file that use Field Procedures.

It is strongly recommended to submit this command to batch.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

IMPORTANT: Before using the DCTFLDENC command to encrypt production data, do the following steps:

1. Make sure you have *ALL authority to the database file containing the field to encrypt.
2. Within a test environment, you should have tested DCTFILFLDE, tested any API calls needed for encryption/decryption and tested your applications thoroughly with encrypted values.
3. No applications or users should be currently using the database file containing the field to encrypt.
4. The DCTFILFLDE command will perform a mass decryption of the current field values. You should allocate enough downtime for the DCTFILFLDE to execute. Execution times will vary depending on the processor speed of your system, the number of records in your database file, and other activity running on the system at the time. In order to estimate the execution time for DCTFILFLDE, you should run the DCTFILFLDE command over some test data first.
5. Check (and double check) the field entry settings using the DSPFLDENC command. Especially make sure the database file name, field name, type and length is correct.

   IMPORTANT: When activating a field using a DB2 Field Procedure, and if there are already other DB2 Field Procedures on the file, then you should have at least *USE authority to the 'Full' Authorization Lists assigned to those other fields, as well as at least *USE authority to the Key Stores that contain the encryption and decryption Keys used by those fields. This is because IBM's ALTER TABLE statement (used in the activation process) runs the decrypt/encrypt processes for all fields that have a DB2 Field Procedure. Failure to have proper authorities will cause loss of data.

The DCTFILFLDE command performs the following primary steps:

1. Obtains an exclusive (*EXCL) lock on the database file containing the field to encrypt.
2. Optional: Creates a backup of the database file (containing the field to encrypt) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
3. Performs a mass encryption of the current field values in the database file. If a DB2 Field Procedure is specified for the field, then it will be added to the field at that time.
4. The exclusive lock will be released on the database file containing the encrypted field.
5. The status of the field entries will be changed to *ACTIVE.

How to Get There

On the File Field Encryption Menu, choose option **4**, Deactivate File Encryption Entry(s).

Options

Deactivate file name (DCTFILE)

Specify the name of the file that contains the field(s) to deactivate.

The possible values are:

file-name The name of the file that contains the field(s) to deactivate.

The possible library values are:

library-name Enter the name of the library where the file is located.
***LIBL** Locate the file within the library list.

Save database file (SAVDTA)

Indicate if the database file (containing the field to encrypt) should be saved (backed up) into a Save File before the activation process begins. It is highly recommended to save the database file for error recovery purposes.

The possible values are:

***YES** Save the database file into a Save File before deactivation begins.

> **NOTE:**
> - The created Save File will be named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
> - Before using this option, ensure that enough disk space is available for a saved copy of the database file.

***NO** Do not save the database file before the deactivation process begins.

Deactivate File Encryption (DCTFILFLDE)

The Deactivate File Fields Encryption (DCTFILFLDE) command will deactivate any *ACTIVE entries in the Field Encryption Registry for the file that uses Field Procedures.

It is strongly recommended to submit this command to batch.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

IMPORTANT: Before using the DCTFILFLDE command to encrypt production data, do the following steps:

1. Make sure you have *ALL authority to the database file containing the field to encrypt.
2. Within a test environment, you should have tested DCTFILFLDE, tested any API calls needed for encryption/decryption and tested your applications thoroughly with encrypted values.
3. No applications or users should be currently using the database file containing the field to encrypt.
4. The DCTFILFLDE command will perform a mass decryption of the current field values. You should allocate enough downtime for the DCTFILFLDE to execute. Execution times will vary depending on the processor speed of your system, the number of records in your database file, and other activity running on the system at the time. In order to estimate the execution time for DCTFILFLDE, you should run the DCTFILFLDE command over some test data first.
5. Check (and double check) the field entry settings using the DSPFLDENC command. Especially make sure the database file name, field name, type and length is correct.

   IMPORTANT: When activating a field using a DB2 Field Procedure, and if there are already other DB2 Field Procedures on the file, then you should have at least *USE authority to the 'Full' Authorization Lists assigned to those other fields, as well as at least *USE authority to the Key Stores that contain the encryption and decryption Keys used by those fields. This is because IBM's ALTER TABLE statement (used in the activation process) runs the decrypt/encrypt processes for all fields that have a DB2 Field Procedure. Failure to have proper authorities will cause loss of data.

The DCTFILFLDE command performs the following primary steps:

1. Obtains an exclusive (*EXCL) lock on the database file containing the field to encrypt.
2. Optional: Creates a backup of the database file (containing the field to encrypt) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
3. Performs a mass decryption of the current field values in the database file. If a DB2 Field Procedure is specified for the field, then it will be removed from the field at that time.
4. The exclusive lock will be released on the database file containing the encrypted field.
5. The status of the field entries will be changed to *INACTIVE.

Deactivate file name (DCTFILE)

Specify the name of the file that contains the field(s) to deactivate.

The possible values are:

file-name The name of the file that contains the field(s) to deactivate.

The possible library values are:

library-name Enter the name of the library where the file is located.
***LIBL** Locate the file within the library list.

Save database file (SAVDTA)

Indicate if the database file (containing the field to encrypt) should be saved (backed up) into a Save File before the deactivation process begins. It is highly recommended to save the database file for error recovery purposes.

The possible values are:

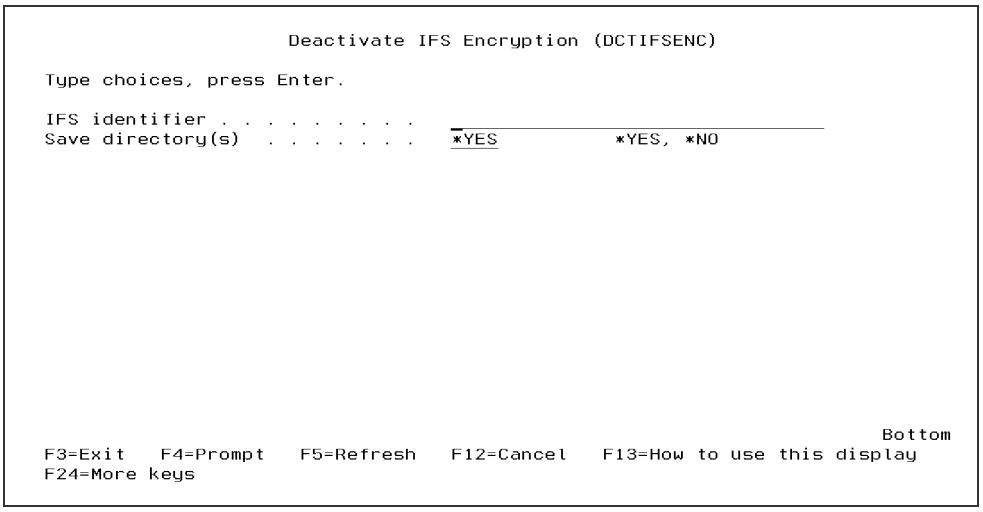
***YES** Save the database file into a Save File before deactivation begins.

> **NOTE:** The created Save File will be named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999. Before using this option, ensure that enough disk space is available for a saved copy of the database file.

***NO** Do not save the database file before the deactivation process begins.

Deactivate IFS Encryption (DCTIFSENC)

The Deactivate IFS Encryption (DCTIFSENC) command allows authorized users to deactivate an *ACTIVE entry in the IFS Encryption Registry.



It is strongly recommended to submit this command to batch.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

IMPORTANT: The following specific authorities are required for users running this command:

- *RWX data authority and *ALL object authority to the directories and files in the Source and Target that are to be decrypted.
- *CHANGE authority to the CRVL003 (Validation List object which contains the IFS Encryption Registry), CRPFIFS, CRPFIFSL1, CRPFIFSL2, CRPFIFSL3, CRPFIFSL4, CRPFIFS2 and CRPFIFSLOG files, which will be updated during this process.
- *USE authority to the Authorization List, Keys and Keystore that are assigned to this entry. “

WARNING: Before using the DCTIFSENC command to decrypt production data, do the following steps:

1. Make sure you have *ALL authority to the Source and Target directories containing the IFS files to decrypt.
2. Make sure you have at least *USE authority to the Key Store(s) which hold the Data Encryption Keys (DEKs) that will be used to decrypt the data. You can use the WRKIFSKEY command to find out which Key Store(s) and DEKs are used to decrypt the IFS values. If you created any of these DEKs yourself, in which you are considered the owner of these DEK(s), then the “DEK decrypt usage by owner” setting (viewable in the DSPKEYPCY command) must be a *YES.
3. Make sure you have at least *USE authority to the Authorization List used to allow for decryption of the data.
4. Within a test environment, you should have tested DCTIFSENC and tested your applications thoroughly with decrypted values.
5. No applications or users should be currently using the directory containing the IFS files to decrypt.
6. The DCTIFSENC command will perform a mass decryption of the current IFS files. You should allocate enough downtime for the DCTIFSENC to execute. Execution times will vary depending on the processor speed of your system, the number of files, and other activity running on the system at the time. In order to estimate the execution time for DCTIFSENC, you should run the DCTIFSENC command over some test data first.

The DCTIFSENC command performs the following primary steps:

1. Optional: Creates a backup of the IFS directory and subdirectories if INCSUBDIR is *YES (containing the source files) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
2. Optional: Creates a backup of the IFS directory and subdirectories if INCSUBDIR is *YES (containing the encrypted files) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
3. Journaling will be stopped for the directories.
4. Performs a mass decryption of the IFS files in the directories.
5. The status of the IFS registry entry will be changed to *INACTIVE.

How to Get There

From the IFS Encryption Menu, choose option **11**. Or, prompt (**F4**) the command **CRYPTO/DCTIFSENC**.

Options

IFS identifier (IFSID)

Specify the IFS identifier to deactivate.

Save database file (SAVDTA)

Indicate if the directory(s) containing the source files and the target directory(s) containing the encrypted files should be saved (backed up) into a Save File before the deactivation process begins. It is highly recommended to save the files for error recovery purposes. The source files and the target (encrypted files) will be saved into two different backup files.

The possible values are:

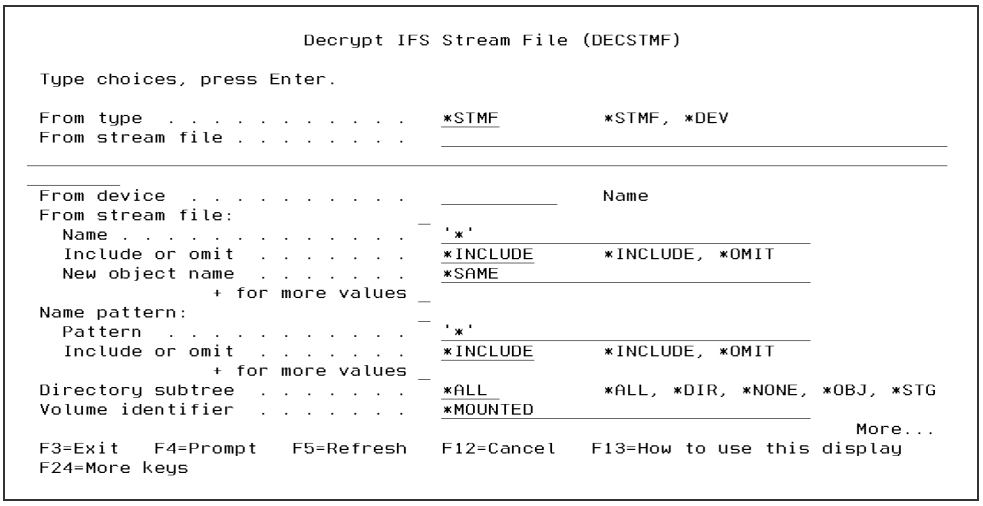
***YES** Save the IFS files into a Save File before deactivation begins.

> **NOTE:**
> - The created Save Files will be named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999. The backup files will be placed in the CRYPTO library.
> - Before using this option, ensure that enough disk space is available for a saved copy of the files.

***NO** Do not save the files before the deactivation process begins.

Decrypt IFS Stream File (DECSTMF)

The DECSTMF command allows authorized users to decrypt IFS stream files, which were encrypted with the ENCSTMF command. Files can be restored/decrypted from a device (physical or virtual) or the IFS. Either a Symmetric Key or a Password can be specified for the decryption.



Monitoring for Errors

When executing the DECSTMF command within a CL program, you can trap for errors by monitoring for message id CRE0701.

Auditing

If a Symmetric Key is used for the DECSTMF command and “Log decryption usage” is enabled for the Symmetric Key, then an audit log entry will be generated in the Powertech Encryption for IBM i journal file each time the Key is used for decryption.

Each audit entry will indicate the Label and Key Store of the Symmetric Key which was used, along with the user, date, time, job number and job name.

How to Get There

On the Library/Object/File Encryption Menu, choose option **6**.

Options

From file type (FROMTYPE)

Indicate the type of file to decrypt.

The possible file values are:

***STMF** Decrypt an IFS stream file.
***DEV** Decrypt data from a device.

From stream file (FROMSTMF)

Specify the location of the stream (IFS) file that holds the stream (IFS) files to decrypt.

The possible values are:

ifs-file-name Specify the absolute path of the IFS file that holds the files to decrypt. For instance: '/ABCcompany/Files/Payroll.AES'

From Device (FROMDEV)

Indicate the name of the device to decrypt data from.

Objects (OBJ)

Specifies the objects to be restored. You can specify an object name pattern for the path name to be used. When a path name is specified that could match many objects, you can specify a value for the Name pattern (PATTERN) parameter to subset the objects that are to be saved. A maximum of 25 path names can be specified.

Element 1: Name

The possible values are:

'*' The objects in the current directory are saved.
path-name Specify an object path name or a pattern that can match many names.

Element 2: Include or omit

Specifies whether names that match the pattern should be included or omitted from the operation. Note that in determining whether a name matches a pattern, relative name patterns are always treated as relative to the current working directory.

The possible values are:

***INCLUDE** The objects that match the object name pattern are to be saved, unless overridden by an *OMIT specification.
***OMIT** The objects that match the object name pattern are not saved.

This overrides an *INCLUDE specification and is intended to be used to omit a subset of a previously selected pattern.

Element 3: New object name

Specifies the new path name of the object.

The possible values are:

***SAME** The objects are to be restored with the same names they had when they were saved.

path-name Specify the path name with which to restore the object. If a pattern is specified in the first element, the new path name must be the directory into which to restore any objects that match the pattern. If an object name is specified in element 1, each component in the new path name must exist with the exception of the last component. If the object described in the last component doesn't exist, it will be restored as new.

Name pattern (PATTERN)

Specifies one or more object name patterns to be used to subset the objects to be saved. The Objects (OBJ) parameter defines the set of candidate objects. A maximum of 25 values can be specified for this parameter.

Element 1: Pattern

The possible values are:

'*' All objects which qualify for the operation are included or omitted.
character-value Specify an object name or a pattern that can match many names.

Element 2: Include or omit

Specifies whether names that match the pattern should be included or omitted from the operation.

The possible values are:

***INCLUDE** Only objects which are included by the OBJ parameter to be saved, and match the PATTERN parameter are included in the save, unless overridden by an *OMIT specification.

***OMIT** All objects which are included by the OBJ parameter are included in the save except those objects which match the PATTERN parameter. This overrides an *INCLUDE specification and is intended to be used to omit a subset of a previously selected pattern.

Directory subtree (SUBTREE)

Specifies whether directory subtrees are included in the restore operation.

The possible values are:

***ALL** The entire subtree of each directory that matches the object name pattern is processed. The subtree includes all subdirectories and the objects within those subdirectories.

***DIR** The objects in the first level of each directory that matches the object name pattern are processed. The subdirectories of each matching directory are included, but the objects in the subdirectories are not included.

***NONE** No subtrees are included in the restore operation. If a directory matches the object name pattern specified, the objects in the directory are included. If the directory has subdirectories, neither the subdirectories nor the objects in the subdirectories are included.

***OBJ** Only the objects that match the object name pattern will be processed. If the object name pattern specifies a directory, objects in the directory are not included.

***STG** The objects that match the object name pattern are processed along with the storage for related objects. Objects can only be restored using this value if they were saved with SUBTREE(*STG).

Volume identifier (VOL)

Specifies the volume identifier on which the data is being restored.

The possible values are:

***NONE** The data is restored from the volume placed in the device.
volume-identifier Specify the identifier of the volume for the restore operation.

Sequence number (SEQNBR)

Specifies the sequence number to use as the starting point for the restore operation.

The possible values are:

***NEXT** Search for the encrypted data in the next sequence number on the tape.
sequence-number Specify the sequence number of the stored encrypted data. Valid values range from 1 through 16777215.

Label (LABEL)

Specifies the name that identifies the data file on the tape that is to be used for the restore operation.

The possible values are:

data-file-identifier Specify the data file identifier of the data file used for the save operation. A maximum of 17 characters can be used.

End of media option (ENDOPT)

Specifies the operation that is automatically done on the tape after the restore operation ends.

The possible values are:

***REWIND** The tape is automatically rewound, but not unloaded, after the operation has ended.
***LEAVE** The tape does not rewind or unload after the operation ends. It remains at the current position on the tape drive.
***UNLOAD** The tape is automatically rewound and unloaded after the operation ends.

Allow object differences (ALWOBJDIF)

Specifies whether differences are allowed between the saved objects and the restored objects.

> **NOTE:** 1. To use this parameter, you need all object (*ALLOBJ) special authority. 2. If differences are found, the final message for the restore operation is an escape message rather than the normal completion message.

The types of differences include:

- Authorization list: The authorization list of an object on the system is different than the authorization list of an object from the save operation. Or the system on which a new object with an authorization list is being restored is different from the system on which it was saved.

- Ownership: The owner of an object on the system is different than the owner of an object from the save operation.
- Primary Group: The primary group of an object on the system is different than the primary group of an object from the save operation.

The possible values are:

***NONE** None of the differences listed above are allowed on the restore operation. See the description of each individual value to determine how differences are handled.

***ALL** All of the differences listed above are allowed on the restore operation. See the description of each individual value to determine how differences are handled.

***AUTL** Authorization list differences are allowed. If an object already exists on the system with a different authorization list than the saved object, the object is restored with the authorization list of the object on the system. New objects that are being restored to a system that is different from which they were saved are restored and linked to their authorization list. If the authorization list does not exist on the new system, the public authority is set to *EXCLUDE.

If this value is not specified, authorization list differences are not allowed. If an object already exists on the system with a different authorization list than the saved object, the object is not restored. New objects that are being restored to a system that is different from which they were saved are restored, but they are not linked to the authorization list, and the public authority is set to *EXCLUDE.

***OWNER** Ownership differences are allowed. If an object already exists on the system with a different owner than the saved object, the object is restored with the owner of the object on the system. If this value is not specified, ownership differences are not allowed. If an object already exists on the system with a different owner than the saved object, the object is not restored.

***PGP** Primary group differences are allowed. If an object already exists on the system with a different primary group than the saved object, the object is restored with the primary group of the object on the system. If this value is not specified, primary group differences are not allowed. If an object already exists on the system with a different primary group than the saved object, the object is not restored.

Use key or password (USEKEYPAS)

Indicate to use a either a key from a key store or a password to decrypt the file. The default is *KEY.

The possible values are:

***KEY** Use a key from a key store to decrypt the file.

***PASS** Use a password to decrypt the file.
Key label (KEYLABEL) Indicate the label of the key to use for decrypting the data.

The possible values are:

key-label Enter the name of a Key Label in the Key Store.
***AUTO** Use the Key Label information stored in the data during encryption.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the data.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

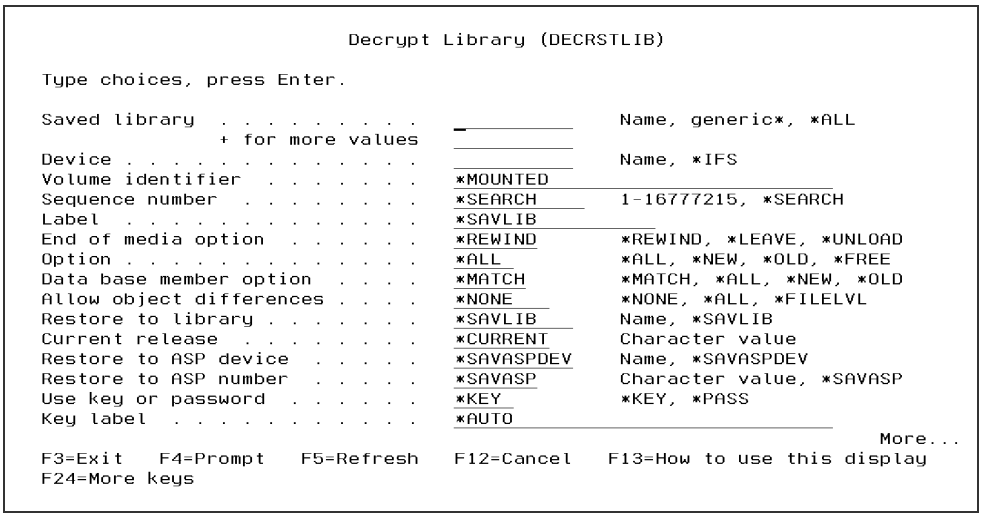
library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.
Password (PASSWORD) Indicate the password to use to decrypt the data.

Decrypt Library (DECRSTLIB)

The DECRSTLIB command allows authorized users to restore and decrypt one or more libraries that were encrypted with the ENCSAVLIB command. Libraries can be restored from a device (physical or virtual) or the IFS. Either a Symmetric Key or a Password can be specified for the decryption process.

Do the following steps to decrypt libraries:

1. Prompt (**F4**) the command of CRYPTO/DECRSTLIB.
2. Press **F1** on any parameter for complete online help text.
3. Press Enter after the parameter values are entered.



The DECRSTLIB command will restore the whole library, including the library description, object descriptions and contents of the objects in the library.

When the owner profile does not exist on the system, the user profile of the system default owner (QDFTOWN) becomes the default owner of any object being restored in the system.

If an object already exists in the library in which it is being restored, the public and private authorities of the existing object are retained. If the object does not exist in the library, all public authorities are restored, but private authorities must be granted again.

If an object is being restored over an existing object on the system, the object auditing value of the existing object is kept. If the object is being restored as new to the system, the object auditing value is restored from the media. Additionally, if the object is a library, the default auditing value for each object created in the library is restored if the library is being restored as new; otherwise, the default auditing value is restored from the media.

Make sure the QSYSWRK subsystem is active for support of the DECRSTLIB command.

Monitoring for Errors

When executing the DECRSTLIB command within a CL program, you can trap for errors by monitoring for message ids. The failure message ids for the DECRSTLIB command are listed below:

CRE0714 - Library(s) were not decrypted. Review JOB LOG.
CRE3773 - &1 objects restored. &2 not restored to &3.
CRE3779 - &1 Library(s) restored. &2 Library(s) were partially restored. &3 not restored.

Auditing

If a Symmetric Key is used for the DECRSTLIB command and “Log decryption usage” is enabled for the Symmetric Key, then an audit log entry will be generated in the Powertech Encryption for IBM i journal file each time the Key is used for decryption.

Each audit entry will indicate the Label and Key Store of the Symmetric Key that was used, along with the user, date, time, job number and job name.

How to Get There

On the Library/Object/File Encryption Menu, choose option **2**.

Options

Saved library (SAVLIB)

Specifies the name of the library or group of libraries to restore to the system.

Only those libraries which were saved with the ENCSAVLIB command will be restored.

The possible values are:

***ALL** All libraries which are found on the current loaded media are restored. *ALL is not valid when DEV(*IFS) is specified.

generic*-library-name Specify the generic name of the library. Any libraries on the current loaded media which match the generic name are restored. A generic name is a character string of one or more characters followed by an asterisk (*); for example, ABC*. The asterisk (*) substitutes for any valid characters. A generic name specifies all libraries with names that begin with the generic prefix, for which the user has authority. If an asterisk is not included with the generic (prefix) name, the system assumes it to be the complete library name. A maximum of 300 generic library names can be specified. Generic names are not valid when DEV(*IFS) is specified.

library-name Specify the names of the libraries to restore. The name of the library being restored must be the same as the name that was used when the library was saved. Up to 300 libraries can be specified.

Device (DEV)

Specifies the name of the device used for the restore operation. The device name must already be known on the system by a device description.

This is a required parameter.

diskette-device-name Specify the name of the diskette device used for the restore operation.
optical-device-name Specify the name of the optical device used for the restore operation.
tape-device-name Specify the name of the tape device used for the restore operation.
***IFS** The data is not restored from a device. Specify *IFS to restore the library from an encrypted stream file located on the IFS.

Volume identifier (VOL)

Specifies the volume identifier of the media or the cartridge identifier of tapes in a tape media library device, from which the data is being restored. The volume that contains the beginning of the file to be restored should be placed in the device.

The possible values are:

***MOUNTED** The data is restored from the volumes placed in the device.
volume-identifier Specify the identifier of the volume for the restore operation.

Sequence number (SEQNBR)

Specifies, when tape is used, the sequence number to use as the starting point for the restore operation.

The possible values are:

***SEARCH** The volume in the device is searched for a data file with an identifier that matches the LABEL parameter value; when a match is found, the object is restored. If the last operation on the device specified *LEAVE for the End of tape option prompt (ENDOPT parameter), indicating that the tape is positioned at the location where the last operation ended, the file search starts with the first data file beyond the current tape position. If *LEAVE was not used for the End of tape option prompt (ENDOPT parameter) of the last operation, or if the tape was manually rewound since the operation, the search starts with the first data file on the volume.

file-sequence-number Specify the sequence number of the file to be used for the restore operation. Valid values range from 1 through 16777215.

Label (LABEL)

Specifies the name that identifies the data file on the tape or diskette used for the restore operation. This label must have been specified on the save command.

The possible values are:

***SAVLIB** The file label is the name specified on the Saved library prompt (SAVLIB parameter).
data-file-identifier Specify the data file identifier of the data file used for the restore operation. A maximum of 17 characters can be used.

End of media option (ENDOPT)

Specifies the operation that is automatically done on the tape or optical volume after the restore operation ends.

> **NOTE:** This parameter is valid only if a tape or optical device name is specified on the DEV parameter. For optical devices, *UNLOAD is the only special value supported, *REWIND and *LEAVE will be ignored.

The possible values are:

***REWIND** The tape is automatically rewound, but not unloaded, after the operation has ended.
***LEAVE** The tape does not rewind or unload after the operation ends. It remains at the current position on the tape drive.
***UNLOAD** The tape is automatically rewound and unloaded after the operation ends. Some optical devices will eject the volume after the operation ends.

Option (OPTION)

Specifies how to handle restoring each object.

The possible values are:

***ALL** All the objects in the saved library are restored to the library. Objects in the saved library replace the current versions in the system library. Objects not having a current version are added to the system library. Objects presently in the library, but not on the media, remain in the library.

***NEW** Only the objects in the saved library that do not exist in the current version of the system library are added to the library. Only objects not known to the system library are restored; known objects are not restored. This option restores objects that were deleted after they were saved or that are new to this library. If any saved objects have a version already in the system library, they are not restored, and an informational message is sent for each one, but the restore operation continues.

***OLD** Only the objects in the library having a saved version are restored; that is, the version of each object currently in the library is replaced by the saved version. Only objects known to the library are restored. If any saved objects are no longer part of the online version of the library, they are not added to the library; an informational message is sent for each one, but the restore continues.

***FREE** The saved objects are restored only if they exist in the system library with their space freed. The saved version of each object is restored on the system in its previously freed space. This option restores objects that had their space freed when they were saved. If any saved objects are no longer part of the current version of the library, or if the space is not free for any object, the object is not restored and an informational message is sent for each one. The restore operation continues, and all of the freed objects are restored.

Data base member option (MBROPT)

Specifies, for database files that exist on the system, which members are restored. If *MATCH is used, the member list in the saved file must match, member for member, the current version on the system. All members are restored for files that do not exist, if the file is restored.

The possible values are:

***MATCH** The saved members are restored if the lists of the members where they exist match, member for member, the lists of the current system version. MBROPT (*MATCH) is not valid when *ALL is specified on the Allow object differences prompt (ALWOBJDIF parameter).
***ALL** All members in the saved file are restored.
***NEW** Only new members (members not known to the system) are restored.
***OLD** Only members already known to the system are restored.

Allow object differences (ALWOBJDIF)

Specifies whether certain differences encountered during a restore operation are allowed. The differences include:

- Ownership: the owner of the object on the system is different than the owner of the object from the save operation.

- File creation date: the creation date of the database file on the system does not match the creation date of the file that was saved.
- Member creation date: the creation date of the database file member on the system does not match the creation date of the member that was saved.
- Validation value verification: The validation value created at the time an object was created does not match the validation value created during the restore operation of an object on a system with a QSECURITY level of 40 or higher.
- Authorization list linking: the object is being restored to a system different from the one on which it was saved.

> **NOTE:** To use this parameter, you need *ALLOBJ special authority.

The possible values are:

***NONE** None of the differences described above are allowed on the restore operation. For validation value verification failure cases, the object is restored but ownership is transferred to QDFTOWN and all authorities are revoked. For authorization list cases, the object is restored, but the object is not linked to the authorization list, and public authority is set to *EXCLUDE. For all other cases, a diagnostic message is sent for the object, and the object is not restored.

***ALL** All of the differences listed above are allowed for the restore operation. An informational message is sent, except for validation value verification and authorization list linking cases, and the object is restored. Notes:

- If object differences are found, the final message for the restore operation is an escape message rather than the normal completion message.
- If the media and system owner of the object do not match, the system owner becomes the owner of the object.
- If there is a file level mismatch and *ALL is specified on this parameter and the Data base member option prompt (MBROPT parameter), the existing version of the file is renamed and the saved version of the file is restored. If there is a member level mismatch, the existing version of the member is renamed and the saved version of the member is restored.
- If the system security level is 40, you are restoring a program, you specify *ALL, and the program's validation value is missing or incorrect, the program is restored without authority changes. For programs without a validation value, specifying *ALL also prevents the system from attempting to translate the program again.
- If you are restoring objects to a system different from the one on which they were saved and the objects are secured by an authorization list, specifying *ALL automatically links the objects to the authorization list again. If the authorization list does not exist on the new system, a message that includes the name of the missing list is issued and the public authority is set to *EXCLUDE.

***FILELVL** An attempt will be made to restore existing physical files even though the physical file on the save media may have a different file level id or member level id than the physical file on the system. The physical file data will only be restored for those physical files whose format level identifiers on the save media match the format level identifiers of the corresponding physical file on the system.

Restore to library (RSTLIB)

Specifies whether the library contents are restored to the same library in which they were saved, or to a different library.

The possible values are:

***SAVLIB** The library contents are restored to the same library or libraries in which they were saved.
library-name Specify the name of the library where the saved library contents are restored. If multiple libraries are specified on the Saved library prompt (SAVLIB parameter), a library name cannot be specified on this parameter.

Current release (CURRLS)

Current release.

Restore to ASP device (RSTASPDEV)

Specifies the name of the auxiliary storage pool (ASP) device to which the data is restored.

> NOTE: You can specify either the RSTASPDEV parameter or the RSTASP parameter, but not both.

The possible values are:

***SAVASPDEV** The data is restored to the same ASP from which it was saved.
auxiliary-storage-pool-device-name The data is restored to the specified independent ASP.

Restore to ASP number (RSTASP)

Specifies whether objects are restored to the auxiliary storage pool (ASP) from which they were saved or to the system ASP (ASP number 1) or to a basic user ASP (ASP numbers 2 through 32).

Some objects cannot be restored to user ASPs. More information about object types which can be restored to user ASPs is in the Backup and Recovery book, SC41-5304. If the library exists in, or is being restored to the system ASP, journals, journal receivers, and save files can be restored to basic user ASPs. All other object types will be restored to the ASP of the library.

> NOTE: System or product libraries (libraries that begin with a Q or #) must not be created in or restored to a user ASP. Doing so can cause unpredictable results.

The possible values are:

***SAVASP** The objects are restored to the ASP from which they were saved.
ASP-number (1 - 32) Specifies the ASP number. When the specified ASP is 1, the specified objects are restored to the system ASP, and when the specified ASP is 2 through 32, the objects are restored to the basic user ASP specified.

Use key or password (USEKEYPAS)

Indicate to use either a key from a key store or a password to decrypt the data. The default is *KEY.

The possible values are:

***KEY** Use a key from a key store to decrypt the data.
***PASS** Use a password to decrypt the data.

Key label (KEYLABEL)

Indicate the label of the key to use for decrypting the data.

The possible values are:

key-label Enter the name of a Key Label in the Key Store.
***AUTO** Use the Key Label information stored in the data during encryption.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the data.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Password (PASSWORD) - Help

Specify the password to decrypt the data.

The possible values are:

password-value Enter a password up to 32 characters in length. Mixed case characters can be entered.

> **NOTE:** The password is case-sensitive.

***SRLNBR** The system serial number is used for the password. This value is retrieved from the system value of QSRLNBR.

Encrypted file directory (RSTFDIR) - Help

This is the directory in which the encrypted Stream file(s) are located.

The possible values are:

/CRYPTOTEMP The IFS directory named /CRYPTOTEMP is where the encrypted Stream file(s) are located.
directory-name Specify the IFS directory name that contains the encrypted Stream file (s).

Remove encrypted file(s) after (RMVENCF) - Help

Specifies if the Stream file(s) should be deleted after a successful restore operation.

The possible values are:

***YES** Each Stream file will be deleted after a successful restore operation to a library.
***NO** The Stream files will not be removed.

Libraries to omit (OMITLIB) - Help

Specifies the names of one of more libraries, or the generic names of each group of libraries, to be excluded from the restore operation. Up to 25 library names can be entered.

The possible values are:

***NONE** No libraries are excluded from the restore operation.

generic-library-name Specify the generic name of the libraries to be excluded. A generic name is a character string of one or more characters followed by an asterisk (*); for example, ABC*. The asterisk (*) substitutes for any valid characters. A generic name specifies all libraries with names that begin with the generic prefix, for which the user has authority. If an asterisk is not included with the generic (prefix) name, the system assumes it to be the complete library name.

library-name Specify the name of the library to be excluded from the restore operation.

Objects to omit (OMITOBJ) - Help

Specifies the objects to be excluded from the operation. Up to 25 objects or generic object values can be specified. Specify the object name(s), library(s) and type(s) to be omitted.

The possible values are:

object-name Specify the name of the object to omit.
generic-object-name Specify the generic name of object to omit. A generic name is specified as a character string that contains one or more characters followed by an asterisk (*); for example, ABC*.
***ALL** Omit all objects in the library and type specified.

The possible library values are:

library-name Specify the name of the library where the object exists to omit.
***ALL** The specified objects are excluded from all libraries that are part of the operation.

The possible type values are:

***ALL** All object types are omitted.
character-value Specify the object type of the objects to be excluded from the operation. To see a complete list of object types when prompting this command, position the cursor on the field for this parameter and press F4 (Prompt).

Output information (OUTPUT)

Specifies whether a list with information about the restored objects is created. The information can be printed with the job's spooled output or directed to a database file.

The possible values are:

***NONE** No output listing is created.
***PRINT** The output is printed with the job's spooled output.
***OUTFILE** The output is directed to the database file specified for the File to receive output (OUTFILE) parameter. Note: You must specify a database file name for the File to receive output (OUTFILE) parameter when OUTPUT(*OUTFILE) is specified.

File to receive output (OUTFILE) - Help

Specifies the database file to which the information is directed when *OUTFILE is specified for the Output (OUTPUT) parameter. If the file does not exist, this command creates a database file in the specified library. If a new file is created, the system uses QASAVOBJ in QSYS with the format name QSRSAV as a model.

The possible values are:

file-name Specify the name of the database file to which output from the command is directed. If this file does not exist, it is created in the specified library.

The possible library values are:

library-name Specify the name of the library to be searched.
***LIBL** All libraries in the library list for the current thread are searched until the first match is found.
***CURLIB** The current library for the thread is used to locate the file. If no library is specified as the current library for the job, the QGPL library is used.

Output member options (OUTMBR) - Help

Specifies the name of the database file member to which the output is directed when *OUTFILE is specified for the Output (OUTPUT) parameter.

Member to receive output

The possible values are:

member-name Specify the name of the database file to which output from the command is directed. If this file does not exist, it is created in the specified library.
***FIRST** The first member in the file receives the output. If OUTMBR(*FIRST) is specified and the member does not exist, the system creates a member with the name of the file specified for the File to receive output (OUTFILE) parameter.

Output option (OUTOPT) - Help

Specifies the output option for the file member to which the output is directed when *OUTFILE is specified for the Output (OUTPUT) parameter.

The possible values are:

***REPLACE** The existing records in the specified database file member are replaced by the new records.
***ADD** The new records are added to the existing information in the specified database file member.

Type of output information (INFTYPE) - Help

Specifies the type of information which is printed or directed to the database file.

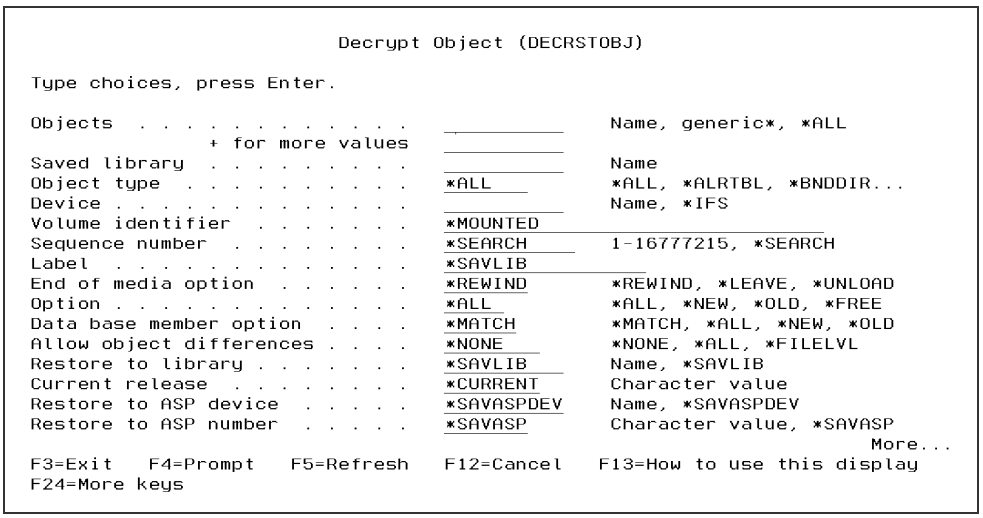
The possible values are:

***OBJ** The list contains an entry for each object requested to be restored.

***MBR** The list contains an entry for each object or, for database files, each member requested to be restored.

Decrypt Object (DECRSTOBJ)

The DECRSTOBJ command allows authorized users to restore and decrypt one or more objects that were encrypted with the ENCSAVOBJ or ENCSAVLIB commands. Objects can be restored from a device (physical or virtual) or the IFS. Either a Symmetric Key or a Password can be specified for the decryption process.



```
                        Decrypt Object (DECRSTOBJ)

Type choices, press Enter.

Objects  . . . . . .  _          Name, generic*, *ALL
           + for more values
Saved library  . . . . . .
Object type  . . . . . .  *ALL     *ALL, *ALRTBL, *BNDDIR...
Device . . . . . .
Volume identifier  . . . . . .  *MOUNTED
Sequence number  . . . . . .  *SEARCH    1-16777215, *SEARCH
Label  . . . . . .  *SAVLIB
End of media option  . . . . . .  *REWIND     *REWIND, *LEAVE, *UNLOAD
Option . . . . . .  *ALL      *ALL, *NEW, *OLD, *FREE
Data base member option  . . . .  *MATCH     *MATCH, *ALL, *NEW, *OLD
Allow object differences . . . .  *NONE      *NONE, *ALL, *FILELVL
Restore to library . . . . . .  *SAVLIB     Name, *SAVLIB
Current release  . . . . . .  *CURRENT     Character value
Restore to ASP device  . . . . .  *SAVASPDEV  Name, *SAVASPDEV
Restore to ASP number  . . . . .  *SAVASP     Character value, *SAVASP
                                                                More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

The types of objects that can be restored by this command are listed on the Object types prompt (OBJTYPE parameter).

The DECRSTOBJ command will restore the object descriptions and their contents.

If logical file access paths were saved (i.e. ACCPTH(*YES) was specified when the objects were saved), the access paths are restored if (1) all based-on physical files are also being restored by the same restore command, (2) the logical file is also being restored by the same restore command, or the logical file already exists on the system (the same file exists, not a re-created version), and (3) MAINT(*IMMED or *DLY) is in effect for the logical file if it still exists on the system.

The user profile of the system default owner (QDFTOWN) becomes the default owner of objects restored in the system whose owner is not known to the system.

If an object is being restored over an existing object on the system, the object auditing value of the existing object is kept. If the object is being restored as new to the system, the object auditing value is restored from the media.

If this command is used to restore a program, the copy of that program that is currently in the system must not be running while the program is being restored. If this occurs, the running program will not be restored.

Make sure the QSYSWRK subsystem is active for support of the DECRSTOBJ command.

Monitoring for Errors

When executing the DECRSTOBJ command within a CL program, you can trap for errors by monitoring for message ids. The message ids for the DECRSTOBJ command are listed below:

CRE0715 - Object(s) were not decrypted. Review JOB LOG.
CRE3773 - &1 Object(s) restored. &2 not restored to &3.

Auditing

If a Symmetric Key is used for the DECRSTOBJ command and “Log decryption usage” is enabled for the Symmetric Key, then an audit log entry will be generated in the Powertech Encryption for IBM i journal file each time the Key is used for decryption. Each audit entry will indicate the Label and Key Store of the Symmetric Key which was used, along with the user, date, time, job number and job name.

How to Get There

On the Library/Object/File Encryption Menu, choose option **4**.

Options

Object (OBJ)

Specifies the names of one or more objects or the generic name of each group of objects to be restored. If the Object type prompt (OBJTYPE parameter) is not specified, all the object types listed in the description of that parameter are restored, provided they are in the specified library and have the specified names. This is a required parameter.

The possible values are:

***ALL** All the objects in the specified library are restored, depending on the value specified on the Object type prompt (OBJTYPE parameter).

generic*-object-name Specify one or more generic names of groups of objects in the specified library to be saved. A generic name is a character string that contains one or more characters followed by an asterisk (*). If an * is not specified with the name, the system assumes that the name is a complete object name.

object-name Specify one or more names of specific objects to save. Both generic names and specific names can be specified
in the same command. A maximum of 300 object names can be specified.

Saved library (SAVLIB)

Specifies the name of the library that contained the saved objects. If the Restore to library prompt (RSTLIB parameter) is not specified, this is also the name of the library to which the objects are restored. This is a required parameter.

The possible values are:

library-name Specify the name of the library that contained the saved objects.

Object type (OBJTYPE)

Specifies the type of system objects to restore. For a complete list of object types that can be restored, move the cursor to the field for the Object type prompt (OBJTYPE parameter) and press the **F4** key.

The possible values are:

***ALL** All object types that are specified by name and are in the specified library are restored. If *ALL is also specified on the Objects prompt (OBJ parameter), then all the objects in the library that are of the types that can be restored are restored.

object-type Specify the value for the types of objects that are saved, such as command (*CMD), file (*FILE) or program (*PGM).

Device (DEV)

Specifies the name of the device used for the restore operation. The device name must already be known on the system by a device description. This is a required parameter.

The possible values are:

diskette-device-name Specify the name of the diskette device used for the restore operation.
optical-device-name Specify the name of the optical device used for the restore operation.
tape-device-name Specify the name of the tape device used for the restore operation.
***IFS** The data is not restored from a device. Specify *IFS to restore the library from an encrypted stream file located on the IFS.

Volume identifier(VOL)

Specifies the volume identifier of the media or the cartridge identifier of tapes in a tape media library device, from which the data is being restored. The volume that contains the beginning of the file to be restored should be placed in the device.

The possible values are:

***MOUNTED** The data is restored from the volumes placed in the device.
volume-identifier Specify the identifier of the volume for the restore operation.

Sequence number (SEQNBR)

Specifies, when tape is used, the sequence number to use as the starting point for the restore operation.

The possible values are:

***SEARCH** The volume in the device is searched for a data file with an identifier that matches the LABEL parameter value; when a match is found, the object is restored. If the last operation on the device specified *LEAVE for the End of tape option prompt (ENDOPT parameter), indicating that the tape is positioned at the location where the last operation ended, the file search starts with the first data file beyond the current tape position. If *LEAVE was not used for the End of tape option prompt (ENDOPT parameter) of the last operation, or if the tape was manually rewound since the operation, the search starts with the first data file on the volume.

file-sequence-number Specify the sequence number of the file to be used for the restore operation. Valid values range from 1 through 16777215.

Label (LABEL)

Specifies the name that identifies the data file on the tape or diskette used for the restore operation. This label must have been specified on the save command.

The possible values are:

***SAVLIB** The file label is the name specified on the Saved library prompt (SAVLIB parameter).
data-file-identifier Specify the data file identifier of the data file used for the restore operation. A maximum of 17 characters can be used.

End of media option (ENDOPT)

Specifies the operation that is automatically done on the tape or optical volume after the restore operation ends.

> NOTE: This parameter is valid only if a tape or optical device name is specified on the DEV parameter. For optical devices, *UNLOAD is the only special value supported, *REWIND and *LEAVE will be ignored.

The possible values are:

***REWIND** The tape is automatically rewound, but not unloaded, after the operation has ended.
***LEAVE** The tape does not rewind or unload after the operation ends. It remains at the current position on the tape drive.
***UNLOAD** The tape is automatically rewound and unloaded after the operation ends. Some optical devices will eject the volume after the operation ends.

Option (OPTION)

Specifies how to handle restoring each object.

The possible values are:

***ALL** All the objects in the saved library are restored to the library. Objects in the saved library replace the current versions in the system library. Objects not having a current version are added to the system library. Objects presently in the library, but not on the media, remain in the library.

***NEW** Only the objects in the saved library that do not exist in the current version of the system library are added to the library. Only objects not known to the system library are restored; known objects are not restored. This option restores objects that were deleted after they were saved or that are new to this library. If any saved objects have a version already in the system library, they are not restored, and an informational message is sent for each one, but the restore operation continues.

***OLD** Only the objects in the library having a saved version are restored; that is, the version of each object currently in the library is replaced by the saved version. Only objects known to the library are restored. If any saved objects are no longer part of the online version of the library, they are not added to the library; an informational message is sent for each one, but the restore continues.

***FREE** The saved objects are restored only if they exist in the system library with their space freed. The saved version of each object is restored on the system in its previously freed space. This option restores objects that had their space freed when they were saved. If any saved objects are no longer part of the current version of the library, or if the space is not free for any object, the object is not restored and an informational message is sent for each one. The restore operation continues, and all of the freed objects are restored.

Data base member option (MBROPT)

Specifies, for database files that exist on the system, which members are restored. If *MATCH is used, the member list in the saved file must match, member for member, the current version on the system. All members are restored for files that do not exist, if the file is restored.

The possible values are:

***MATCH** The saved members are restored if the lists of the members where they exist match, member for member, the lists of the current system version. MBROPT (*MATCH) is not valid when *ALL is specified on the Allow object differences prompt (ALWOBJDIF parameter).
***ALL** All members in the saved file are restored.
***NEW** Only new members (members not known to the system) are restored.
***OLD** Only members already known to the system are restored.

Allow object differences (ALWOBJDIF)

Specifies whether certain differences encountered during a restore operation are allowed. The differences include:

- Ownership: the owner of the object on the system is different than the owner of the object from the save operation.
- File creation date: the creation date of the database file on the system does not match the creation date of the file that was saved.
- Member creation date: the creation date of the database file member on the system does not match the creation date of the member that was saved.
- Validation value verification: The validation value created at the time an object was created does not match the validation value created during the restore operation of an object on a system with a QSECURITY level of 40 or higher.
- Authorization list linking: the object is being restored to a system different from the one on which it was saved.

> **NOTE:** To use this parameter, you need *ALLOBJ special authority.

The possible values are:

***NONE** None of the differences described above are allowed on the restore operation. For validation value verification failure cases, the object is restored but ownership is transferred to QDFTOWN and all authorities are revoked. For authorization list cases, the object is restored, but the object is not linked to the authorization list, and public authority is set to *EXCLUDE. For all other cases, a diagnostic message is sent for the object, and the object is not restored.

***ALL** All of the differences listed above are allowed for the restore operation. An informational message is sent, except for validation value verification and authorization list linking cases, and the object is restored. Notes:

- If object differences are found, the final message for the restore operation is an escape message rather than the normal completion message.
- If the media and system owner of the object do not match, the system owner becomes the owner of the object.
- If there is a file level mismatch and *ALL is specified on this parameter and the Data base member option prompt (MBROPT parameter), the existing version of the file is renamed and the saved version of the file is restored. If there is a member level mismatch, the existing version of the member is renamed and the saved version of the member is restored.
- If the system security level is 40, you are restoring a program, you specify *ALL, and the program's validation value is missing or incorrect, the program is restored without authority changes. For programs without a validation value, specifying *ALL also prevents the system from attempting to translate the program again.
- If you are restoring objects to a system different from the one on which they were saved and the objects are secured by an authorization list, specifying *ALL automatically links the objects to the authorization list again. If the authorization list does not exist on the new system, a message that includes the name of the missing list is issued and the public authority is set to *EXCLUDE.

***FILELVL** An attempt will be made to restore existing physical files even though the physical file on the save media may have a different file level id or member level id than the physical file on the system. The physical file data will only be restored for those physical files whose format level identifiers on the save media match the format level identifiers of the corresponding physical file on the system.

Restore to library (RSTLIB)

Specifies whether the object contents are restored to the same library in which they were saved, or to a different library.

The possible values are:

***SAVLIB** The object contents are restored to the same library in which they were saved.
library-name Specify the name of the library where the saved object contents are restored.
Current release (CURRLS) Current release.

Restore to ASP device (RSTASPDEV)

Specifies the name of the auxiliary storage pool (ASP) device to which the data is restored.

> NOTE: You can specify either the RSTASPDEV parameter or the RSTASP parameter, but not both.

The possible values are:

***SAVASPDEV** The data is restored to the same ASP from which it was saved.
auxiliary-storage-pool-device-name The data is restored to the specified independent ASP.

Restore to ASP number (RSTASP)

Specifies whether objects are restored to the auxiliary storage pool (ASP) from which they were saved or to the system ASP (ASP number 1) or to a basic user ASP (ASP numbers 2 through 32).

Some objects cannot be restored to user ASPs. More information about object types which can be restored to user ASPs is in the Backup and Recovery book, SC41-5304. If the library exists in, or is being restored to the system ASP, journals, journal receivers, and save files can be restored to basic user ASPs. All other object types will be restored to the ASP of the library.

> NOTE: System or product libraries (libraries that begin with a Q or #) must not be created in or restored to a user ASP. Doing so can cause unpredictable results.

The possible values are:

***SAVASP** The objects are restored to the ASP from which they were saved.
ASP-number (1 - 32) Specifies the ASP number. When the specified ASP is 1, the specified objects are restored to the system ASP, and when the specified ASP is 2 through 32, the objects are restored to the basic user ASP specified.

Use key or password (USEKEYPAS)

Indicate to use either a key from a key store or a password to decrypt the data. The default is *KEY.

The possible values are:

***KEY** Use a key from a key store to decrypt the data.
***PASS** Use a password to decrypt the data.

Key label (KEYLABEL)

Indicate the label of the key to use for decrypting the data.

The possible values are:

key-label Enter the name of a Key Label in the Key Store.
***AUTO** Use the Key Label information stored in the data during encryption.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the data.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Password (PASSWORD) - Help

Specify the password to decrypt the data.

The possible values are:

password-value Enter a password up to 32 characters in length. Mixed case characters can be entered.

> NOTE: The password is case-sensitive.

***SRLNBR** The system serial number is used for the password. This value is retrieved from the system value of QSRLNBR.

Encrypted file name (ENCFNAM) - Help

Specifies the name of the encrypted Stream file that was used when the ENCSAVOBJ command was issued.

> NOTE: This value was specified for the ENCFNAM parameter on the ENCSAVOBJ command.

The possible values are:

ENCSAVOBJ The name of ENCSAVOBJ was used for the encrypted Stream file when the ENCSAVOBJ command was issued.
file-name Specify the name of the Stream file that was used when the ENCSAVOBJ command was issued.

Restore file directory (RSTFDIR) - Help

This is the directory in which the encrypted Stream file is located.

The possible values are:

/CRYPTOTEMP The IFS directory named /CRYPTOTEMP contains the encrypted Stream file.
directory-name Specify the IFS directory name that contains the encrypted Stream file.

Remove encrypted file after (RMVENCF) - Help

Specifies if the encrypted Stream file should be removed after it is decrypted into the staging Save file.

The possible values are:

***YES** The Stream file will be removed after it is decrypted into the staging Save file. This option will minimize the disk usage required for this command.
***NO** The Stream file will not be removed.

Objects to omit (OMITOBJ) - Help

Specifies the objects to be excluded from the operation. Up to 25 objects or generic object values can be specified. Specify the object name(s), library(s) and type(s) to be omitted.

The possible values are:

object-name Specify the name of the object to omit.

generic-object-name Specify the generic name of object to omit. A generic name is specified as a character string that contains one or more characters followed by an asterisk (*); for example, ABC*.

***ALL** Omit all objects in the library and type specified.

The possible library values are:

library-name Specify the name of the library where the object exists to omit.
***ALL** The specified objects are excluded from all libraries that are part of the operation.

The possible type values are:

***ALL** All object types are omitted.

character-value Specify the object type of the objects to be excluded from the operation. To see a complete list of object types when prompting this command, position the cursor on the field for this parameter and press F4 (Prompt).

Output information (OUTPUT) Specifies whether a list with information about the restored objects is created. The information can be printed with the job's spooled output or directed to a database file.

The possible values are:

***NONE** No output listing is created.
***PRINT** The output is printed with the job's spooled output.
***OUTFILE** The output is directed to the database file specified for the File to receive output (OUTFILE) parameter. Note: You must specify a database file name for the File to receive output (OUTFILE) parameter when OUTPUT(*OUTFILE) is specified.

File to receive output (OUTFILE) - Help

Specifies the database file to which the information is directed when *OUTFILE is specified for the Output (OUTPUT) parameter. If the file does not exist, this command creates a

database file in the specified library. If a new file is created, the system uses QASAVOBJ in QSYS with the format name QSRSAV as a model.

The possible values are:

file-name Specify the name of the database file to which output from the command is directed. If this file does not exist, it is created in the specified library.

The possible library values are:

library-name

Specify the name of the library to be searched.

***LIBL** All libraries in the library list for the current thread are searched until the first match is found.
***CURLIB** The current library for the thread is used to locate the file. If no library is specified as the current library for the job, the QGPL library is used.

Output member options (OUTMBR) - Help

Specifies the name of the database file member to which the output is directed when *OUTFILE is specified for the Output (OUTPUT) parameter.

Member to receive output

The possible values are:

member-name Specify the name of the database file to which output from the command is directed. If this file does not exist, it is created in the specified library.

***FIRST** The first member in the file receives the output. If OUTMBR(*FIRST) is specified and the member does not exist, the system creates a member with the name of the file specified for the File to receive output (OUTFILE) parameter.

Output option (OUTOPT) - Help

Specifies the output option for the file member to which the output is directed when *OUTFILE is specified for the Output (OUTPUT) parameter.

The possible values are:

***REPLACE** The existing records in the specified database file member are replaced by the new records.

***ADD** The new records are added to the existing information in the specified database file member.

Type of output information (INFTYPE) - Help

Specifies the type of information which is printed or directed to the database file.

The possible values are:

***OBJ** The list contains an entry for each object requested to be restored.
***MBR** The list contains an entry for each object or, for database files, each member requested to be restored.

Delete Alert (DLTCCALR)

The DLTCCALR command allows an authorized user to delete a Security Alert entry.

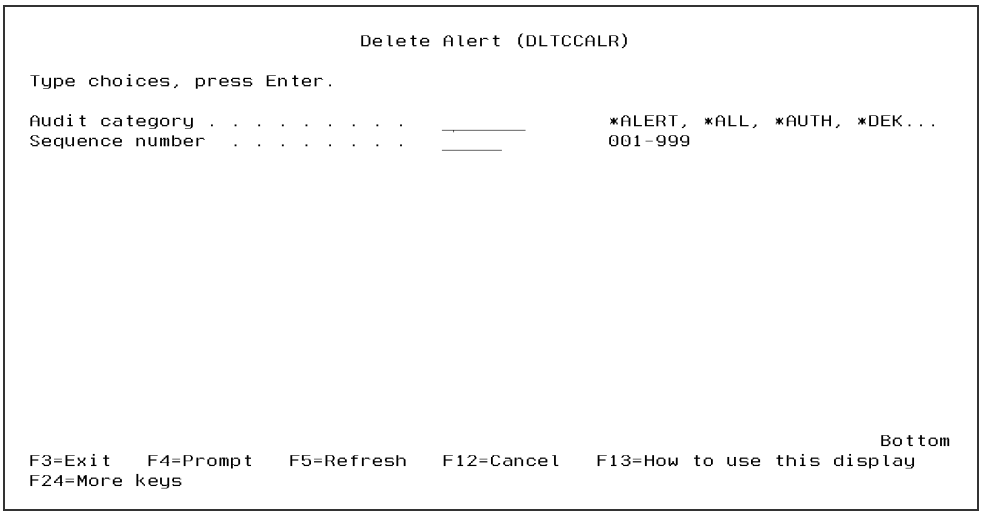
> **NOTE:** Any maintenance to the Security Alerts is logged into an audit file.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key policy and alerts” authority setting

Do the following steps to delete an Alert:

1. Prompt (**F4**) the command **CRYPTO/DLTCCALR**.
2. Type in the audit category and sequence number.
3. Press Enter to delete the entry.



Delete Symmetric Key (DLTSYMKEY)

The DLTSYMKEY command allows authorized users to delete a Data Encryption Key (Symmetric Key) from a Key Store.

NOTE: By default, the Key Policy does not allow the deletion of keys.

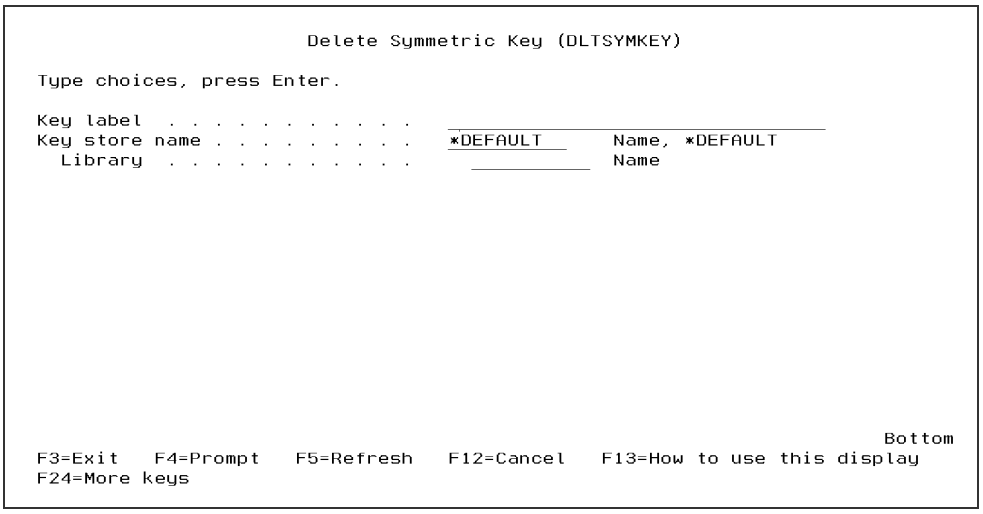
WARNING: Do not delete keys that may be needed to decrypt existing data.

The following users can use the DLTSYMKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain DEKs” authority setting

The user must have *CHANGE authority to the Validation List (*VLDL) object containing the Key Store(s) from which the Key will be deleted, and *USE authority to the library that contains the Key Store.

Before a Key is deleted, the Validation List (*VLDL) object, which contains the Key Store, is backed up into a Save File object (sequentially named) within the Powertech Encryption for IBM i library.



How to Get There

From the Symmetric Encryption Key Menu, choose option **16**.

Options

Key label (KEYLABEL)

Indicate the unique name (label) of the Symmetric Key to delete.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to delete.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

Display a Field Encryption Pending Key (DSPPNDKEY)

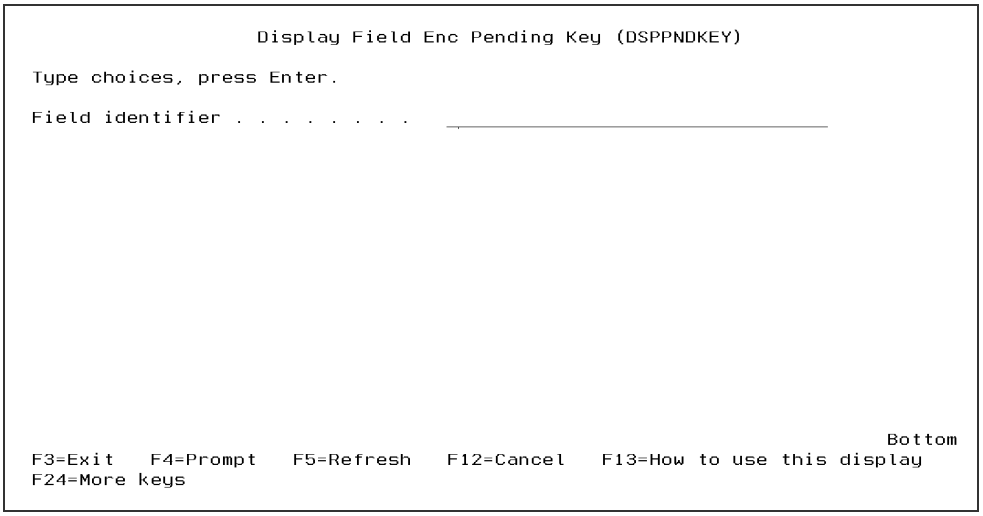
The Display a Field Encryption Pending Key (DSPPNDKEY) command allows authorized users to display a Pending key in a Field entry that can be used in the next key rotation process.

This command can be used for *ACTIVE field entries that use Field Procedures.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.



How to Get There

In the File Field Encryption Menu, choose option **9**.

Options

Field identifier (FLDID)

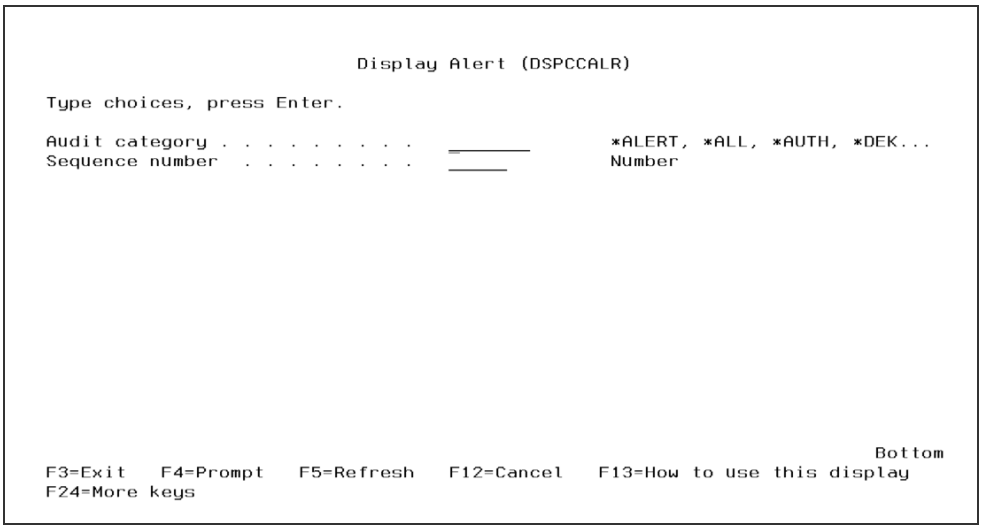
Indicate the unique name of the field entry to display the Pending key.

Display Alert (DSPCCALR)

The DSPCCALR command allows an authorized user to display the settings for a Security Alert entry.

Do the following steps to view the settings for an Alert:

1. Prompt (**F4**) the command **CRYPTO/DSPCCALR**.
2. Type in the audit category and sequence number, then press Enter.
3. The settings for the Alert will be displayed, along with the user and time in which the Alert was added or last changed.
4. Press **F1** on any parameter for complete online help text.



Options

Audit category (AUDITCAT)

Indicates the Audit category to monitor.

The possible values are:

***ALL** All audit categories are monitored.
***ALERT** Any maintenance to the Security Alert settings will trigger an alert, which includes the following audit types:

- 35-Alert added.
- 36-Alert changed.
- 37-Alert deleted.

***AUTH** Any authority errors that are encountered in Powertech Encryption will trigger an alert, which includes the following audit types:

- 50-Authority error

***DEK** Any maintenance to the Data Encryption Keys will trigger an alert, which includes the following audit types:

- 08-Key Store created.
- 09-Key Store translated.
- 10-Symmetric Key created.
- 11-Symmetric Key changed.
- 12-Symmetric Key copied.
- 13-Symmetric Key deleted.
- 21-Symmetric Key exported.

***FLDREG** Any maintenance to the Field Encryption Registry will trigger an alert, which includes the following audit types:

- 14-Entry added
- 15-Encryption Key changed
- 16-Entry removed
- 17-Entry activated
- 18-Entry changed
- 19-Entry deactivated
- 22-Unable to Activate Entry
- 23-Unable to Deactivate Entry
- 24-Entry copied
- 25-SQL Triggers added to file
- 26-SQL Triggers removed from file
- 27-Field keys translated

***IFSREG** Any maintenance to the IFS Encryption Registry will trigger an alert, which includes the following audit types:

- 60-Entry added
- 61-Encryption Key changed
- 62-Entry removed
- 63-Entry activated
- 64-Entry changed
- 65-Entry deactivated
- 66-Unable to Activate Entry
- 67-Unable to Deactivate Entry
- 68-Encryption Failed
- 69-Decryption Failed
- 70-IFS Monitor Issue

***KEYOFR** Any maintenance to the Key Officer settings will trigger an alert, which includes the following audit types:

- 02-Key Officer added.
- 03-Key Officer changed.
- 04-Key Officer removed.

***KEYPCY** Any maintenance to the Key Policy settings will trigger an alert, which includes the following audit types:

- 01-Key Policy setting(s) changed.

***MEK** Any maintenance to the Master Encryption Keys will trigger an alert, which includes the following audit types:

- 05-Master Key passphrase part loaded.
- 06-Master Key was Set.
- 07-Master Key cleared.

Sequence number (SEQNBR)

Indicates the sequence number within the Audit Category. Valid sequence numbers range from 001 to 999.

Action (ACTION)

Indicates the Action to perform for the alert.

The possible values are:

***EMAIL** Send email to one or more recipients. Uses the SNDDST command.
***MSGQBRK** Send break messages to the message queue specified. Uses the SNDBRKMSG command.
***MSGQINF** Send information messages to the message queue specified. Uses the SNDMSG command.
***PTGLOG** Send log messages to the Protegrity Defiance Enterprise Security Administrator (ESA).
***QAUDJRN** Write journal entries into the QAUDJRN journal file.
***QHST** Send messages to the QHST log message queue. Uses the SNDMSG command.
***QSYSOPR** Send messages to the QSYSOPR message queue. Uses the SNDMSG command.
***SYSLOG** Send messages to an external log server using SYSLOG protocol.
***USER** Send messages to the user specified. Uses the SNDMSG command.

To user profile (TOUSER)

Valid for *USER action type. Indicates the user profile name to send the alert message to.

To message queue (TOMSGQ)

Valid for *MSGQINF and *MSGQBRK action types. Indicates the message queue name and library to send the alert message to.

The possible library values are:

***LIBL** The library list is used to find the message queue.
message-queue-library Specify the library of the message queue.

To email address (TOEMAIL)

Valid for *EMAIL action type. Indicates the email address(s) to send the alert to. Multiple email addresses can be specified by separating them with a comma. Example: john@abc.com,mike@abc.com,jim@abc.com

Log host (LOGHOST)

Valid for *SYSLOG and *PTGLOG action types. The host name or IP address of the log server.

Destination port (LOGSRCPORT)

Valid for *SYSLOG action type. The local port to use when connecting to the log server.

> **NOTE:** When the local port is set to 0, the system will search for an available local port to use.

Destination port (LOGDSTPORT)

Valid for *SYSLOG and *PTGLOG action types. The port for the log server. The default port for syslog servers is 514.

Log facility (LGFACILITY)

Valid for *SYSLOG action type. The log facility. The facility value is a way of determining which process of the machine created the message.

Log severity (LGSEVERITY)

Valid for *SYSLOG action type. The log severity. The severity value is a way of determining the importance of the message.

Client Application (LOGCLNTAPP)

Valid for *PTGLOG action type. The Client Application ID to use. The Client Application is created in the IBM Digital Certificate Manager. This client application links the Client Certificate to use with this application.

Last modified by user (MODUSER)

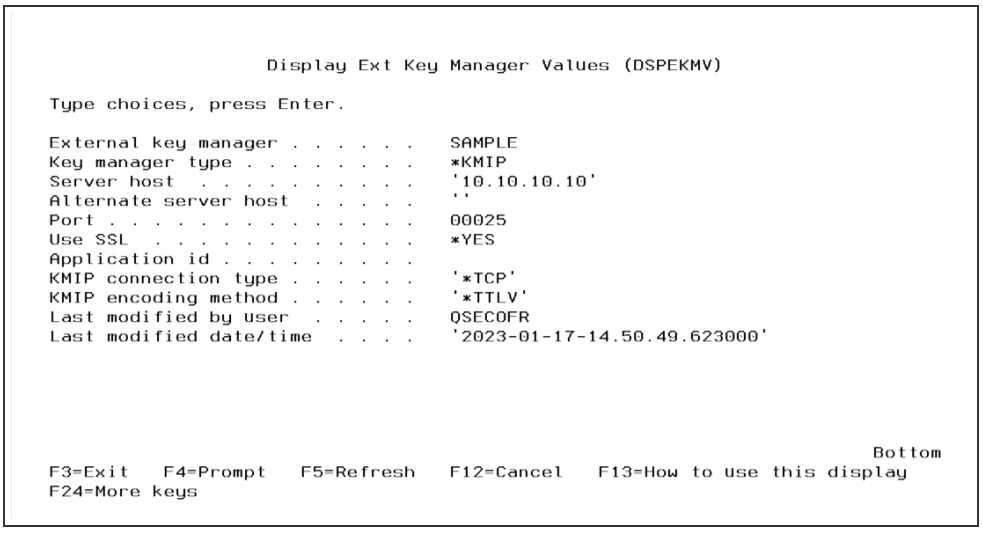
Indicates the user profile that created or last modified the Security Alert.

Last modified date/time (MODDATETIM)

Indicates the date and time in which the Security Alert settings were last modified.

Display External Key Manager (DSPEKM)

The DSPEKM command allows an authorized user to display the properties for an External Key Manager.



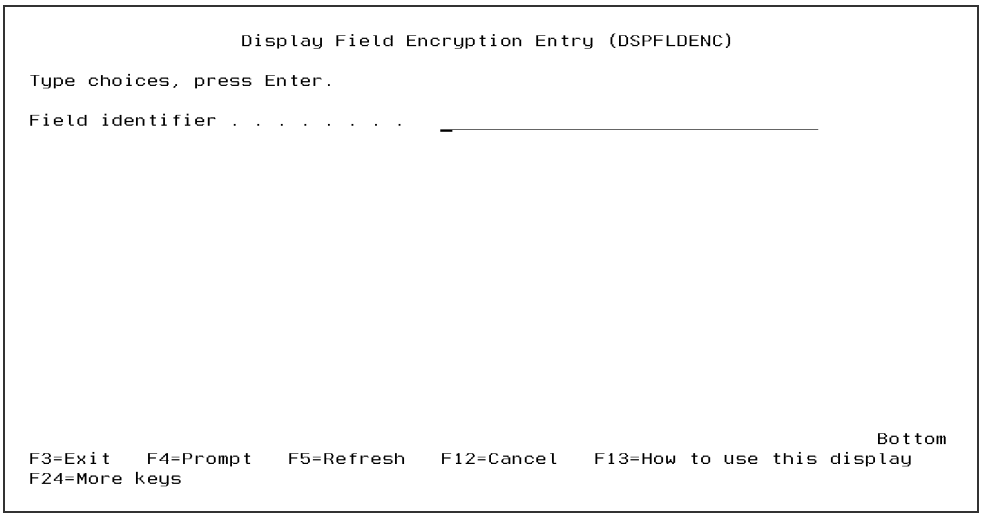
How to Get There

From the External Key Manager Menu, choose option **4**. Or, submit the command **DSPEKM**.

Display Field Encryption Entry (DSPFLDENC)

The DSPFLDENC command allows authorized users to display a field entry's settings within a Field Encryption Registry.

This command requires that you have *USE authority to the CRVL002 Validation List (*VLDL) object, which contains the Field Encryption Registry.



How to Get There

Do the following steps to display a field entry's settings within the Encryption Registry:

1. Prompt (**F4**) the command **CRYPTO/DSPFLDENC**.
2. Enter the Field identifier to display, and then press Enter.
3. The current field entry settings (parameter values) will be displayed, along with the user and timestamp recorded when the field entry was added or last changed.
4. Press **F1** on any parameter for complete online help text.
5. For an explanation of the parameters, see Add Field Encryption Entry (ADDFLDENC).

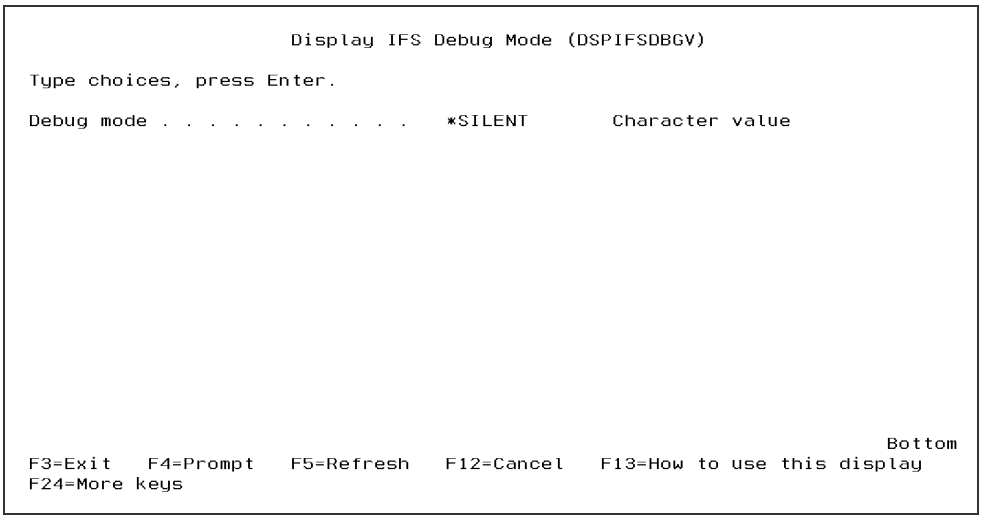
Options

Field identifier (FLDID)

Specify the unique name of the entry to display.

Display IFS Debug Mode (DSPIFSDBG)

The Display IFS Debug mode (DSPIFSDBGV) command allows authorized users to view the Debug Mode.



How to Get There

From the IFS Utility Menu, choose option **5**, Display IFS Debug Mode. Or, prompt (**F4**) the command **CRYPTO/DSPIFSDBG**.

Options

Debug Mode (DEBUG)

Indicates the Debug Mode.

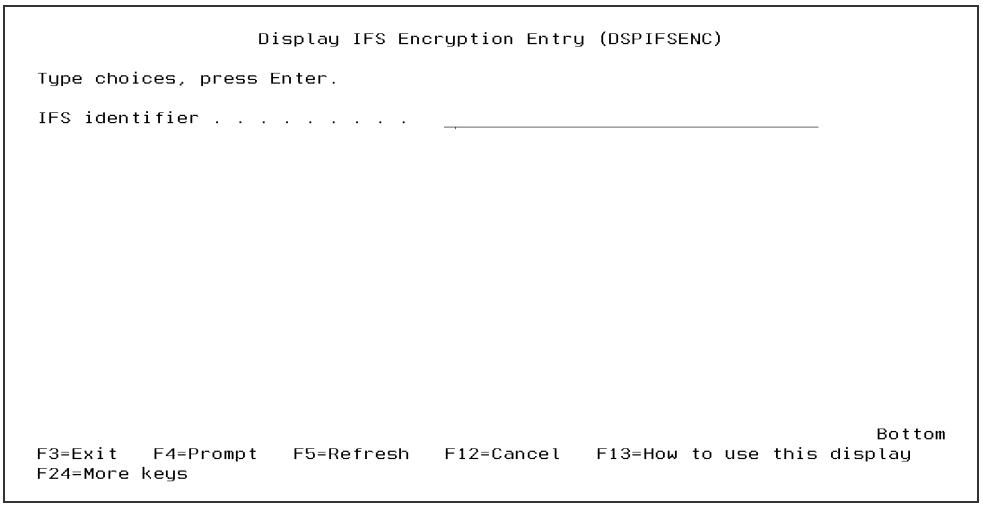
The possible values are:

***SILENT** No information will be written to the CRPFIFSLOG File.
***NORMAL** Encryption and Decryption information will be written to the CRPFIFSLOG File.
***DEBUG** Encryption and Decryption information as well as other Debug information will be written to the CRPFIFSLOG File.

Display IFS Encryption Entry (DSPIFSENC)

The Display IFS Encryption Entry (DSPIFSENC) command allows authorized users to view an existing entry in the IFS Encryption Registry.

This command requires the user to have *USE authority to the CRVL003 Validation List (*VLDL) object which contains the IFS Encryption Registry.



How to Get There

From the IFS Encryption Menu, choose option **4**. Or, prompt (**F4**) the command of **CRYPTO/DSPIFSENC**.

Options

IFS identifier (IFSID)

Specify the unique name of the entry to display.

Display Journal (DSPJRN)

The Display Journal (DSPJRN) command allows you to convert journal entries (contained in one or more receivers) into a form suitable for external representation. Output of the command can be displayed or printed with the job's spooled printer output or directed to a database output file. If the database output file exists, records may either replace or be added to the current data in the indicated file member. The system creates the specified database file and member if they do not exist. Database files created by the system have a standard format. A warning message is sent and the records are truncated if any of the entries are longer than the specified maximum record length of the output files.

The contents of selected entries in the journal receivers may be converted for output. It is also possible to selectively limit the entries that are displayed. If no journal entries satisfy the selection or limitation criteria, an escape message is sent indicating that fact.

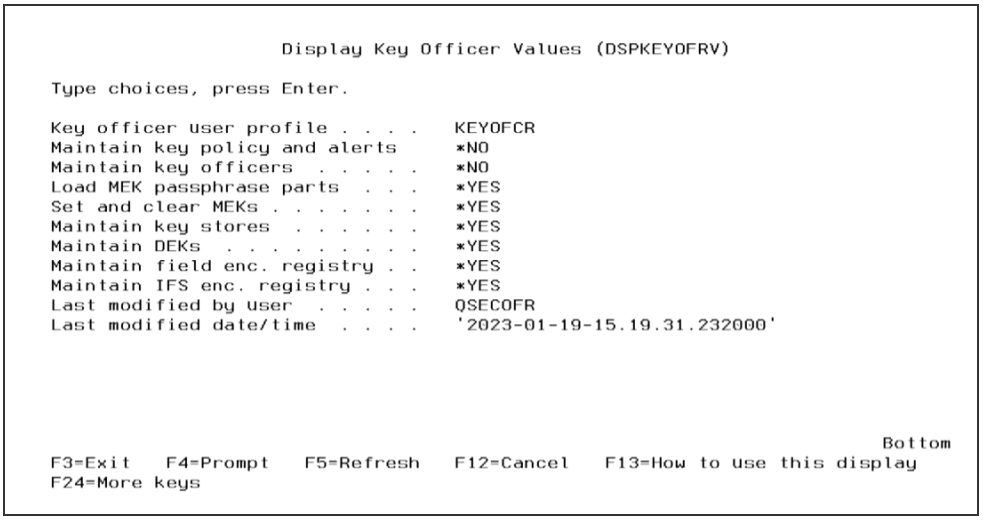
Gaps may exist in the sequence numbers of the entries converted. These occur because some of the journal entries represent internal system information. These internal entries can be shown by specifying INCHIDENT(*YES).

It is possible to show journal entries whose journal sequence numbers are reset in the chain of receivers being specified.

For complete information, see Display Journal (DSPJRN) on the IBM Documentation website.

Display Key Officer (DSPKEYOFR)

The DSPKEYOFR command allows an authorized user to display a Key Officer's authority settings.



How to Get There

From the Key Policy and Security Menu, choose option **13**, Display Key Officer. Or, prompt (**F4**) the command **CRYPTO/DSPKEYOFR**.

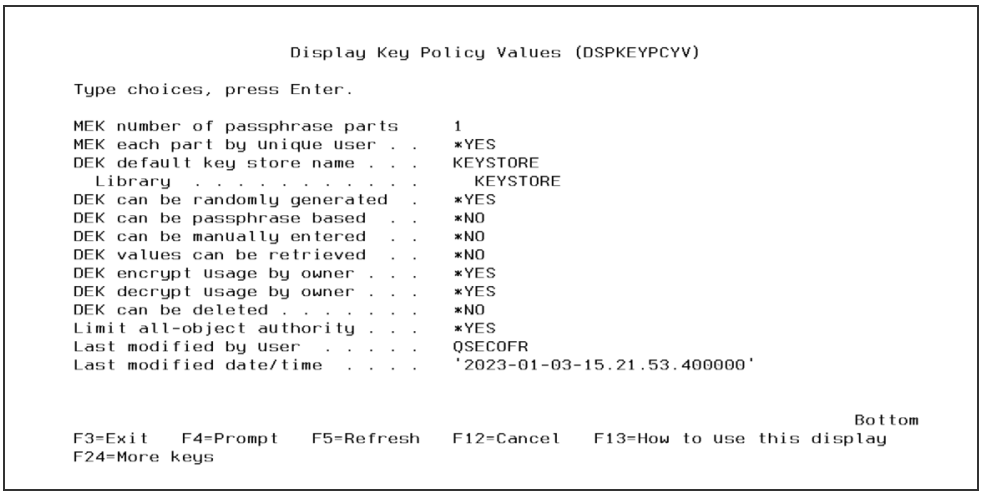
Options

Key officer user profile (USRPRF)

Specify the Key Officer's user profile on the IBM i.

Display Key Policy (DSPKEYPCY)

The DSPKEYPCY command allows authorized users to view the policy settings for the Symmetric Key environment. The current policy settings (parameter values) are displayed, along with the user id and time in which the settings were last changed.



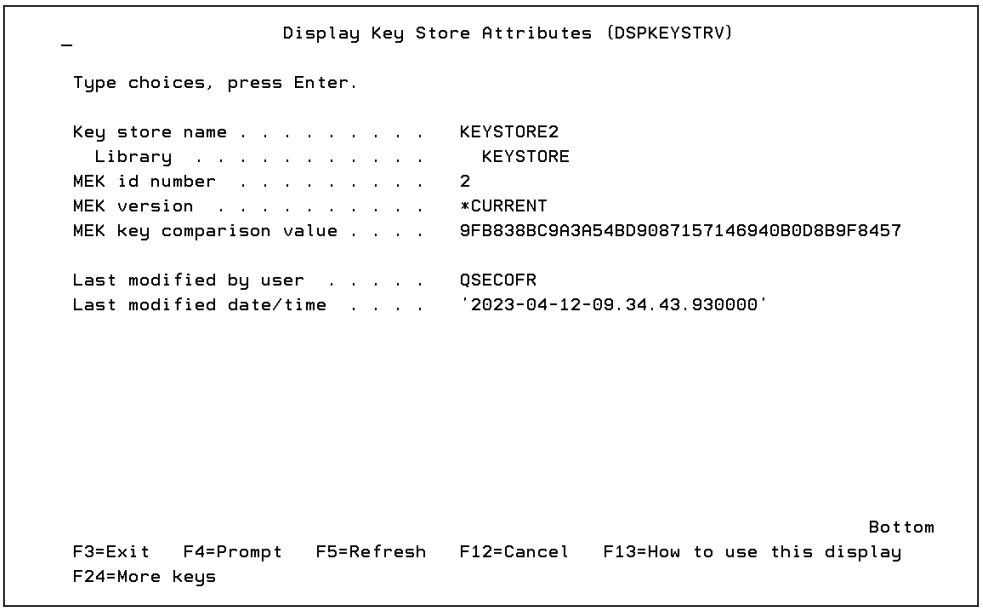
How to Get There

From the Key Policy and Security Menu, choose option **2**, Display Key Policy. Or, enter the command **CRYPTO/DSPKEYPCY**.

For a description of these values, see Change Key Policy (CHGKEYPCY).

Display Key Store Attributes (DSPKEYSTR)

The Display Key Store Attributes (DSPKEYSTR) command allows authorized users to display the attributes for a Key Store. This is primarily useful for viewing the Master Encryption Key (MEK) id number and version in which the Key Store entries are encrypted under.



How to Get There

From the Symmetric Encryption Key Menu, choose option **2**.

Options

Key store name (KEYSTR)

Indicate the Key Store name and Library to display attributes for.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

> **NOTE:**
> A Key comparison value (KEYCV) is generated by Powertech Encryption for IBM i for each Master Key. The KEYCV is a different value (and has a different purpose) than the actual value of the Master Key. This value is a unique identifier for comparing and verifying a key.

IMPORTANT: This value was formerly known as a Key verification value (KEYVV) and has been updated in Powertech Encryption 4.0. It is therefore not suitable for comparison across earlier versions. Please contact Technical Support for assistance in comparing values across versions.

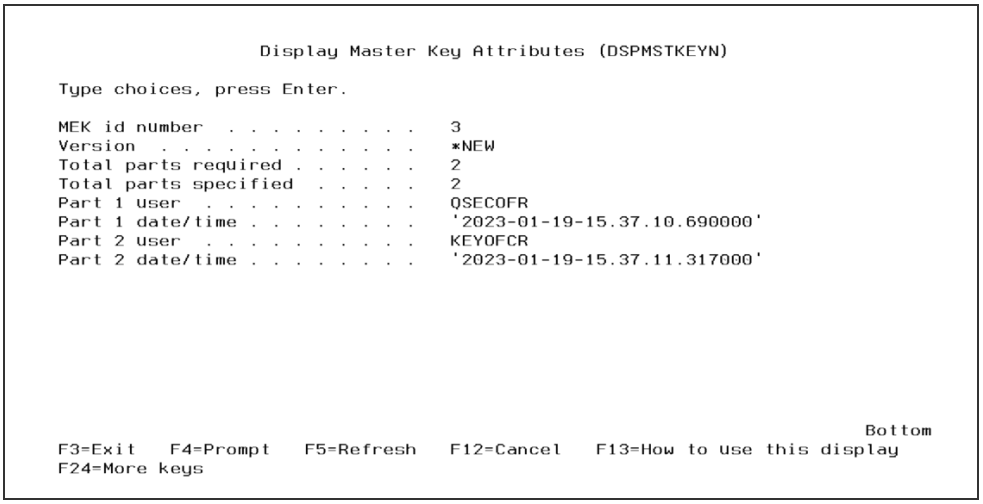
Display Master Key Attributes (DSPMSTKEY)

The DSPMSTKEY command allows authorized users to display the attributes for a Master Key.

NOTE: The actual key value of a Master Encryption Key cannot be displayed.

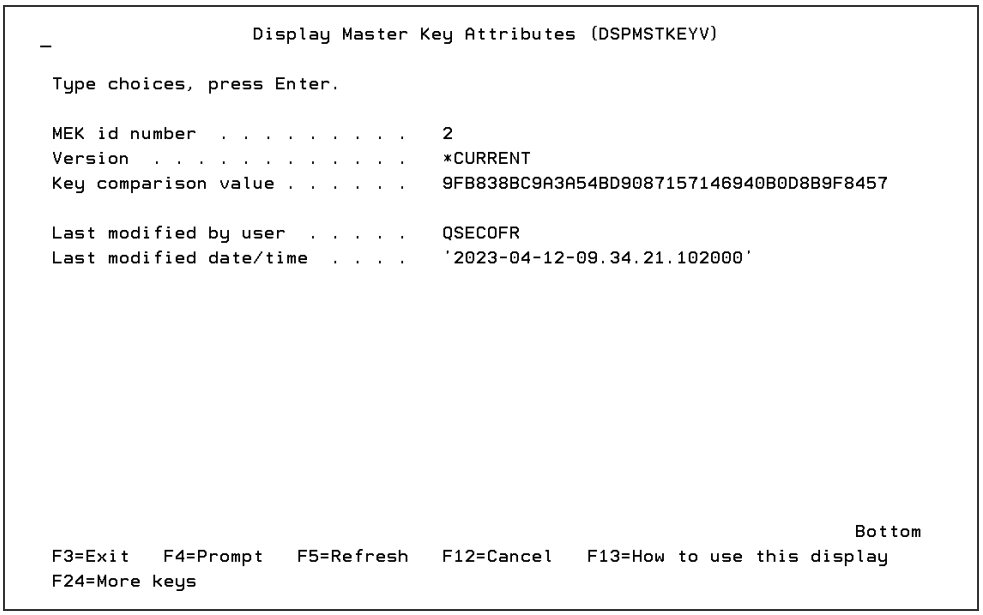
For a *NEW master key, the attributes displayed will be the total passphrase parts specified for a Master Key, along with the user profiles (and timestamps) that specified those parts.

*Example of a *NEW Master Key*



For a *CURRENT or *OLD Master Key, the attributes displayed will be the Key comparison value, along with the user profile (and timestamp) which Set the Master Key with the SETMSTKEY command.

*Example of a *CURRENT Master Key*



How to Get There

From the Master Encryption Key Menu, choose option **3**, Display Master Key Attributes. Or, prompt (**F4**) the command **CRYPTO/DSPMSTKEY**.

NOTE:
A Key comparison value (KEYCV) is generated by Powertech Encryption for IBM i for each Master Key. The KEYCV is a different value (and has a different purpose) than the actual value of the Master Key. The Master Key's KEYCV value will be stored with each Key Store created using that Master Key.

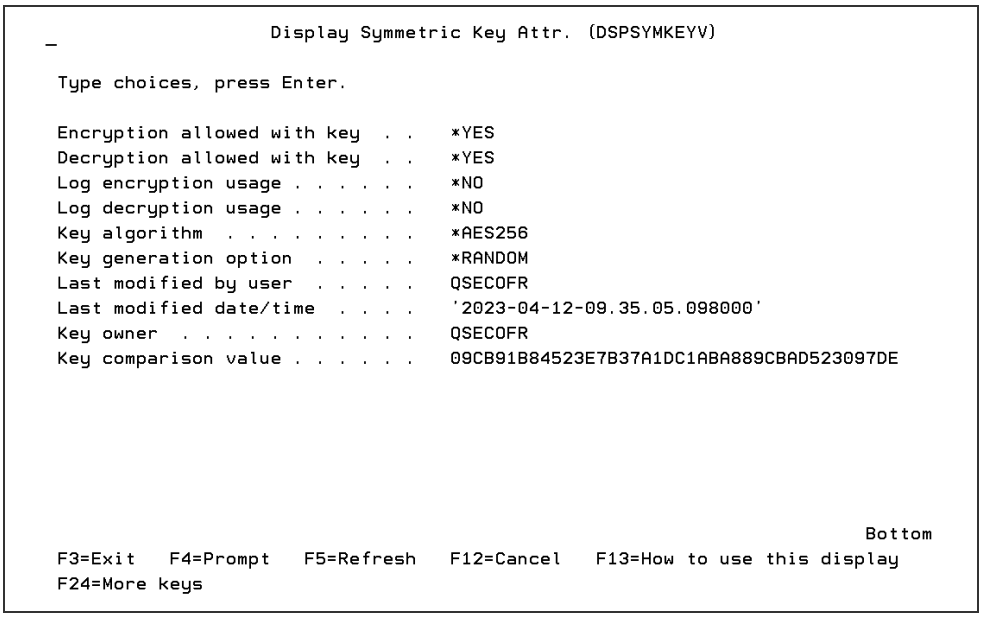
When a Key Store is accessed by a user or application, Powertech Encryption for IBM i will compare the KEYCV values between the Key Store and its corresponding Master Key. If the KEYCV values match, then the Master Key is determined as valid for the Key Store.

IMPORTANT: This value was formerly known as a Key verification value (KEYVV) and has been updated in Powertech Encryption 4.0. It is therefore not suitable for comparison across earlier versions. Please contact Technical Support for assistance in comparing values across versions.

Display Symmetric Key Attributes (DSPSYMKEY)

The DSPSYMKEY command allows an authorized user to display the attributes for a Symmetric Key (Data Encryption Key).

NOTE: The actual value of the Symmetric Key is not displayed with this command. Only the attribute settings will be displayed.



How to Get There

From the Symmetric Encryption Key Menu, choose option **13**.

Options

Key label (KEYLABEL)

Indicate the label of the Symmetric Key to display attributes for.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

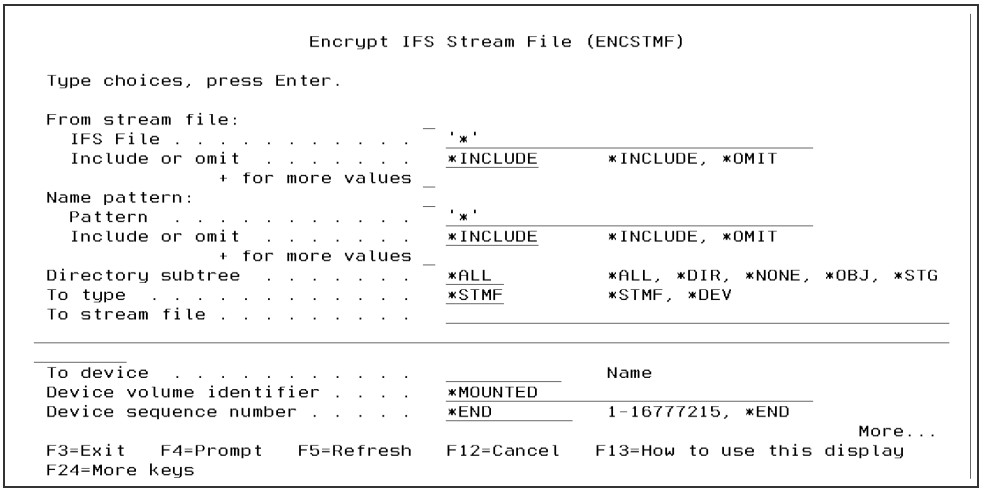
Symmetric key comparison (SYMKVV)

The symmetric key comparison value is a unique value generated for an individual key. This value is intended only for use in comparison actions to ensure the identity of a key.

> **NOTE:** This value is not the same as the actual symmetric key value.

Encrypt IFS Stream File (ENCSTMF)

The ENCSTMF command allows authorized users to encrypt IFS stream files to a device (physical or virtual) or to the IFS. Encryption algorithms provided are AES128, AES192 and AES256. Either a Symmetric Key or a Password can be specified for the encryption.



Monitoring for Errors

When executing the ENCSTMF command within a CL program, you can trap for errors by monitoring for message id CRE0700.

Auditing

If a Symmetric Key is used for the ENCSTMF command and “Log encryption usage” is enabled for the Symmetric Key, then an audit log entry will be generated in the Powertech Encryption for IBM i journal file each time the Key is used for encryption. Each audit entry will indicate the Label and Key Store of the Symmetric Key which was used, along with the user, date, time, job number and job name.

How to Get There

On the Library/Object/File Encryption Menu, choose option **5**.

Options

Object (OBJ)

Specifies the objects to be saved. You can specify an object name pattern for the path name to be used. When a path name is specified that could match many objects, you can specify a value for the Name pattern (PATTERN) parameter to subset the objects that are to be saved.

A maximum of 25 path names can be specified.

Element 1: Name

The possible values are:

'*' The objects in the current directory are saved.
path-name Specify an object path name or a pattern that can match many names.

Element 2: Include or omit

Specifies whether names that match the pattern should be included or omitted from the operation. Note that in determining whether a name matches a pattern, relative name patterns are always treated as relative to the current working directory.

> **NOTE:** The SUBTREE parameter determines whether the subtrees are included or omitted.

The possible values are:

***INCLUDE** The objects that match the object name pattern are to be saved, unless overridden by an *OMIT specification.
***OMIT** The objects that match the object name pattern are not saved. This overrides an *INCLUDE specification and is intended to be used to omit a subset of a previously selected pattern.

Name pattern (PATTERN)

Specifies one or more object name patterns to be used to subset the objects to be saved. The Objects (OBJ) parameter defines the set of candidate objects. A maximum of 25 values can be specified for this parameter.

Element 1: Pattern

The possible values are:

'*' All objects which qualify for the operation are included or omitted.
character-value Specify an object name or a pattern that can match many names.

Element 2: Include or omit

Specifies whether names that match the pattern should be included or omitted from the operation.

The possible values are:

***INCLUDE** Only objects which are included by the OBJ parameter to be saved, and match the PATTERN parameter are included in the save, unless overridden by an *OMIT specification.

***OMIT** All objects which are included by the OBJ parameter are included in the save except those objects which match the PATTERN parameter. This overrides an *INCLUDE specification and is intended to be used to omit a subset of a previously selected pattern.

Directory subtree (SUBTREE)

Specifies whether directory subtrees are included in the restore operation.

The possible values are:

***ALL** The entire subtree of each directory that matches the object name pattern is processed. The subtree includes all subdirectories and the objects within those subdirectories.

***DIR** The objects in the first level of each directory that matches the object name pattern are processed. The subdirectories of each matching directory are included, but the objects in the subdirectories are not included.

***NONE** No subtrees are included in the restore operation. If a directory matches the object name pattern specified, the objects in the directory are included. If the directory has subdirectories, neither the subdirectories nor the objects in the subdirectories are included.

***OBJ** Only the objects that match the object name pattern will be processed. If the object name pattern specifies a directory, objects in the directory are not included.

***STG** The objects that match the object name pattern are processed along with the storage for related objects. Objects can only be restored using this value if they were saved with SUBTREE(*STG).

To type (TOTYPE)

Indicate where to write the encrypted data.

The possible values are:

***STMF** Write the encrypted data into an IFS stream file.
***DEV** Write the encrypted data to a tape device.

To stream file (TOSTMF)

Specify the path to the stream (IFS) file to store the encrypted file(s).

The possible values are:

ifs-file-name

Specify the absolute IFS path to store the encrypted stream file(s). For instance: '/ABCcompany/Files/Payroll.aes'

To Device (TODEV)

Indicate the name of the device to write the encrypted data to.

Volume identifier (VOL)

Specifies the volume identifier on which the data is saved.

The possible values are:

***NONE** The data is saved on the volume placed in the device.
volume-identifier Specify the identifier of the volume for the save operation.

Sequence number (SEQNBR)

Specifies the tape sequence number to store the encrypted data.

The possible values are:

***END** The encrypted data will be saved after the last sequence number on the tape.
sequence-number Specify the sequence number to store the encrypted data. Valid values range from 1 through 16777215.

Label (LABEL)

Specifies the name that identifies the data file on the tape that is to be used for the save operation.

The possible values are:

data-file-identifier Specify the data file identifier of the data file used for the save operation. A maximum of 17 characters can be used.

File expiration date (EXPDATE)

Specifies the expiration date of the file created by the save operation. If a date is specified, the file is protected and cannot be overwritten until the specified expiration date. The expiration date must be later than or equal to the current date.

> **NOTE:** Specifying this parameter does not protect against a later save operation specifying CLEAR(*ALL).

The possible values are:

***PERM** The file is protected permanently.
expiration-date Specify the date when protection for the file ends.

End of media option (ENDOPT)

Specifies the operation that is automatically done on the tape after the save operation ends.

The possible values are:

***REWIND** The tape is automatically rewound, but not unloaded, after the operation has ended.
***LEAVE** The tape does not rewind or unload after the operation ends. It remains at the current position on the tape drive.
***UNLOAD** The tape is automatically rewound and unloaded after the operation ends.

Target release (TGTRLS)

Specifies the release of the operating system on which you intend to restore and use the object.

When specifying the target-release value, the format VxRxMx is used to specify the release, where Vx is the version, Rx is the release, and Mx is the modification level. For example, V2R3M0 is version 2, release 3 modification level 0.

To specify that an object be saved for distribution to a system at a different release level than the system on which the save operation is to occur, the procedure differs for program or non-program objects and by the release level on which a program object is created. If, for example, you are saving an object for distribution to a target system running on an earlier release, you have the following choices:

For program objects

- If the program object was created at a release level more current than the targeted earlier release, you must (1) create the program object again specifying the targeted earlier release, (2) save the program object specifying the targeted earlier release, and then (3) restore the program object on the target system.
- If the program object was created at the same release level as the target system, you can (1) save the program object specifying the targeted earlier release and then (2) restore the program object on the target system.

For non-program objects

- You can (1) save the object specifying the targeted earlier release and then (2) restore the object on the target system.

The possible values are:

***CURRENT** The object is to be restored to, and used on, the release of the operating system currently running on your system. The object can also be restored to a system with any subsequent release of the operating system installed.

***PRV** The object is to be restored to the previous release with modification level 0 of the operating system. The object can also be restored to a system with any subsequent release of the operating system installed.

target-release Specify the release in the format VxRxMx. The object can be restored to a system with the specified release or with any subsequent release of the operating system installed. Valid values depend on the current version, release, and modification level, and they change with each new release.

Update history (UPDHST)

Specifies whether the save history information of each saved object is changed with the date, time, and location of this save operation. The save history information for an object is displayed using the Display Object Description (DSPOBJD) command. The save history information is used to determine which journal entries are processed when RCVRNG (*LASTSAVE) and FROMENT(*LASTSAVE) are used on the Apply

Journaled Changes (APYJRNCHG) command.

The possible values are:

***YES** The last save date, time, and location is updated in each object saved.
***NO** The save history information contained in the description of each object saved is not updated.

Object pre-check (PRECHK)

Specifies whether the save operation for a library ends if any of the following are true:

- The objects do not exist
- The library or the objects were previously found to be damaged
- The library or the objects are locked by another job
- The requester of the save operation does not have authority to the library or to save the objects.

The possible values are:

***NO** The save operation for a library continues, saving only those objects that can be saved.
***YES** If, after all specified objects are checked, one or more objects cannot be saved, the save operation for a library ends before any data is written. If multiple libraries are specified, the save operation continues with the next library.

Save active (SAVACT)

Specifies whether an object can be updated while it is being saved.

> **NOTE:** If your system is in a restricted state, this parameter is ignored and the save operation is performed as if SAVACT(*NO) was specified.

The possible values are:

***NO** Objects that are in use are not saved. Objects cannot be updated while being saved.

***YES** Objects can be saved and used at the same time. The object checkpoints can occur at different times.
***SYNC** Objects can be saved and used at the same time. All of the object checkpoints occur at the same time.

Save active option (SAVACTOPT)

Specifies options to be used with the save while active parameter.

The possible values are:

***NONE** No special save while active options will be used.
***ALWCKPWRT** Enables objects to be saved while they are being updated if the corresponding system attribute for the object is set.

> NOTE: This option should only be used by applications to save objects that are associated with the application and that have additional backup and recovery considerations. For more information see the Backup and Recovery book, SC41-5304.

Save active message queue (SAVACTMSGQ)

Specifies the message queue that the save operation uses to notify the user that the checkpoint processing is complete. For more information on specifying path names, refer to "Object naming rules" in "CL concepts and reference" in the CL concepts and reference topic in the iSeries Information Center at http://www.ibm.com/eserver/iseries/infocenter.

The possible values are:

***NONE** No notification message is sent.
***WRKSTN** The notification message is sent to the work station message queue.
path-name Specify the path name of the message queue to be used.

ASP device (ASPDEV)

Specifies the name of the auxiliary storage pool (ASP) device to be included in the save operation.

The possible values are:

***** The operation includes the system ASP (ASP number 1), all basic user ASPs (ASP numbers 2-32), and, if the current thread has an ASP group, all independent ASPs in the ASP group.

***SYSBAS** The system ASP and all basic user ASPs are included in the save operation.

***CURASPGRP** If the current thread has an ASP group, all independent ASPs in the ASP group are included in the save operation.
auxiliary-storage-pool-device-name The specified independent ASP is included in the save operation.

Algorithm (ALGORITHM)

Indicate which algorithm to use to encrypt the file. The default is *AES256.

The possible values are:

***AES128** A 128 bit key size is utilized for the encryption process. This is the fastest encryption option and the least secure.
***AES192** A 192 bit key size is utilized for the encryption process.
***AES256** A 256 bit key size is utilized for the encryption process. This is the slowest encryption option and the most secure.

Compress data (COMPRESS)

Specifies whether to compress the data during the backup. The compression option may increase the save times. This command uses the TERSE compression algorithm.

The possible values are:

***YES** Compress the data.
***NO** Do not compress the data.

Use key or password (USEKEYPAS)

Indicate to use a either a key from a key store or a password to encrypt the data. The default is *KEY.

The possible values are:

***KEY** Use a key from a key store to encrypt the data.
***PASS** Use a password to encrypt the data.

Key label (KEYLABEL)

Indicate the label of the key to use for encrypting the data.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for encryption of the data.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Store key information (STRKEYINF)

Indicate whether to store the key label and key store library/name in the encrypted data. This is useful in that you will not have to remember which key label to use on the decryption process. The default is *YES.

The possible values are:

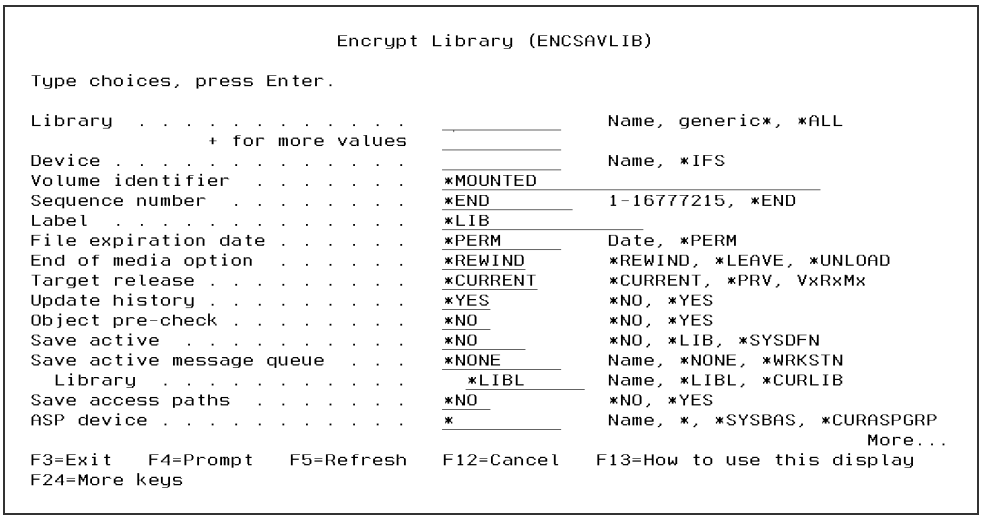
***YES** Store the key label and key store library/name in the encrypted data.
***NO** Do not store the key label and key store library/name in the encrypted data.

Password (PASSWORD)

Indicate the password to use to encrypt the data with.

Encrypt Library (ENCSAVLIB)

The ENCSAVLIB command allows authorized users to encrypt and save a copy of one or more libraries to a device (physical or virtual) or to the IFS. Encryption algorithms provided are AES128, AES192 and AES256. Either a Symmetric Key or a Password can be specified for the encryption process.



The ENCSAVLIB command saves the entire library, including the library description, the object descriptions and the contents of the objects in the library. The libraries and their objects are not affected in the system.

For job queues, message queues, output queues, and logical files, only the object definitions are saved, not the contents. Logical file access paths may be saved, however, by using the ACCPTH parameter.

Make sure the QSYSWRK subsystem is active for support of the ENCSAVLIB command.

Monitoring for Errors

When executing the ENCSAVLIB command within a CL program, you can trap for errors by monitoring for message ids. The failure message ids for the ENCSAVLIB command are listed below:

CRE0712 - Library(s) were not encrypted. Review JOB LOG.
CRE3701 - &1 objects were saved; &2 objects were not saved.
CRE3751 - Some Libraries not saved.

Auditing

If a Symmetric Key is used for the ENCSAVLIB command and “Log encryption usage” is enabled for the Symmetric Key, then an audit log entry will be generated in the Powertech Encryption for IBM i journal file each time the Key is used for encryption.

Each audit entry will indicate the Label and Key Store of the Symmetric Key that was used, along with the user, date, time, job number and job name.

How to Get There

On the Library/Object/File Encryption Menu, choose option 1.

Options

Library (LIB)

Specifies which libraries are saved.

***ALL** All libraries will be saved. The exception is that the following IBM libraries will NOT be saved: QDOC, QDOCxxxx, QRCYxxxxx, QRECOVERY, QRPLOBJ, QRPLxxxxx, QSPL, QSPLxxxx, QSRV, QSYS, QSYSxxxxx and QTEMP.

generic*-library-name Specify the generic name of the library. A generic name is a character string of one or more characters followed by an asterisk (*); for example, ABC*. The asterisk (*) substitutes for any valid characters. A generic name specifies all libraries with names that begin with the generic prefix, for which the user has authority. If an asterisk is not included with the generic (prefix) name, the system assumes it to be the complete library name. A maximum of 300 generic library names can be specified.

library-name Specify the names of a maximum of 300 libraries to be saved. The libraries QDOC, QDOCxxxx, QRCYxxxxx, QRECOVERY, QRPLOBJ, QRPLxxxxx, QSPL, QSPLxxxx, QSRV, QSYS, QSYSxxxxx, and QTEMP cannot be specified.

Device (DEV)

Specifies the name of the device used for the save operation. The device name must already be known on the system by a device description. This is a required parameter.

The possible values are:

diskette-device-name Specify the name of the diskette device used for the save operation.
optical-device-name Specify the name of the optical device used for the save operation.
tape-device-name Specify the name of the tape device used for the save operation.
***IFS** The library is encrypted into a Stream file on the IFS, but is not saved to a device.

Volume identifier(VOL)

Specifies the volume identifier on which the data is saved.

The possible values are:

***MOUNTED** The data is saved on the volumes placed in the device.

volume-identifier Specify the identifier of the volume for the save operation.

Sequence number (SEQNBR)

Specifies, when tape is used, the sequence number to use as the starting point for the save operation.

> **NOTE:** Each library will be saved onto its own sequence number.

The possible values are:

***END** The save operation begins after the last sequence number on the first tape. If the first tape is full, an error message is issued and the operation ends.
file-sequence-number Specify the sequence number of the file to be used for the save operation. Valid values range from 1 through 16777215.

Label (LABEL)

Specifies the name that identifies the data file on the tape or diskette that is to be used for the save operation.

The possible values are:

***LIB** The file label is created by the system using the name of the library specified on the Library prompt (LIB parameter).
data-file-identifier Specify the data file identifier of the data file used for the save operation. A maximum of 17 characters can be used.

File expiration date (EXPDATE)

Specifies the expiration date of the file created by the save operation. If a date is specified, the file is protected and cannot be overwritten until the specified expiration date. The expiration date must be later than or equal to the current date.

> **NOTE:** This parameter is valid for tape, diskette, and optical files. For save operations to diskette, the expiration date specified must be later than the date of the save operation. Otherwise, the save and restore files whose expiration date has been exceeded may be lost when the next save and restore file is written during the save operation.

> **NOTE:** Specifying this parameter does not protect against a later save operation specifying CLEAR(*ALL).

The possible values are:

***PERM** The file is protected permanently.
expiration-date Specify the date when protection for the file ends.

End of media option (ENDOPT)

Specifies the operation that is automatically done on the tape or optical volume after the save operation ends.

> **NOTE:** This parameter is valid only if a tape or optical device name is specified on the DEV parameter. For optical devices, *UNLOAD is the only special value supported, *REWIND and *LEAVE will be ignored.

The possible values are:

***REWIND** The tape is automatically rewound, but not unloaded, after the operation has ended.
***LEAVE** The tape does not rewind or unload after the operation ends. It remains at the current position on the tape drive.
***UNLOAD** The tape is automatically rewound and unloaded after the operation ends. Some optical devices will eject the volume after the operation ends.

Target release (TGTRLS)

Specifies the release of the operating system on which you intend to restore and use the object.

When specifying the target-release value, the format VxRxMx is used to specify the release, where Vx is the version, Rx is the release, and Mx is the modification level. For example, V2R3M0 is version 2, release 3 modification level 0.

To specify that an object be saved for distribution to a system at a different release level than the system on which the save operation is to occur, the procedure differs for program or non-program objects and by the release level on which a program object is created. If, for example, you are saving an object for distribution to a target system running on an earlier release, you have the following choices:

For program objects

- If the program object was created at a release level more current than the targeted earlier release, you must (1) create the program object again specifying the targeted earlier release, (2) save the program object specifying the targeted earlier release, and then (3) restore the program object on the target system.

- If the program object was created at the same release level as the target system, you can (1) save the program object specifying the targeted earlier release and then (2) restore the program object on the target system.

For non-program objects

- You can (1) save the object specifying the targeted earlier release and then (2) restore the object on the target system.

The possible values are:

***CURRENT** The object is to be restored to, and used on, the release of the operating system currently running on your system. The object can also be restored to a system with any subsequent release of the operating system installed.

***PRV** The object is to be restored to the previous release with modification level 0 of the operating system. The object can also be restored to a system with any subsequent release of the operating system installed.

target-release Specify the release in the format VxRxMx. The object can be restored to a system with the specified release or with any subsequent release of the operating system installed. Valid values depend on the current version, release, and modification level, and they change with each new release.

Update history (UPDHST)

Specifies whether the save history information of each saved object is changed with the date, time, and location of this save operation. The save history information for an object is displayed using the Display Object Description (DSPOBJD) command. The save history information is used to determine which journal entries are processed when RCVRNG (*LASTSAVE) and FROMENT(*LASTSAVE) are used on the Apply Journaled Changes (APYJRNCHG) command.

The possible values are:

***YES** The last save date, time, and location is updated in each object saved.
***NO** The save history information contained in the description of each object saved is not updated.

Object pre-check (PRECHK)

Specifies whether the save operation for a library ends if any of the following are true:

- The objects do not exist
- The library or the objects were previously found to be damaged
- The library or the objects are locked by another job
- The requester of the save operation does not have authority to the library or to save the objects.

The possible values are:

***NO** The save operation for a library continues, saving only those objects that can be saved.
***YES** If, after all specified objects are checked, one or more objects cannot be saved, the save operation for a library ends before any data is written. If multiple libraries are specified, the save operation continues with the next library.

Save active (SAVACT)

Specifies whether an object can be updated while it is being saved.

> **NOTE:** If your system is in a restricted state and the SAVACT parameter is specified, the save operation is performed as if SAVACT(*NO) was specified.

The possible values are:

***NO** Objects that are in use are not saved. Objects cannot be updated while being saved.
***LIB** Objects in a library can be saved while they are in use by another job. All of the objects in a library reach a checkpoint together and are saved in a consistent state in relationship to each other.

> **NOTE:** Libraries with thousands of objects may be too large for this option.

***SYSDFN** Objects in a library can be saved while they are in use by another job. Objects in a library may reach checkpoints at different times and may not be in a consistent state in relationship to each other.

> **NOTE:** Note: Specifying this value eliminates some size restrictions and may enable a library to be saved that could not be saved with SAVACT(*LIB).

Save active wait time (SAVACTWAIT)

Specifies the amount of time to wait for a commit boundary or an object that is in use before continuing the save. If an object remains in use for the specified time, the object is not saved. If a commit boundary is not reached in the specified time, the save operation is ended.

The possible values are:

120 The system waits up to 120 seconds for a commit boundary or an object lock before continuing the save operation.
***NOMAX** No maximum wait time exists.
wait-time Specify the time (in seconds) to wait for a commit boundary or an object lock before continuing the save operation. Valid values range from 0 through 99 999.

Save active message queue (SAVACTMSGQ)

Specifies the message queue that the save operation uses to notify the user that the checkpoint processing for a library is complete. A separate message is sent for each library to be saved when the *SYSDFN or *LIB value is specified on the Save active prompt (SAVACT parameter).

The possible values are:

***NONE** No notification message is sent.
***WRKSTN** The notification message is sent to the work station message queue. This value is not valid in batch mode.
message-queue-name Specify the name of the message queue.

The possible library values are:

***LIBL** All libraries in the job's library list are searched until the first match is found.
***CURLIB** The current library for the job is used to locate the message queue. If no library is specified as the current library for the job, the QGPL library is used.
library-name Specify the name of the library where the message queue is located.

Save access paths (ACCPTH)

Specifies whether the logical file access paths that are dependent on the physical files being saved are also saved.

The access paths are saved only in the case of the following:

- All members on which the access paths are built are included in this save operation.
- The access paths are not invalid or damaged at the time of the save.

> **NOTE:** If the based-on physical files and the logical files are in different libraries, the access paths are saved.

However, if the logical files and the based-on physical files are in different libraries and the logical files or physical files do not exist at restore time (such as during disaster recovery or the files were deleted) the access paths are not restored. They are rebuilt.

For the fastest possible restore operation for logical files, the logical files and the based-on physical files must be in the same library and must be saved at the same time.

The possible values are:

***NO** Only those objects specified on the command are saved. No logical file access paths are saved.
***YES** The specified physical files and all eligible logical file access paths over them are saved.

> **NOTE:** Specifying this value does not save the logical files.

ASP device (ASPDEV)

Specifies the name of the auxiliary storage pool (ASP) device to be included in the save operation.

The possible values are:

***** The operation includes the system ASP (ASP number 1), all basic user ASPs (ASP numbers 2-32), and, if the current thread has an ASP group, all independent ASPs in the ASP group.

***SYSBAS** The system ASP and all basic user ASPs are included in the save operation.

***CURASPGRP** If the current thread has an ASP group, all independent ASPs in the ASP group are included in the save operation.

auxiliary-storage-pool-device-name The specified independent ASP is included in the save operation.

Algorithm (ALGORITHM)

Indicate which algorithm to use to encrypt the data.

The default is *AES256

The possible values are:

***AES128** A 128 bit key size is utilized for the encryption process. This is the fastest encryption option and the least secure.

***AES192** A 192 bit key size is utilized for the encryption process.
***AES256** A 256 bit key size is utilized for the encryption process. This is the slowest encryption option and the most secure.

Compress data (COMPRESS)

Specifies whether to compress the data during the backup. The compression option may increase the save times. This command uses the TERSE compression algorithm.

The possible values are:

***YES** Compress the data.
***NO** Do not compress the data.

Use key or password (USEKEYPAS)

Indicate to use either a key from a key store or a password to encrypt the data.

The default is *KEY

The possible values are:

***KEY** Use a key from a key store to encrypt the data.
***PASS** Use a password to encrypt the data.

Key label (KEYLABEL)

Indicate the label of the key to use for encrypting the data.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for encryption of the data.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Store key information (STRKEYINF)

Indicate whether to store the key label and key store library/name in the encrypted data. This is useful in that you will not have to remember which key label to use on the decryption process. The default is *YES.

The possible values are:

***YES** Store the key label and key store library/name in the encrypted data.
***NO** Do not store the key label and key store library/name in the encrypted data.

Password (PASSWORD) - Help

Specify the password to encrypt the data.

The possible values are:

password-value Enter a password up to 32 characters in length. Mixed case characters can be entered.

> **NOTE:** The password is case-sensitive.

***SRLNBR** The system serial number is used for the password. This value is retrieved from the system value of QSRLNBR.

Encrypted file directory (ENCFDIR) - Help

Specifies the IFS directory to store the encrypted Stream file(s).

An encrypted Stream file will be created in this IFS directory for each library saved. The name of each Stream file will correspond with the name of the library saved into it.

The file extension of .AES will be appended.

For instance, the library named OEDATA would be encrypted into a Stream file named OEDATA.AES.

The possible values are:

/CRYPTOTEMP The IFS directory named /CRYPTOTEMP will be used to store the encrypted Stream file(s). This directory will be created if it does not exist.

directory-name Specify the IFS directory name to store the encrypted Stream file(s). This directory will be created if it does not exist.

Libraries to omit (OMITLIB) - Help

Specifies the names of one of more libraries, or the generic names of each group of libraries, to be excluded from the save operation. Up to 25 library(s) or generic names can be entered.

The possible values are:

***NONE** No libraries are excluded from the save operation.

generic-library-name Specify the generic name of the libraries to be excluded. A generic name is a character string of one or more characters followed by an asterisk (*); for example, ABC*. The asterisk (*) substitutes for any valid characters. A generic name specifies all libraries with names that begin with the generic prefix, for which the user has authority. If an asterisk is not included with the generic (prefix) name, the system assumes it to be the complete library name.

library-name Specify the name of the library to be excluded from the save operation.

Objects to omit (OMITOBJ) - Help

Specifies the objects to be excluded from the operation. Up to 25 objects or generic object values can be specified. Specify the object name(s), library(s) and type(s) to be omitted.

The possible values are:

object-name Specify the name of the object to omit.
generic-object-name Specify the generic name of object to omit. A generic name is specified as a character string that contains one or more characters followed by an asterisk (*); for example, ABC*.
***ALL** Omit all objects in the library and type specified.

The possible library values are:

library-name Specify the name of the library where the object exists to omit.
***ALL** The specified objects are excluded from all libraries that are part of the operation.

The possible type values are:

***ALL** All object types are omitted.
character-value Specify the object type of the objects to be excluded from the operation. To see a complete list of object types when prompting this command, position the cursor on the field for this parameter and press F4 (Prompt).

Output information (OUTPUT)

Specifies whether a list with information about the saved objects is created. The information can be printed with the job's spooled output or directed to a database file.

The possible values are:

***NONE** No output listing is created.
***PRINT** The output is printed with the job's spooled output.
***OUTFILE** The output is directed to the database file specified for the File to receive output (OUTFILE) parameter. Note: You must specify a database file name for the File to receive output (OUTFILE) parameter when OUTPUT(*OUTFILE) is specified.

File to receive output (OUTFILE) - Help

Specifies the database file to which the information is directed when *OUTFILE is specified for the Output (OUTPUT) parameter. If the file does not exist, this command creates a database file in the specified library. If a new file is created, the system uses QASAVOBJ in QSYS with the format name QSRSAV as a model.

The possible values are:

file-name Specify the name of the database file to which output from the command is directed. If this file does not exist, it is created in the specified library.

The possible library values are:

library-name Specify the name of the library to be searched.
***LIBL** All libraries in the library list for the current thread are searched until the first match is found.
***CURLIB** The current library for the thread is used to locate the file. If no library is specified as the current library for the job, the QGPL library is used.

Output member options (OUTMBR) - Help

Specifies the name of the database file member to which the output is directed when *OUTFILE is specified for the Output (OUTPUT) parameter.

Member to receive output

The possible values are:

member-name Specify the name of the database file to which output from the command is directed. If this file does not exist, it is created in the specified library.

***FIRST** The first member in the file receives the output. If OUTMBR(*FIRST) is specified and the member does not exist, the system creates a member with the name of the file specified for the File to receive output (OUTFILE) parameter.

Output option (OUTOPT) - Help

Specifies the output option for the file member to which the output is directed when *OUTFILE is specified for the Output (OUTPUT) parameter.

The possible values are:

***REPLACE** The existing records in the specified database file member are replaced by the new records.
***ADD** The new records are added to the existing information in the specified database file member.

Type of output information (INFTYPE) - Help

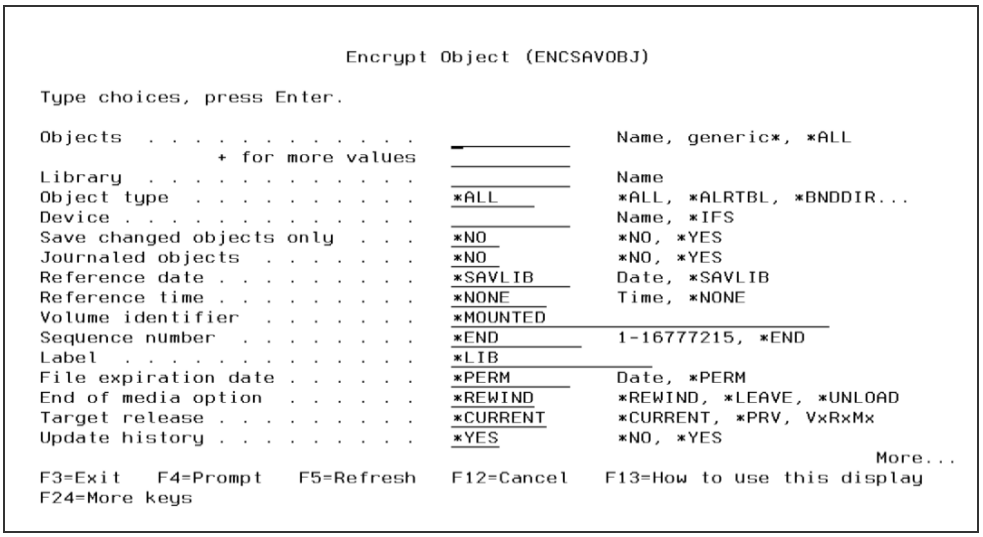
Specifies the type of information which is printed or directed to the database file.

The possible values are:

***OBJ** The list contains an entry for each object requested to be saved.
***ERR** The list contains information about the command, an entry for each library, and an entry for each object that was not successfully saved.
***LIB** The list contains a library entry for each library requested to be saved.
***MBR** The list contains an entry for each object or, for database files, each member requested to be saved.

Encrypt Object (ENCSAVOBJ)

The ENCSAVOBJ command allows authorized users to encrypt and save a copy of one or more objects to a device (physical or virtual) or to the IFS. Encryption algorithms provided are AES128, AES192 and AES256. Either a Symmetric Key or a Password can be specified for the encryption.



The system will save the specified objects by writing a copy of each object. The objects are not affected in the system. However, the description of each object is changed with the date, time, and place when it was last saved, unless *NO is specified on the Update history prompt (UPDHST parameter).

For job queues, message queues, output queues, and logical files, only the object definitions are saved, not the contents. Logical file access paths may be saved, however, by using the ACCPTH parameter.

Make sure the QSYSWRK subsystem is active for support of the ENCSAVOBJ command.

Monitoring for Errors

When executing the ENCSAVOBJ command within a CL program, you can trap for errors by monitoring for message ids. The message ids for the ENCSAVOBJ command are listed below:

CRE0713 - Object(s) were not encrypted. Review JOB LOG.
CRE3701 - &1 objects were saved; &2 objects were not saved.

Auditing

If a Symmetric Key is used for the ENCSAVOBJ command and “Log encryption usage” is enabled for the Symmetric Key, then an audit log entry will be generated in the Powertech Encryption for IBM i journal file each time the Key is used for encryption.

Each audit entry will indicate the Label and Key Store of the Symmetric Key which was used, along with the user, date, time, job number and job name.

How to Get There

On the Library/Object/File Encryption Menu, choose option **3**.

Options

Objects (OBJ)

Specifies the names of one or more objects or the generic name of each group of objects to be saved. All the objects must be in the library specified on the Library prompt (LIB parameter). If the Object type prompt (OBJTYPE parameter) is not specified, all the object types listed in the description of that parameter are saved, provided they are in the specified library and have the specified names. This is a required parameter.

The possible values are:

***ALL** All the objects in the specified library are saved, depending on the value specified on the Object type prompt (OBJTYPE parameter).

generic*-object-name Specify one or more generic names of groups of objects in the specified library to be saved. A generic name is a character string that contains one or more characters followed by an asterisk (*). If an * is not specified with the name, the system assumes that the name is a complete object name.

object-name Specify one or more names of specific objects to save. Both generic names and specific names can be specified in the same command. A maximum of 300 object names can be specified.

Library (LIB)

Specifies the name of the library that contains the objects. This is a required parameter.

The possible values are:

library-name Specify the name of the library containing the objects.

Object type (OBJTYPE)

Specifies the type of system objects to save. For a complete list of object types that can be saved, move the cursor to the field for the Object type prompt (OBJTYPE parameter) and press the F4 key.

The possible values are:

***ALL** All object types that are specified by name and are in the specified library are saved. If *ALL is also specified on the Objects prompt (OBJ parameter), then all the objects in the library that are of the types that can be saved are saved.

object-type Specify the value for the types of objects that are saved, such as command (*CMD), file (*FILE) or program (*PGM).

Device (DEV)

Specifies the name of the device used for the save operation. The device name must already be known on the system by a device description. This is a required parameter.

The possible values are:

diskette-device-name Specify the name of the diskette device used for the save operation.
optical-device-name Specify the name of the optical device used for the save operation.
tape-device-name Specify the name of the tape device used for the save operation.
***IFS** The save file is encrypted into a Stream file on the IFS, but is not saved to a device.

Save changed objects only (SAVCHGOBJ)

Specify whether or not to save changed objects.

The possible values are:

***YES** Objects changed since the specified date and time are saved.
***NO** The objects changed date and time will be ignored when determining if they should be saved or not.

Journaled objects (OBJJRN)

Specifies whether to save changed objects that are currently being journaled and that have been journaled since the date and time specified on the REFDATE and REFTIME parameters.

The possible values are:

***YES** Objects whose changes are entered in a journal are saved.
***NO** Objects being journaled are not saved. If journaling was started after the specified date and time, the changed objects or changed database file members are saved. The date and time of the last journal start operation can be shown by using the Display Object Description (DSPOBJD) command.

Reference date (REFDATE)

Specifies the reference date. Objects that have been changed since this date are saved.

The possible values are:

***SAVLIB** The objects that have been changed since the date of the last running of the Save Library (SAVLIB) command are saved. If the specified library was never saved, a message is issued and the library is not saved, but the operation continues.

reference-date Specify the reference date. Objects that have been changed since this date are saved. If you specify a date later than the date of the running of this command, a message is issued and the operation ends. The date must be specified in the job date format.

Reference time (REFTIME)

Specifies the reference time. Objects that have been changed since this time on the specified date are saved.

The possible values are:

***NONE** No explicit time is specified. Any objects changed since the date specified by the Reference date prompt (REFDATE parameter) are saved.

reference-time Specify the reference time. Objects that have been changed since this time on the specified date are saved. If *SAVLIB is specified on the Reference date prompt (REFDATE parameter), no reference time can be specified. If you specify a time later than the time of the running of this command, a message is issued and the operation ends. The time can be specified in the format of hhmmss.

Volume identifier (VOL)

Specifies the volume identifier on which the data is saved.

The possible values are:

***MOUNTED** The data is saved on the volumes placed in the device.
volume-identifier Specify the identifier of the volume for the save operation.

Sequence number (SEQNBR)

Specifies, when tape is used, the sequence number to use as the starting point for the save operation.

The possible values are:

***END** The save operation begins after the last sequence number on the first tape. If the first tape is full, an error message is issued and the operation ends.
file-sequence-number Specify the sequence number of the file to be used for the save operation. Valid values range from 1 through 16777215.

Label (LABEL)

Specifies the name that identifies the data file on the tape or diskette that is to be used for the save operation.

The possible values are:

***LIB** The file label is created by the system using the name of the library specified for the Library (LIB) parameter.
data-file-identifier Specify the data file identifier of the data file used for the save operation. A maximum of 17 characters can be used.

File expiration date (EXPDATE)

Specifies the expiration date of the file created by the save operation. If a date is specified, the file is protected and cannot be overwritten until the specified expiration date. The expiration date must be later than or equal to the current date.

> **NOTE:** This parameter is valid for tape, diskette, and optical files. For save operations to diskette, the expiration date specified must be later than the date of the save operation. Otherwise, the save and restore files whose expiration date has been exceeded may be lost when the next save and restore file is written during the save operation.

> **NOTE:** Specifying this parameter does not protect against a later save operation specifying CLEAR(*ALL).

The possible values are:

***PERM** The file is protected permanently.
expiration-date Specify the date when protection for the file ends.

End of media option (ENDOPT)

Specifies the operation that is automatically done on the tape or optical volume after the save operation ends.

> **NOTE:** This parameter is valid only if a tape or optical device name is specified on the DEV parameter. For optical devices, *UNLOAD is the only special value supported, *REWIND and *LEAVE will be ignored.

The possible values are:

***REWIND** The tape is automatically rewound, but not unloaded, after the operation has ended.
***LEAVE** The tape does not rewind or unload after the operation ends. It remains at the current position on the tape drive.
***UNLOAD** The tape is automatically rewound and unloaded after the operation ends. Some optical devices will eject the volume after the operation ends.

Target release (TGTRLS)

Specifies the release of the operating system on which you intend to restore and use the object.

When specifying the target-release value, the format VxRxMx is used to specify the release, where Vx is the version, Rx is the release, and Mx is the modification level. For example, V2R3M0 is version 2, release 3 modification level 0.

To specify that an object be saved for distribution to a system at a different release level than the system on which the save operation is to occur, the procedure differs for program or non-program objects and by the release level on which a program object is created. If, for example, you are saving an object for distribution to a target system running on an earlier release, you have the following choices:

For program objects

- If the program object was created at a release level more current than the targeted earlier release, you must (1) create the program object again specifying the targeted earlier release, (2) save the program object specifying the targeted earlier release, and then (3) restore the program object on the target system.
- If the program object was created at the same release level as the target system, you can (1) save the program object specifying the targeted earlier release and then (2) restore the program object on the target system.

For non-program objects

- You can (1) save the object specifying the targeted earlier release and then (2) restore the object on the target system.

The possible values are:

***CURRENT** The object is to be restored to, and used on, the release of the operating system currently running on your system. The object can also be restored to a system with any subsequent release of the operating system installed.

***PRV** The object is to be restored to the previous release with modification level 0 of the operating system. The object can also be restored to a system with any subsequent release of the operating system installed.

target-release Specify the release in the format VxRxMx. The object can be restored to a system with the specified release or with any subsequent release of the operating system installed. Valid values depend on the current version, release, and modification level, and they change with each new release.

Update history (UPDHST)

Specifies whether the save history information of each saved object is changed with the date, time, and location of this save operation. The save history information for an object is displayed using the Display Object Description (DSPOBJD) command. The save history information is used to determine which journal entries are processed when RCVRNG (*LASTSAVE) and FROMENT(*LASTSAVE) are used on the Apply Journaled Changes (APYJRNCHG) command.

The possible values are:

***YES** The last save date, time, and location is updated in each object saved.
***NO** The save history information contained in the description of each object saved is not updated.

Object pre-check (PRECHK)

Specifies whether the save operation for a library ends if any of the following are true:

- The objects do not exist
- The library or the objects were previously found to be damaged
- The library or the objects are locked by another job
- The requester of the save operation does not have authority to the library or to save the objects.

The possible values are:

***NO** The save operation for a library continues, saving only those objects that can be saved.
***YES** If, after all specified objects are checked, one or more objects cannot be saved, the save operation for a library ends before any data is written.

Save active (SAVACT)

Specifies whether an object can be updated while it is being saved.

> NOTE: If your system is in a restricted state and the SAVACT parameter is specified, the save operation is performed as if SAVACT(*NO) was specified.

The possible values are:

***NO** Objects that are in use are not saved. Objects cannot be updated while being saved.
***LIB** Objects in a library can be saved while they are in use by another job. All of the objects in a library reach a checkpoint together and are saved in a consistent state in relationship to each other.

> NOTE: Libraries with thousands of objects may be too large for this option.

***SYSDFN** Objects in a library can be saved while they are in use by another job. Objects in a library may reach checkpoints at different times and may not be in a consistent state in relationship to each other.

> NOTE: Specifying this value eliminates some size restrictions and may enable a library to be saved that could not be saved with SAVACT(*LIB).

Save active message queue (SAVACTMSGQ)

Specifies the message queue that the save operation uses to notify the user that the checkpoint processing for a library is complete. A separate message is sent for each library to be saved when the *SYSDFN or *LIB value is specified on the Save active prompt (SAVACT parameter)

The possible values are:

***NONE** No notification message is sent.
***WRKSTN** The notification message is sent to the work station message queue. This value is not valid in batch mode.
message-queue-name Specify the name of the message queue.

The possible library values are:

***LIBL** All libraries in the job's library list are searched until the first match is found.
***CURLIB** The current library for the job is used to locate the message queue. If no library is specified as the current library for the job, the QGPL library is used.
library-name Specify the name of the library where the message queue is located.

Save access paths (ACCPTH)

Specifies whether the logical file access paths that are dependent on the physical files being saved are also saved.

The access paths are saved only in the case of the following:

- All members on which the access paths are built are included in this save operation.
- The access paths are not invalid or damaged at the time of thesave.

> **NOTE:** If the based-on physical files and the logical files are in different libraries, the access paths are saved.
> However, if the logical files and the based-on physical files are in different libraries and the logical files or physical files do not exist at restore time (such as during disaster recovery or the files were deleted) the access paths are not restored. They are rebuilt.
> For the fastest possible restore operation for logical files, the logical files and the based-on physical files must be in the same library and must be saved at the same time.

The possible values are:

***NO** Only those objects specified on the command are saved. No logical file access paths are saved.
***YES** The specified physical files and all eligible logical file access paths over them are saved.

> **NOTE:** Specifying this value does not save the logical files.

ASP device (ASPDEV)

Specifies the name of the auxiliary storage pool (ASP) device to be included in the save operation.

The possible values are:

***** The operation includes the system ASP (ASP number 1), all basic user ASPs (ASP numbers 2-32), and, if the current thread has an ASP group, all independent ASPs in the ASP group.
***SYSBAS** The system ASP and all basic user ASPs are included in the save operation.

***CURASPGRP** If the current thread has an ASP group, all independent ASPs in the ASP group are included in the save operation.
auxiliary-storage-pool-device-name The specified independent ASP is included in the save operation.

Algorithm (ALGORITHM)

Indicate which algorithm to use to encrypt the data. The default is *AES256.

The possible values are:

***AES128** A 128 bit key size is utilized for the encryption process. This is the fastest encryption option and the least secure.
***AES192** A 192 bit key size is utilized for the encryption process.
***AES256** A 256 bit key size is utilized for the encryption process. This is the slowest encryption option and the most secure.

Compress data (COMPRESS)

Specifies whether to compress the data during the backup. The compression option may increase the save times. This command uses the TERSE compression algorithm.

The possible values are:

***YES** Compress the data.
***NO** Do not compress the data.

Use key or password (USEKEYPAS)

Indicate to use either a key from a key store or a password to encrypt the data. The default is *KEY.

The possible values are:

***KEY** Use a key from a key store to encrypt the data.
***PASS** Use a password to encrypt the data.

Key label (KEYLABEL)

Indicate the label of the key to use for encrypting the data.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for encryption of the data.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Store key label (STRKEYINF)

Indicate whether to store the key label and key store library/name in the encrypted data. This is useful in that you will not have to remember which key label to use on the decryption process. The default is *YES.

The possible values are:

***YES** Store the key label and key store library/name in the encrypted data.
***NO** Do not store the key label and key store library/name in the encrypted data.

Password (PASSWORD) - Help

Specify the password to encrypt the data.

The possible values are:

password-value Enter a password up to 32 characters in length. Mixed case characters can be entered.

> **NOTE:** The password is case-sensitive.

***SRLNBR** The system serial number is used for the password. This value is retrieved from the system value of QSRLNBR.

Encrypted file directory (ENCFDIR) - Help

Specifies the IFS directory to store the encrypted Stream file.

The possible values are:

/CRYPTOTEMP The IFS directory named /CRYPTOTEMP will be used to store the encrypted Stream file. This directory will be created if it does not exist.
directory-name Specify the IFS directory name to store the encrypted Stream file. This directory will be created if it does not exist.

Encrypted file name (ENCFNAM) - Help

Specifies the name of the encrypted Stream file to create.

The possible values are:

ENCSAVOBJ The stream file named ENCSAVOBJ will be used to store the encrypted data. This file will be created if it does not exist.
stream-file-name Specify the name of the stream file to store the encrypted data. This file will be created if it does not exist.

Objects to omit (OMITOBJ) - Help

Specifies the objects to be excluded from the operation. Up to 25 objects or generic object values can be specified. Specify the object name(s), library(s) and type(s) to be omitted.

The possible values are:

object-name Specify the name of the object to omit.
generic-object-name Specify the generic name of object to omit. A generic name is specified as a character string that contains one or more characters followed by an asterisk (*); for example, ABC*.
***ALL** Omit all objects in the library and type specified.

The possible library values are:

library-name Specify the name of the library where the object exists to omit.
***ALL** The specified objects are excluded from all libraries that are part of the operation.

The possible type values are:

***ALL** All object types are omitted.
character-value Specify the object type of the objects to be excluded from the operation. To see a complete list of object types when prompting this command, position the cursor on the field for this parameter and press F4 (Prompt).

Output (OUTPUT)

Specifies whether a list with information about the saved objects is created. The information can be printed with the job's spooled output or directed to a database file.

The possible values are:

***NONE** No output listing is created.
***PRINT** The output is printed with the job's spooled output.
***OUTFILE** The output is directed to the database file specified for the File to receive output (OUTFILE) parameter. Note: You must specify a database file name for the File to receive output (OUTFILE) parameter when OUTPUT(*OUTFILE) is specified.

File to receive output (OUTFILE) - Help

Specifies the database file to which the information is directed when *OUTFILE is specified for the Output (OUTPUT) parameter. If the file does not exist, this command creates a database file in the specified library. If a new file is created, the system uses QASAVOBJ in QSYS with the format name QSRSAV as a model.

The possible values are:

file-name Specify the name of the database file to which output from the command is directed. If this file does not exist, it is created in the specified library.

The possible library values are:

library-name Specify the name of the library to be searched.
***LIBL** All libraries in the library list for the current thread are searched until the first match is found.
***CURLIB** The current library for the thread is used to locate the file. If no library is specified as the current library for the job, the QGPL library is used.

Output member options (OUTMBR) - Help

Specifies the name of the database file member to which the output is directed when *OUTFILE is specified for the Output (OUTPUT) parameter.

Member to receive output

The possible values are:

member-name Specify the name of the database file to which output from the command is directed. If this file does not exist, it is created in the specified library.

***FIRST** The first member in the file receives the output. If OUTMBR(*FIRST) is specified and the member does not exist, the system creates a member with the name of the file specified for the File to receive output (OUTFILE) parameter.

Replace or add records (OUTOPT) - Help

Specifies the output option for the file member to which the output is directed when *OUTFILE is specified for the Output (OUTPUT) parameter.

The possible values are:

***REPLACE** The existing records in the specified database file member are replaced by the new records.
***ADD** The new records are added to the existing information in the specified database file member.

Type of output information (INFTYPE) - Help

Specifies the type of information which is printed or directed to the database file.

The possible values are:

***OBJ** The list contains an entry for each object requested to be saved.
***ERR** The list contains information about the command, an entry for each library, and an entry for each object that was not successfully saved.
***LIB** The list contains a library entry for each library requested to be saved
***MBR** The list contains an entry for each object or, for database files, each member requested to be saved.

Save active wait time (SAVACTWAIT)

Specifies the amount of time to wait for a commit boundary or an object that is in use before continuing the save. If an object remains in use for the specified time, the object is not saved. If a commit boundary is not reached in the specified time, the save operation is ended.

The possible values are:

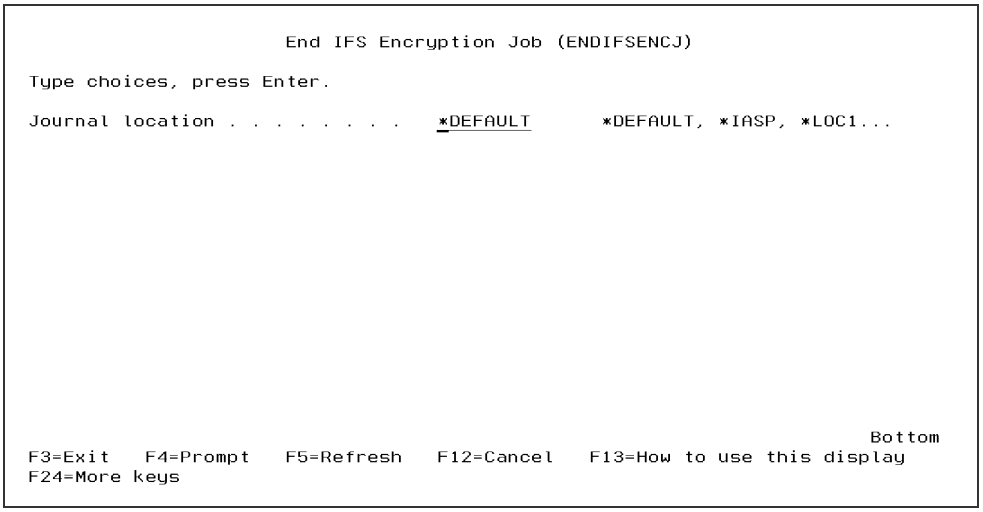
120 The system waits up to 120 seconds for a commit boundary or an object lock before continuing the save operation.
***NOMAX** No maximum wait time exists.
wait-time Specify the time (in seconds) to wait for a commit boundary or an object lock before continuing the save operation. Valid values range from 0 through 99 999.

End IFS Encryption Job (ENDIFSENCJ)

The End IFS Server Job (ENDIFSENCJ) command will notify any IFSENCJOB jobs to end. This is accomplished by changing the CRSRVRUN data area to 'N' which is monitored by the IFSENCJOB jobs.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain IFS Enc. Registry" authority setting



How to Get There

From the IFS Utility Menu, choose option **2**, End IFS Server Job. Or, prompt (**F4**) the command **CRYPTO/ENDIFSENCJ**.

Options

Journal location (JRNLOC)

Indicate which Encryption job to end.

The possible values are:

***DEFAULT** The server job (IFSENCJOB) which runs over the journal in the CRYPTO Library will be ended. All entries in the IFS Encryption Registry that use *DEFAULT for the Journal Loc will use this job (IFSENCJOB).

***IASP** The server job (IFSENCJOBA) which runs over the journal in the Journal Library defined for *IASP will be ended. All entries in the IFS Encryption Registry that use *IASP for the Journal Loc will use this job (IFSENCJOBA).

***LOC1** The server job (IFSENCJOB1) which runs over the journal in the Journal Library defined for *LOC1 will be ended All entries in the IFS Encryption Registry that use *LOC1 for the Journal Loc will use this job (IFSENCJOB1).

***LOC2** The server job (IFSENCJOB2) which runs over the journal in the Journal Library defined for *LOC2 will be ended. All entries in the IFS Encryption Registry that use *LOC2 for the Journal Loc will use this job (IFSENCJOB2).

***LOC3** The server job (IFSENCJOB3) which runs over the journal in the Journal Library defined for *LOC3 will be ended. All entries in the IFS Encryption Registry that use *LOC3 for the Journal Loc will use this job (IFSENCJOB3).

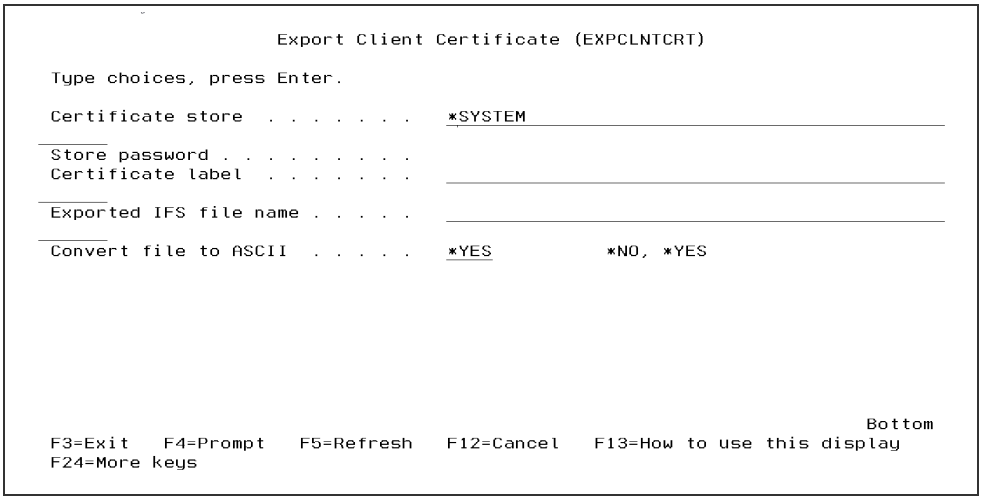
***LOC4** The server job (IFSENCJOB4) which runs over the journal in the Journal Library defined for *LOC4 will be ended. All entries in the IFS Encryption Registry that use *LOC4 for the Journal Loc will use this job (IFSENCJOB4).

***LOC5** The server job (IFSENCJOB5) which runs over the journal in the Journal Library defined for *LOC5 will be ended. All entries in the IFS Encryption Registry that use *LOC5 for the Journal Loc will use this job (IFSENCJOB5).

Export Client Certificate (EXPCLNTCRT)

The Export Client Certificate (EXPCLNTCRT) command allows authorized users to export a public client certificate from the Digital Certificate Manager into the IFS.

This command requires that the user have *ALLOBJ and *SECADM special authorities.



How to Get There

From the External Key Manager Menu, choose option **7**. Or, submit the command **EXPCLNTCRT**.

Options

Certificate Store (CRTSTR)

Specify the Certificate store name that holds the certificate to export

The possible values are:

certificate-store Enter the full path of the certificate store that holds the certificate to export.
***SYSTEM** The system certificate store will be used when exporting the certificate

Store password (PASSWORD)

Enter the password for the Certificate store where the certificate resides that will be exported.

Certificate label (CRTLBL)

Enter the label for the certificate you wish to export. CA Certificates, server and client certificates can be exported.

IFS file (IFSFIL)

Enter the full path of the IFS file to hold the exported certificate.

Convert to ASCII (CNVTASCII)

The exported certificate file and its contents by default are in EBCDIC. When this option is set to *YES then the file and its contents will be converted to ASCII.

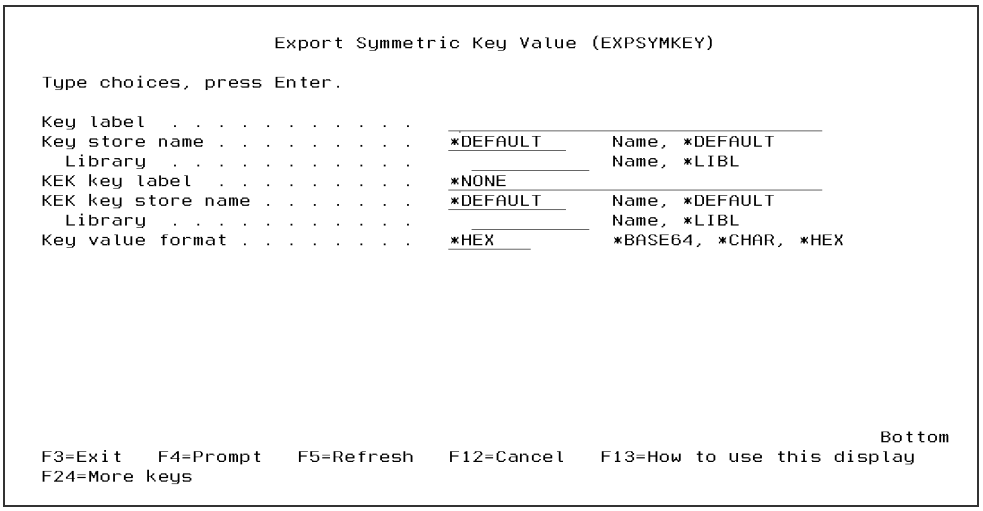
The possible values are:

***YES** The certificate will be exported to a temporary file in the /tmp directory called CRRM098.crt. If the /tmp directory does not exist then it will be created. The file will be copied to the path name in the IFSFILE parameter and converted to ASCII.

***NO** The certificate will be exported directly to the path name in the IFSFILE parameter. The file and its contents will be in EBCDIC format.

Export Symmetric Key (EXPSYMKEY)

The EXPSYMKEY command allows authorized users to extract the value of a Symmetric Key (DEK) contained within a Key Store. This command is useful if the key value needs to be shared with another computer system (which is not an IBM i) which needs to encrypt or decrypt data using the same key.



> **WARNING:** If a key value can be exported (retrieved), then the key value could be used to decrypt data without using Powertech Encryption for IBM i's APIs and security mechanisms. This security risk can be eliminated by keeping the Key Policy parameter setting DEKRTVVAL(*NO).

It is recommended to specify a KEK (Key Encryption Key) to protect the exported Symmetric Key.

The Key Policy must allow key values to be retrieved with the parameter setting of DEKRTVVAL(*YES) or (*KEK).

The following users can use the EXPSYMKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the "Maintain DEKs" authority setting

How to Get There

From the Symmetric Encryption Key Menu, choose option **15**.

Options

Key label (KEYLABEL)

Indicate the label of the Symmetric Key to export.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key.

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

KEK key label (KKEYLABEL)

Indicate the label of the Key Encryption Key (KEK) to use to encrypt the Symmetric key that will be exported.

The possible values are:

key-label Enter the key label that will be used.
***NONE** The key will not be encrypted before being exported.

KEK key store name (KKEYSTR)

Indicate the object name and library of the Key Store which contains the Key Encryption Key (KEK).

The possible values are:

kek-key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

Key value format (KEYVALFMT)

Indicate if the key should be exported in hexidecimal, base64 or character format. Generally, the key should always be exported in hexadecimal or base64 format to ensure compatibility with other computer systems.

The possible values are:

***BASE64** The key value will be displayed in base64 format.
***CHAR** The key value will be displayed in character format.

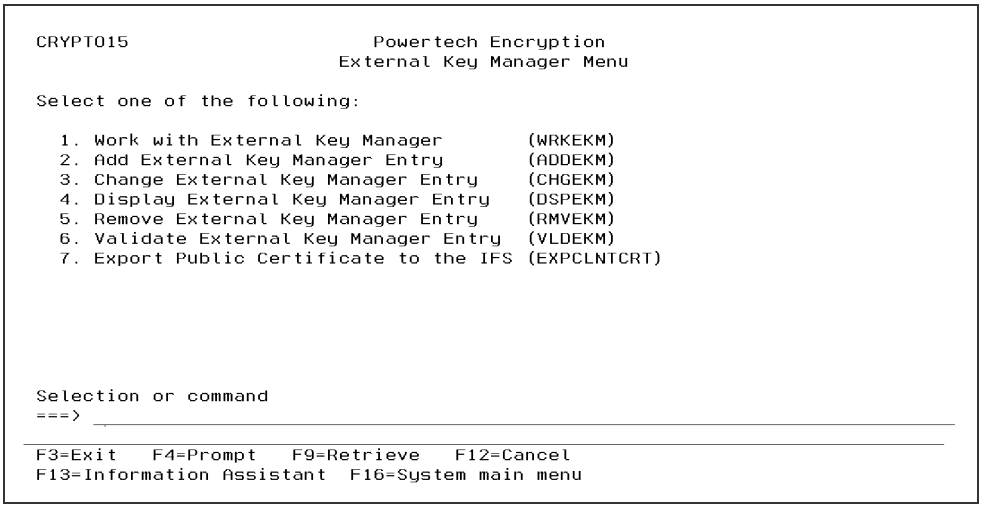
> WARNING: The key value may contain special characters which are non-displayable. Therefore it is recommended to use this setting only if the key was manually entered in character format.

***HEX** The key value will be displayed in hexidecimal format.

External Key Manager Menu

Powertech Encryption for IBM i can use keys from External Key Managers to use for encryption and decryption processes. The locations and credentials to these Key Managers must be predefined to the product before they can be used. Currently, the supported External Key Managers are Powertech Encryption for IBM i, Key Management Interoperability Protocol (KMIP), Safenet, and Vormetric.

> NOTE: If you intend to use KMIP, you will need a Client Certificate (used instead of a user ID and password), which will need to be added to your External Key Manager. See Appendix C: Adding a Client Certificate to an External Key Manager.



How to Get There

From the Main Menu, choose option **20**. Or, submit the command **GO CRYPTO/CRYPTO15**.

Options

1. Work with External Key Manager (WRKEKM)

Choose this option to open the Work with External Key Managers (WRKEKM) panel, which allows an organization to work with the external key managers in the environment.

2. Add External Key Manager Entry (ADDEKM)

Choose this option to open the Add External Key Manager (ADDEKM) panel, which allows an organization to work with the external key managers in the environment.

3. Change External Key Manager Entry (CHGEKM)

Choose this option to open the Change External Key Manager (CHGEKM) panel, which allows authorized users to change the properties for an EKM entry.

4. Display External Key Manager Entry (DSPEKM)

Choose this option to open the Display External Key Manager (DSPEKM) panel, which allows authorized users to display the properties for an External Key Manager.

5. Remove External Key Manager Entry (RMVEKM)

Choose this option to open the Remove External Key Manager Entry (RMVEKM) panel, which allows authorized users to remove an external key manager entry.

6. Validate External Key Manager Entry (VLDEKM)

Choose this option to open the Validate External Key Manager Entry (VLDEKM) panel, which allows users to test the connection to an existing key manager.

7. Export Public Certificate to the IFS (EXPCLNTCRT)

Choose this option to open the Export Client Certificate (EXPCLNTCRT) panel, which allows authorized users to export a public client certificate from the Digital Certificate Manager into the IFS.

Field Encryption Menu

Field values can be encrypted and decrypted using a variety of methods in Powertech Encryption for IBM i, providing a great deal of flexibility for an organization. For each database field, you can choose the technique to utilize based on your application requirements.

The preferred option for field encryption is to use Powertech Encryption for IBM i's innovative Field Encryption Registry, which allows an organization to indicate (register) the database fields to encrypt. When a field is "activated" in the Registry, Powertech Encryption for IBM i will perform a mass encryption of the current values for that field. Powertech Encryption for IBM i can then automatically encrypt the field values on an ongoing basis as new database records are added and when existing field values are changed.

The automated encryption function in Powertech Encryption for IBM i's Field Encryption Registry will eliminate the need to make changes to your application programs for data encryption. If DB2 Field Procedures (available in IBM i V7R1) are utilized, the values can also be automatically decrypted without program changes. Otherwise, simple program changes can be made to decrypt values using Powertech Encryption for IBM i's APIs.

You can optionally modify your applications to encrypt data through program (API) calls to Powertech Encryption for IBM i's encryption procedures and programs. Powertech Encryption for IBM i also includes stored procedures and SQL functions, which can be called from within native applications or other external clients (i.e. graphical or web-based front ends) for encryption/decryption.

Encryption Basics

Encrypted data (Cipher Text) is in alphanumeric format. Since the encryption algorithms use the full character set, you will see encrypted data as a combination of letters, special characters and numbers.

> EXAMPLE:
> **Before:** The quick brown fox jumped over the lazy dog
>
> **After:** „Œ \ËKä°BBY ý\åê·Ñ,C‹Ÿ^{F+rAÀJ[13]˜()$j1Ï(¾Y½i›”®t

Encryption Algorithms

Powertech Encryption for IBM i implements the AES and TDES encryption algorithms (ciphers). Both of these algorithms follow standard (non-proprietary) specifications as published by the United States National Institute of Standards and Technology (NIST).

The TDES (Triple DES) standard was introduced in 1998. It is so named because it applies the Data Encryption Standard (DES) cipher algorithm three times to each data block. TDES is slowly disappearing from use because of performance issues and weaker key sizes.

The AES (Advanced Encryption Standard) standard was introduced in 2001. AES is the first publicly accessible and open cipher approved by the US Government for top secret information. AES offers high performance and provides strong key lengths up to 256 bits. AES is a very popular cipher for field encryption because of those attributes.

Encryption Modes

The AES and TDES standards offer different operating “modes” that you can choose from. Powertech Encryption for IBM i provides support for the CUSP, ECB and CBC modes.

CUSP Mode (Cryptographic Unit Support Program)

CUSP mode is supported in the AES algorithm. This is a stream-based mode, which means that the length of the encrypted data will equal the length of the input data. This mode is useful if the field data is not divisible by a block length and if you want to store the encrypted values in your existing field (if not using a DB2 Field Procedure).

With CUSP mode, you can optionally specify an Initialization Vector (IV). An Initialization Vector (IV) is an arbitrary value that you can enter, which will be used as an additional input to the encryption algorithm. Therefore, the encrypted output is dependent on the combination of the Initialization Vector, Encryption Key and the Plain Text (the data you want to encrypt).

ECB Mode (Electronic Code Book)

ECB mode is supported in both AES and TDES algorithms. This is a block-based * mode. With ECB mode, you cannot specify an Initialization Vector.

CBC Mode (Cipher Block Chaining)

CBC mode is supported in both AES and TDES algorithms. This is a block-based * mode. With CBC mode, you can optionally specify an Initialization Vector (IV) to use as an additional input to the encryption algorithm.

*Notes on Block-based Modes

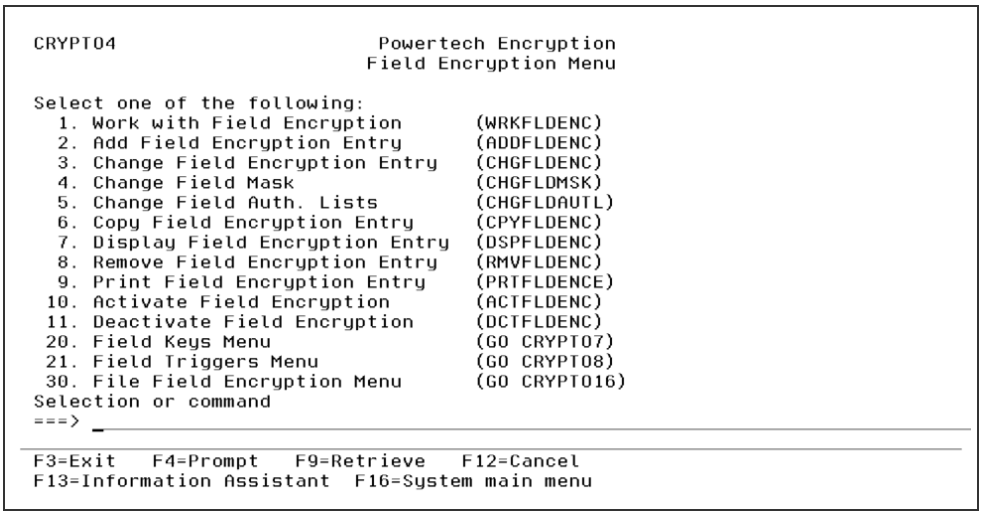
When using the AES algorithm with CBC or ECB modes, the length of the encrypted data will be a minimum of 16 bytes long. Its “block-based” length will be divisible by 16 or 24. For instance:

> NOTE: For CBC and ECB modes: If the alphanumeric field length is not divisible by the block length, then you can choose to store the encrypted values in a separate external file or use a DB2 Field Procedure.

Original Field Length	Encrypted Length
10 bytes	16 bytes
16 bytes	16 bytes
17 bytes	24 bytes
24 bytes	24 bytes
31 bytes	32 bytes

When using the TDES algorithm with CBC or ECB modes, the length of the encrypted data will be a minimum of 8 bytes long. Its “block-based” length will be divisible by 8. For instance:

Original Field Length	Encrypted Length
5 bytes	8 bytes
8 bytes	8 bytes
9 bytes	16 bytes
16 bytes	16 bytes



How to Get There

From the Main Menu, choose option **4**, Field Encryption Menu. Or, submit the command **GO CRYPTO/CRYPTO4**.

Options

1. Work with Field Encryption (WRKFLDENC)

Choose this option to open the Work with Field Encryption Registry (WRKFLDENC) panel, which allows you to specify (register) the database fields that require encryption.

2. Add Field Encryption Entry (ADDFLDENC)

Choose this option to open the Add Field Encryption Entry (ADDFLDENC) panel, which allows authorized users to add a new entry into the Field Encryption Registry.

3. Change Field Encryption Entry (CHGFLDENC)

Choose this option to open the Change Field Encryption Entry (CHGFLDENC) panel, which allows authorized users to change an entry in the Field Encryption Registry.

4. Change Field Mask (CHGFLDMSK)

Displays the Change Field Mask (CHGFLDMSK) panel, where you can change the field mask using the CHGFLDMSK command.

5. Change Field Auth. Lists (CHGFLDAUTL)

Displays the Change Authorization Lists (CHGFLDAUTL) panel, where you can change the field Authority lists using the CHGFLDAUTL command.

6. Copy Field Encryption Entry (CPYFLDENC)

Displays the Copy Field Encryption Entry (CPYFLDENC) panel, where you can copy the field entry using the CPYFLDENC command.

7. Display Field Encryption Entry (DSPFLDENC)

Displays the Display Field Encryption Entry (DSPFLDENC) panel, which shows the values for the field entry using the DSPFLDENC command.

8. Remove Field Encryption Entry (RMVFLDENC)

Displays a prompt to confirm the removal of the field entry using the RMVFLDENC command.

9. Print Field Encryption Entry (PRTFLDENCE)

Displays a prompt to print field encryption entries using the PRTFLDENCE command.

10. Activate Field Encryption (ACTFLDENC)

Displays a prompt to activate the field entry for encryption using the ACTFLDENC command.

11. Deactivate Field Encryption (DCTFLDENC)

Displays a prompt to deactivate the field entry from encryption using the DCTFLDENC command.

20. Field Keys Menu (GO CRYPTO/CRYPTO7)

Choose this option to open the Field Keys Menu.

21. Field Triggers Menu (GO CRYPTO/CRYPTO8)

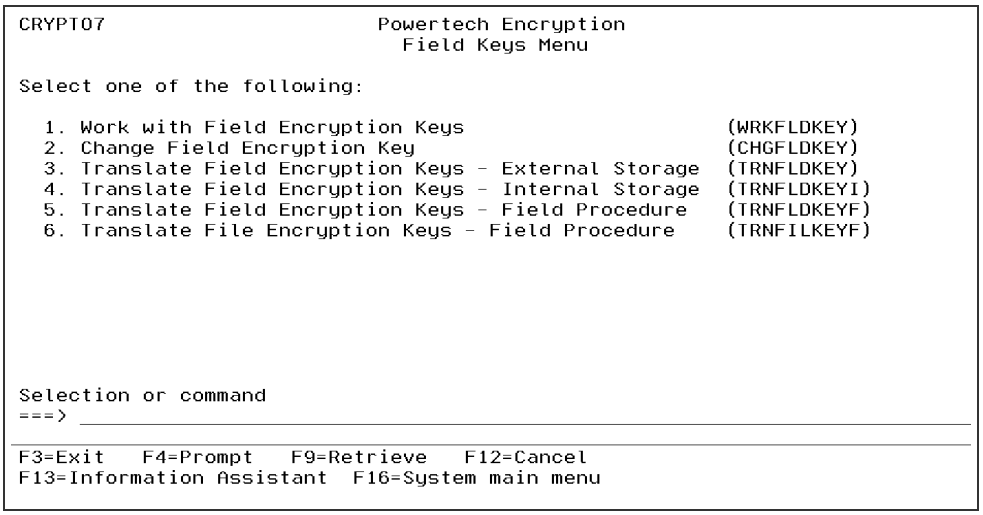
Choose this option to open the Field Triggers Menu, which allows you to Remove or Add Field Triggers.

30. File Field Encryption Menu (GO CRYPTO/CRYPTO16)

Choose this option to open the File Field Encryption Menu, where you can work with file fields.

Field Keys Menu

The Key Policy and Security Menu allows you to control the environment settings for Powertech Encryption for IBM i's Key Management System. These settings are encrypted with the Product Encryption Key (PEK) and are stored in the CRYPTO library by default.



How to Get There

From the Field Encryption Menu, choose option **20**, Field Keys Menu. Or, execute the command **GO CRYPTO/CRYPTO7**.

Options

1. Work with Field Encryption Keys (WRKFLDKEY)

Choose this option to open the Work with Field Encryption Keys (WRKFLDKEY) panel, where authorized users can display the keys used for encrypting/decrypting a field entry in the Registry.

2. Change Field Encryption Key (CHGFLDKEY)

Choose this option to open the Change Field Encryption Key (CHGFLDKEY) panel, which allows authorized users to change (rotate) the keys used to encrypt and decrypt data for a field entry in the Encryption Registry.

3. Translate Field Encryption Keys - External Storage (TRNFLDKEY)

Choose this option to open the Translate Field Encryption Keys – External Storage (TRNFLDKEY) panel, which allows authorized users to translate (re-encrypt) any field values, which were encrypted under older Keys, up to the most current Key for the specified Field Identifier.

4. Translate Field Encryption Keys - Internal Storage (TRNFLDKEYI)

Choose this option to open the Translate Field Encryption Key – Internal Storage (TRNFLDKEYI) panel (internal), which allows authorized users to translate (re-encrypt) field values to a new key.

5. Translate Field Encryption Keys - Field Procedure (TRNFLDKEYF)

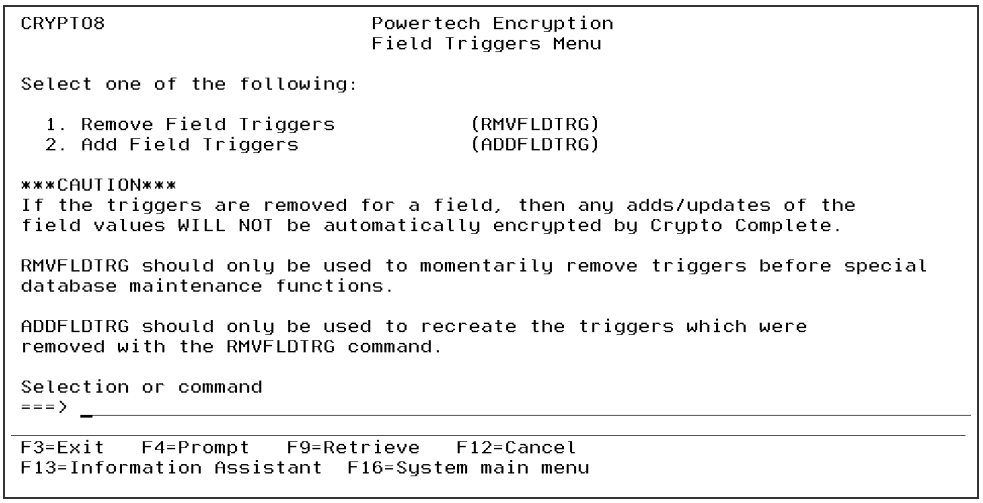
Choose this option to open the Translate Field Encryption Key – Field Procedure (TRNFLDKEYF) panel (fields), which allows authorized users to translate (re-encrypt) field values to a new key for the specified Field Identifier.

6. Translate File Encryption Keys - Field Procedure (TRNFILKEYF)

Choose this option to open the Translate File Encryption Key - Field Procedure (TRNFILKEYF) panel, which allows authorized users to translate (re-encrypt) field values in a file (when a Pending Key has been entered) to the new pending Keys previously entered.

Field Triggers Menu

This menu allows you to add or remove field triggers.



How to Get There

On the Field Encryption Menu, choose option **21**, Field Triggers Menu. Or, execute the command **GO CRYPTO/CRYPTO8**.

Options

1. Remove Field Triggers (RMVFLDTRG)

Choose this option to open the Remove Field Triggers (RMVFLDTRG) panel, where you can remove any SQL triggers that were created by Powertech Encryption on the database file for the specified Field Identifier (when it was activated with ACTFLDENC).

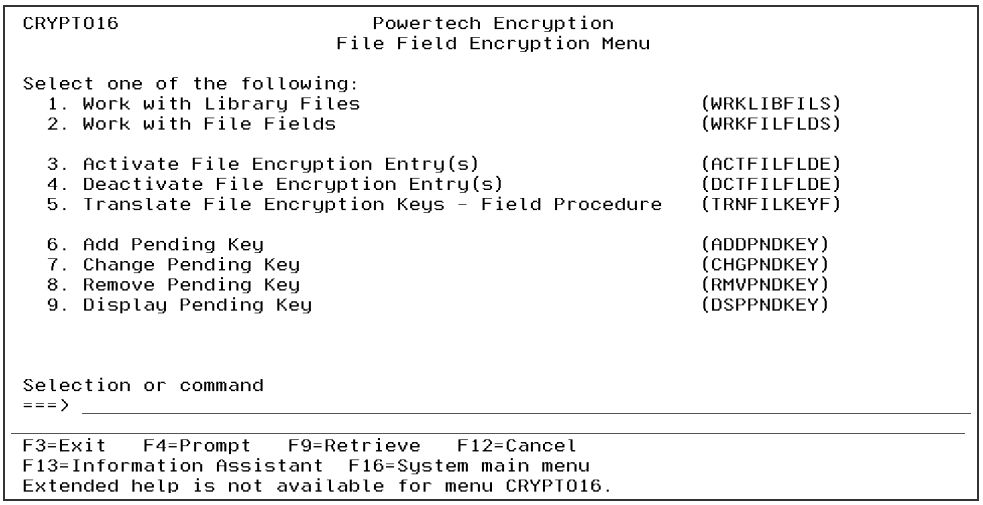
2. Add Field Triggers (ADDFLDTRG)

Choose this option to open the Add Field Triggers (ADDFLDTRG) panel, where you can recreate any SQL triggers that were removed with the RMVFLDTRG command.

WARNING:

- If the triggers are removed for a field, then any adds/updates of the field values WILL NOT be automatically encrypted by Powertech Encryption for IBM i.
- RMVFLDTRG should only be used to momentarily remove triggers before special database maintenance functions.
- ADDFLDTRG should only be used to recreate the triggers which were removed with the RMVFLDTRG command.

File Field Encryption Menu



How to Get There

From the Field Encryption Menu, choose option **30**. Or, submit the command **GO CRYPTO/CRYPTO16**.

Options

1. Work with Library Files (WRKLIBFILS)

Choose this option to open the Work with Files in Library (WRKLIBFILS) panel, which allows authorized users to work with files in a library and show which files may have encrypted fields.

2. Work with File Fields (WRKFILFLDS)

Choose this option to open the Work with File Fields (WRKFILFLDS) panel, which allows authorized users to work with fields in a file and the Field Encryption Registry.

3. Activate File Encryption Entry(s) (ACTFILFLDE)

Choose this option to open the Activate Field Entries (ACTFILFLDE) panel, which allows you to activate any *INACTIVE entries in the Field Encryption Registry for the file that use Field Procedures.

4. Deactivate File Encryption Entry(s) (DCTFILFLDE)

Choose this option to open the Deactivate File Encryption (DCTFILFLDE) panel, which allows you to activate any *ACTIVE entries in the Field Encryption Registry for the file that use Field Procedures.

5. Translate File Encryption Keys - Field Procedure (TRNFILKEYF)

Choose this option to open the Translate File Encryption Key - Field Procedure (TRNFILKEYF) panel allows authorized users to translate (re-encrypt) field values in a file (when a Pending Key has been entered) to the new pending Keys previously entered.

6. Add Pending Key (ADDPNDKEY)

Choose this option to open the Add a Field Encryption Pending Key (ADDPNDKEY) panel, where authorized users to add a Pending key to a Field entry that can be used in the next key rotation process.

7. Change Pending Key (CHGPNDKEY)

Choose this option to open the Change a Field Encryption Pending Key (CHGPNDKEY) panel, which allows authorized users to change a Pending key in a Field entry that can be used in the next key rotation process.

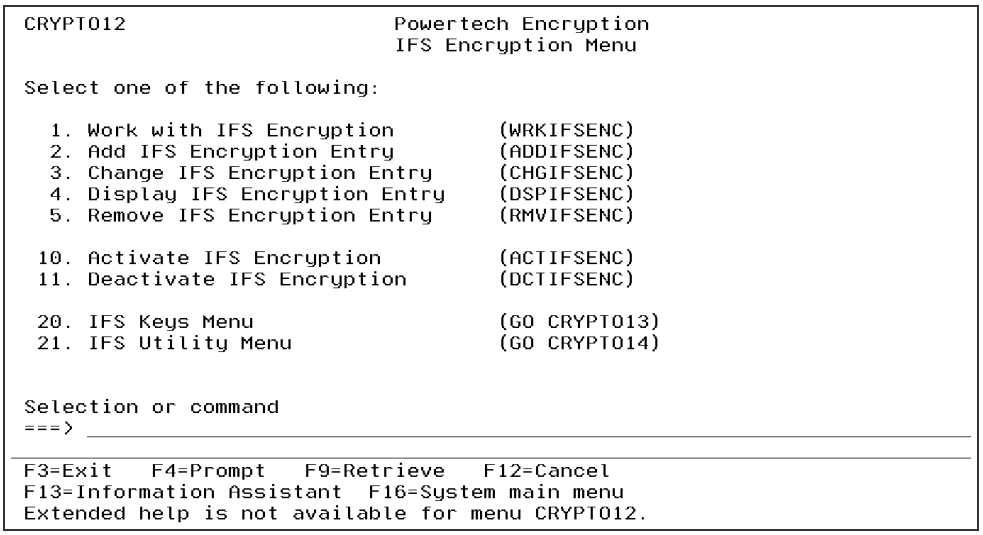
8. Remove Pending Key (RMVPNDKEY)

Choose this option to open the Remove a Field Encryption Pending Key (RMVPNDKEY) panel, which allows authorized users to remove an existing Pending key.

9. Display Pending Key (DSPPNDKEY)

Choose this option to open the Display a Field Encryption Pending Key (DSPPNDKEY) panel, which allows authorized users to display a Pending key in a Field entry that can be used in the next key rotation process.

IFS Encryption Menu



How to Get There

On the Main Menu, choose option **7**. Or, execute the command **GO CRYPTO/CRYPTO12**.

Options

1. Work with IFS Encryption (WRKIFSENC)

Choose this option to open the Work with IFS Encryption Registry (WRKIFSENC) panel, which allows authorized users to work with the entries in the IFS Encryption Registry.

2. Add IFS Encryption Entry (ADDIFSENC)

Choose this option to open the Add IFS Encryption Entry (ADDIFSENC) panel, which allows authorized users to add a new entry into the IFS Encryption Registry.

3. Change IFS Encryption Entry (CHGIFSENC)

Choose this option to open the Change IFS Encryption Entry (CHGIFSENC) panel, which allows authorized users to change an entry in the IFS Encryption Registry.

4. Display IFS Encryption Entry (DSPIFSENC)

Choose this option to open the Display IFS Encryption Entry (DSPIFSENC) panel, which allows authorized users to view an existing entry in the IFS Encryption Registry.

5. Remove IFS Encryption Entry (RMVIFSENC)

Choose this option to open the Remove IFS Encryption Entry (RMVIFSENC) panel, which allows authorized users to remove an existing entry from the IFS Encryption Registry.

10. Activate IFS Encryption (ACTIFSENC)

Choose this option to open the Activate IFS Encryption (ACTIFSENC) panel, which activates an *INACTIVE entry in the IFS Encryption Registry.

11. Deactivate IFS Encryption (DCTIFSENC)

Choose this option to open the Deactivate IFS Encryption (DCTIFSENC) panel, which allows authorized users to deactivate an *ACTIVE entry in the IFS Encryption Registry.

20. IFS Keys Menu (GO CRYPTO/CRYPTO13)

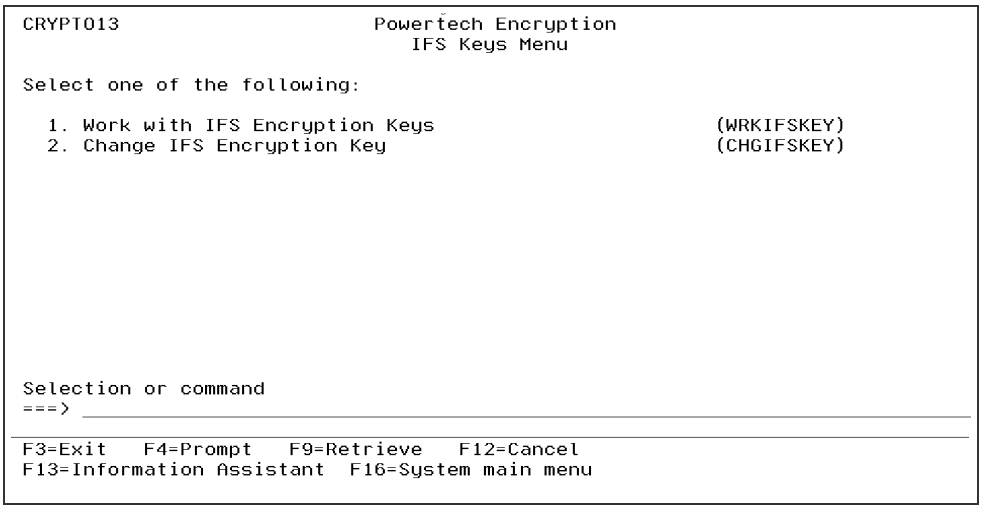
Choose this option to open the IFS Keys Menu.

21. IFS Utility Menu (GO CRYPTO/CRYPTO14)

Choose this option to open the IFS Utility Menu.

IFS Keys Menu

Use this menu to work with IFS Encryption Keys.



How to Get There

From the IFS Encryption Menu, choose option **20**. Or, submit the command **GO CRYPTO/CRYPTO13**.

Options

1. Work with IFS Encryption Keys (WRKIFSKEY)

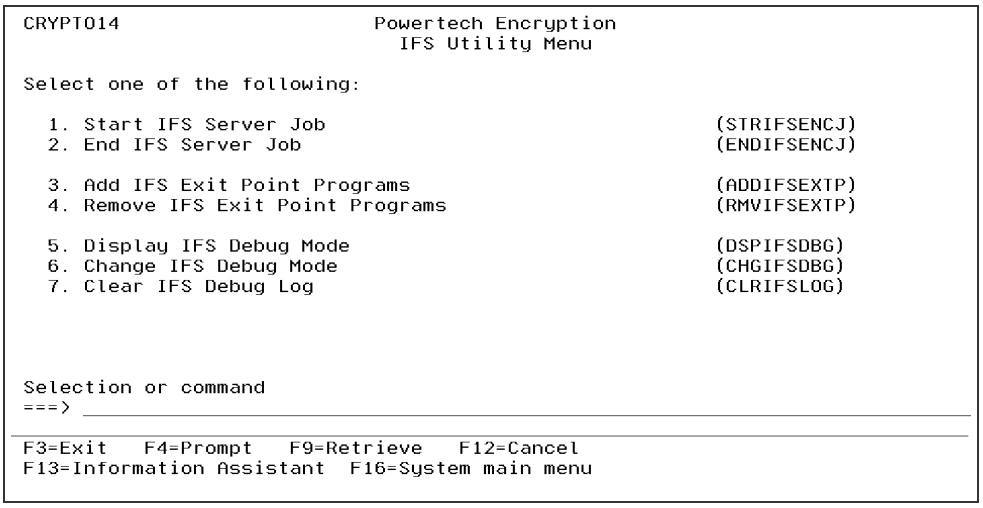
Choose this option to open the Work with IFS Encryption Keys (WRKIFSKEY) panel, which allows authorized users to display the keys used for an entry in the IFS encryption registry.

2. Change IFS Encryption Key (CHGIFSKEY)

Choose this option to open the Change IFS Encryption Key (CHGIFSKEY) panel, which allows authorized users to change (rotate) the keys used for an entry in the IFS Encryption Registry.

IFS Utility Menu

Use this menu to work with IFS Utilities.



How to Get There

From the IFS Encryption Menu, choose option **21**. Or, submit the command **GO CRYPTO/CRYPTO14**.

Options

1. Start IFS Server Job (STRIFSENCJ)

Choose this option to open the Start IFS Encryption Job (STRIFSENCJ) panel, which submits the IFSENCJOB server job to batch.

2. End IFS Server Job (ENDIFSENCJ)

Choose this option to open the End IFS Server Job (ENDIFSENCJ) panel, which notifies any IFSENCJOB jobs to end.

3. Add IFS Exit Point Programs (ADDIFSEXTP)

Choose this option to open the Add IFS Exit Point Programs panel, which adds the exit programs for the QIBM_QP0L_SCAN_CLOSE, QIBM_QP0L_SCAN_OPEN and QIBM_QPWFS_FILE_SERV exit points. These programs will capture IFS-related events in order to encrypt/decrypt the files as needed.

4. Remove IFS Exit Point Programs (RMVIFSEXTP)

Choose this option to open the Remove IFS Exit Point Programs panel, which removes the exit programs for the QIBM_QP0L_SCAN_CLOSE, QIBM_QP0L_SCAN_OPEN and QIBM_QPWFS_FILE_SERV exit points.

5. Display IFS Debug Mode (DSPIFSDBG)

Choose this option to open the Display IFS Debug Mode panel, which allows authorized users to view the Debug Mode.

6. Change IFS Debug Mode (CHGIFSDBG)

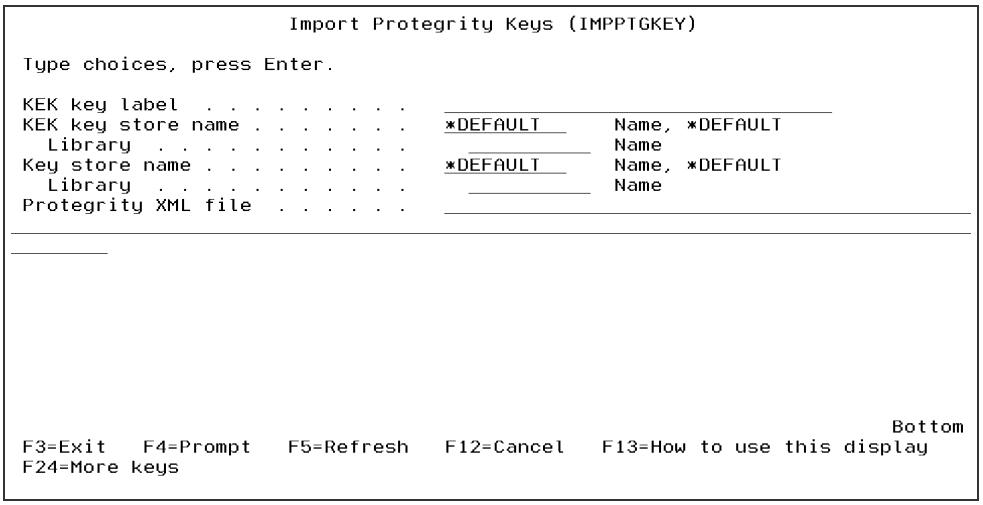
Choose this option to open the Change IFS Debug Mode panel, which allows authorized users to change the IFS encryption debug mode.

7. Clear IFS Debug Log (CLRIFSLOG)

Choose this option to open the Clear IFS Debug Log panel, which clears all records from the CRPFIFSLOG Log File.

Import Protegrity Key (IMPPTGKEY)

The IMPPTGKEY command allows authorized users to import Symmetric Keys that were generated by the Protegrity Defiance Enterprise Security Administrator (ESA). ESA is a separate licensed solution that allows for enterprise key management, which can be used to centrally manage keys and then serve those keys to multiple platforms (e.g. DB2, Oracle, SQL Server, etc.). See www.Protegrity.com to learn more about their ESA solution.



The following users can use the IMPPTGKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain DEKs” authority setting

The user must have *CHANGE authority to the Key Store Validation List (*VLDL) object into which the Key(s) will be imported and *USE authority to the library that contains the Key Store.

How to Get There

From the Symmetric Encryption Key Menu, choose option **20**.

Options

KEK key label (KEYLABEL)

Indicate the label of the Key Encryption Key (KEK) to use to decrypt Protegrity's Symmetric Keys that will be imported.

KEK key store name (KKEYSTR)

Indicate the object name and library of the Key Store which contains the Key Encryption Key (KEK).

The possible values are:

kek-key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

Key store name (KEYSTR)

Indicate the object name and library of the Key Store to store the imported Protegrity Symmetric Keys.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

XML File (XMLFILE)

Indicate the name and location (absolute path) of the XML file which contains the Protegrity Keys to import.

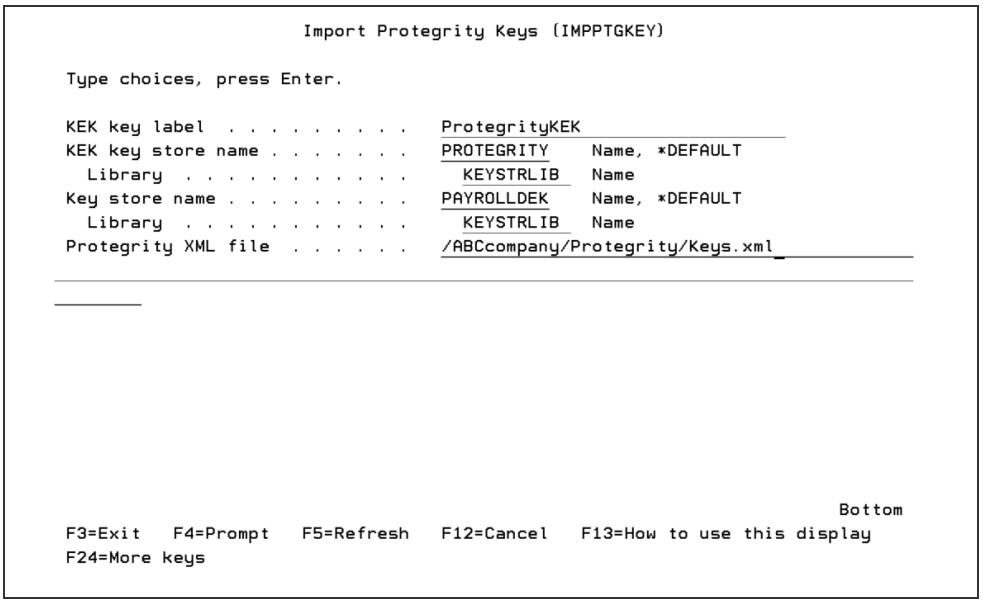
For instance: '/ABCcompany/Protegrity/Keys.xml'

The value for the XML file path can be up to 512 characters in length. The initial parameter size is 132, but can be expanded by placing an & in the first position.

EXAMPLE:

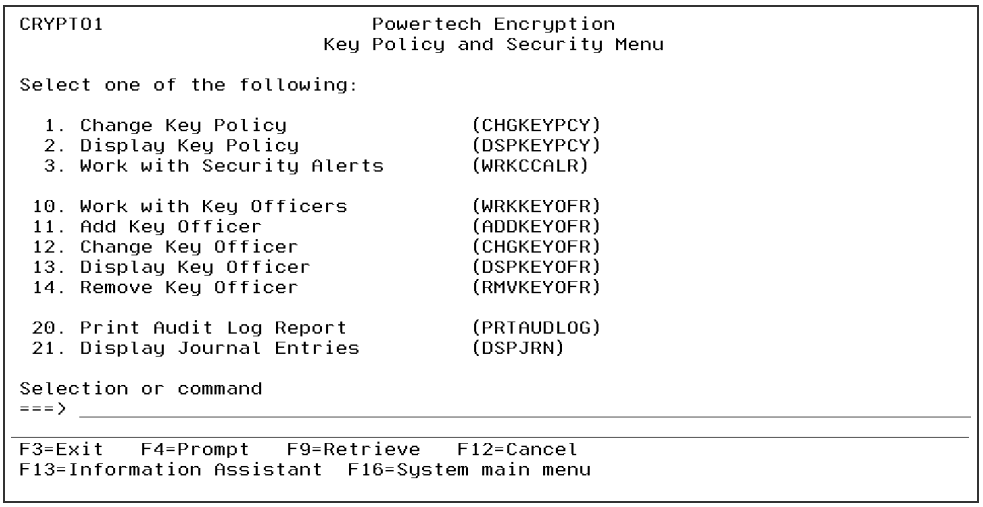
```
CRYPTO/CRTSYMKEY KEYLABEL(ProtegrityKEK) KEYSTR
```

```
(KEYSTRLIB/PROTEGRITY) +
                    ALGORITHM(*AES256) GENOPT(*PASS) PASSPHRASE
(Password) +
                    SALT(User) ITER(32000) ASCII(*YES)
```



Key Policy and Security Menu

The Key Policy and Security Menu allows you to control the environment settings for Powertech Encryption for IBM i's Key Management System. These settings are encrypted with the Product Encryption Key (PEK) and are stored in the CRYPTO library by default.



How to Get There

From the Main Menu, choose option **1**, Key Policy and Security Menu.

Options

1. Change Key Policy (CHGKEYPCY)

Choose this option to open the Change Key Policy (CHGKEYPCY) panel, where you can specify the policy settings for the Symmetric Key Management System.

2. Display Key Policy (DSPKEYPCY)

Choose this option to open the Display Key Policy (DSPKEYPCY) panel, where you can view the policy settings for the Symmetric Key Management System.

3. Work with Security Alerts (WRKCCALR)

Choose this option to open the Work with Security Alerts (WRKCCALR) panel, which allows authorized users to configure and view the Security Alert settings.

10. Work with Key Officers (WRKKEYOFR)

Choose this option to open the Work with Key Officers (WRKKEYOFR) panel, which allows authorized users to work with the Key Officers in the Symmetric Key environment.

11. Add Key Officer (ADDKEYOFR)

Choose this option to open the Add Key Officer (ADDKEYOFR) panel, which allows an authorized user to add a new Key Officer into the Symmetric Key Management System.

12. Change Key Officer (CHGKEYOFR)

Choose this option to open the Change Key Officer (CHGKEYOFR) panel, which allows an authorized user to change a Key Officer in the Symmetric Key Management System.

13. Display Key Officer (DSPKEYOFR)

Choose this option to open the Display Key Officer (DSPKEYOFR) panel, which allows an authorized user to view a Key Officer's authority settings.

14. Remove Key Officer (RMVKEYOFR)

Choose this option to open the Remove Key Officer (RMVKEYOFR) panel, which allows an authorized user to remove a Key Officer from the Symmetric Key Management System.

20. Print Audit Log Report (PRTAUDLOG)

Choose this option to open the Print Audit Log (PRTAUDLOG) panel, which allows authorized users to print the audit log.

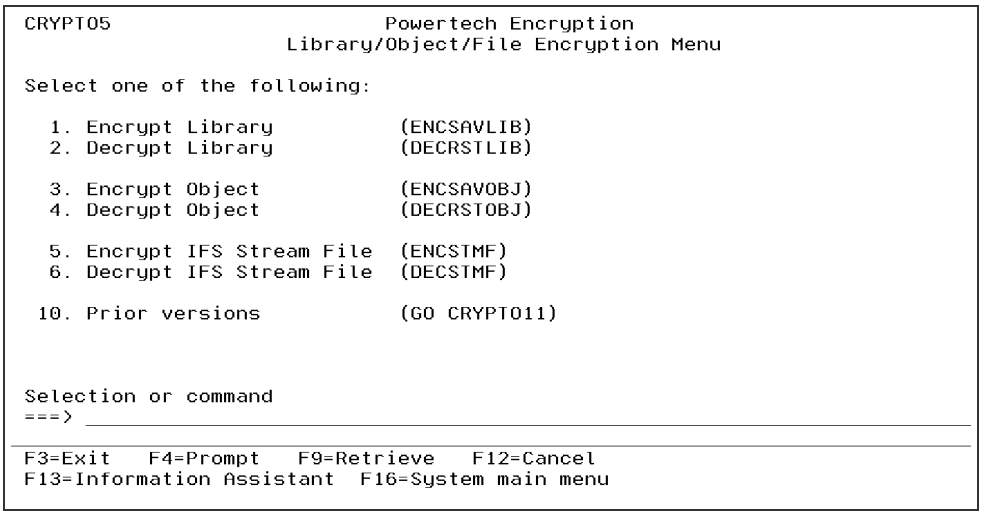
21. Display Journal Entries (DSPJRN)

Choose this option to open the Display Journal (DSPJRN) panel, where you can convert journal entries (contained in one or more receivers) into a form suitable for external representation.

Library/Object/File Encryption Menu

To view the list of available encryption and decryption commands, launch the Library/Object/File Encryption Menu by running the IBM i command of **GO CRYPTO/CRYPTO5**.

A command can be executed by selecting its corresponding menu option or by typing in the name of the command (listed in parenthesis).



How to Get There

From the Main Menu, choose option **5**, Library/Object/File Encryption Menu. Or, execute the command **GO CRYPTO/CRYPTO5**.

Options

1. Encrypt Library (ENCSAVLIB)

Choose this option to open the Encrypt Library (ENCSAVLIB) panel, which allows you to encrypt and save a copy of one or more libraries.

2. Decrypt Library (DECRSTLIB)

Choose this option to open the Decrypt Library (DECRSTLIB) panel, which allows you to restore and decrypt one or more libraries that were saved with the ENCSAVLIB command.

3. Encrypt Object (ENCSAVOBJ)

Choose this option to open the Encrypt Object (ENCSAVOBJ) panel, which allows you to encrypt and save a copy of one or more objects.

4. Decrypt Object (DECRSTOBJ)

Choose this option to open the Decrypt Object (DECRSTOBJ) panel, which allows you to restore and decrypt one or more objects that were saved with the ENCSAVLIB or ENCSAVOBJ commands.

5. Encrypt IFS Stream File (ENCSTMF)

Choose this option to open the Encrypt IFS Stream File (ENCSTMF) panel, which allows you to encrypt IFS stream files to a device (physical or virtual) or the IFS. Encryption algorithms provided are AES128, AES192 and AES256. Either a symmetric key or a password can be specified for the encryption.

6. Decrypt IFS Stream File (DECSTMF)

Choose this option to open the Decrypt IFS Stream File (DECSTMF) panel, which allows you to decrypt IFS stream files that were encrypted with the ENCSAVFIL command. Files can be

restored/decrypted from a device (physical or virtual) or the IFS. Either a symmetric key or a password can be specified for the decryption.

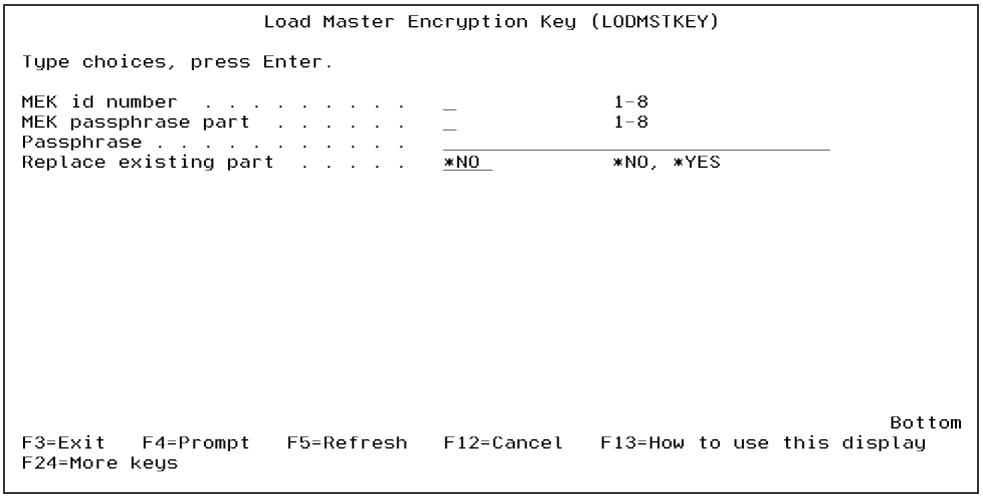
10. Prior versions (GO CRYPTO/CRYPTO11)

Choose this option to open the Legacy Encryption Commands panel. The legacy commands listed here have been replaced by faster commands, but can still be used to maintain compatibility with older processes. The latest library/object/file encryption commands are on the GO CRYPTO/CRYPTO5 menu.

Load Master Encryption Key (LODMSTKEY)

The LODMSTKEY command allows authorized users to specify the passphrase parts for a *NEW version of the Master Encryption Key (MEK).

See Preparing a Master Encryption Key (MEK) by Loading the Passphrase Parts in Getting Started.



NOTE: The *CURRENT or *OLD versions of the MEK will not be affected by LODMSTKEY command.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Load MEK passphrase parts” authority setting

The default Key Policy requires that each part of the passphrase is entered by a unique user profile.

How to Get There

From the Master Encryption Key Menu, choose option **1**, Load Master Encryption Key. Or, prompt (**F4**) the command **CRYPTO/LODMSTKEY**.

Options

MEK id number

The id number of the *NEW Master Encryption Key (MEK) that loads with a passphrase.

Possible values: 1-8

MEK passphrase part

The part of the passphrase entered.

Possible values: 1-8

Rules: The maximum parts (as defined in the Key Policy) cannot be exceeded for the MEK. The parts may be entered in different orders, for instance, part 3 can first be specified, then part 1, then part 2.

Passphrase

The passphrase being used.

Possible values: The passphrase can be up to 32 characters long.

Rules: The passphrase is case-sensitive and cannot be the same as a passphrase already entered on another part of the *NEW MEK.

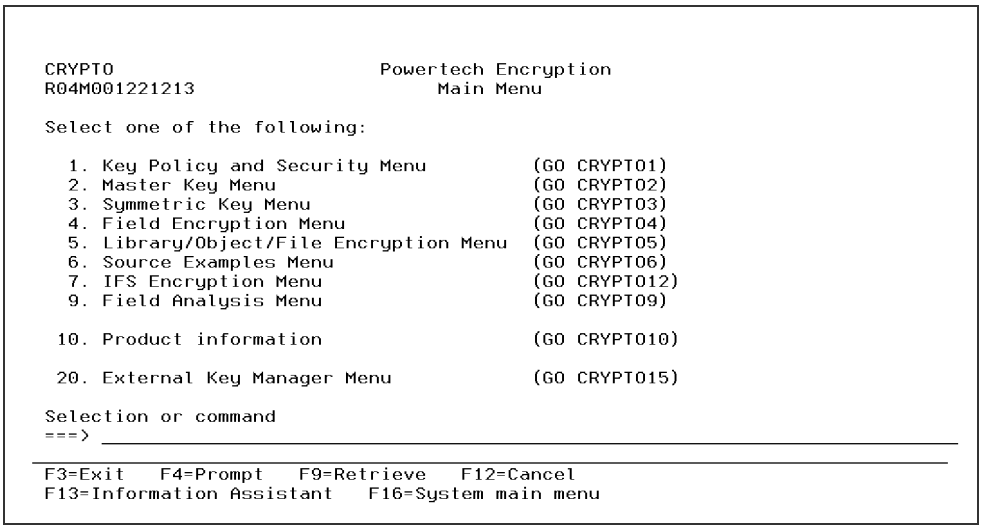
Replace existing part

Indicates whether the passphrase will replace an existing passphrase for the part specified. This is useful if the prior passphrase was entered incorrectly.

WARNING:
The passphrase parts used to load a MEK should be recorded in a safe place (not on the IBM i). An MEK will not be usable if it's copied or restored to another IBM i serial number. If you want to recreate the same MEK on another IBM i serial number (i.e. in a disaster recovery situation), these same passphrase parts will have to be re-entered (loaded) in the same order.

Main Menu

All of the Powertech Encryption for IBM i commands are accessible from a main menu and its submenus.



How to Get There

To access the Main Powertech Encryption for IBM i Menu, run the command **GO CRYPTO/CRYPTO**.

Commands can be executed from the menu by entering the corresponding menu option. A command can also be executed from the command line using its command name (in parentheses).

Options

1. Key Policy and Security Menu

Choose this option to open the Key Policy and Security Menu, where you can work with Key Policies, Key Officers, print audit log reports, and display journal entries.

2. Master Key Menu

Choose this option to open the Master Encryption Key Menu, where you can work with Master Encryption Keys.

3. Symmetric Key Menu

Choose this option to open the Symmetric Encryption Key Menu, which allows you to work with Symmetric Keys.

4. Field Encryption Menu

Choose this option to open the Field Encryption Menu, which allows you to work with Encrypted Fields.

5. Library/Object/File Encryption Menu

Choose this option to open the Library/Object/File Encryption Menu, which allows you to work with Library, Object, and File Encryption.

6. Source Examples Menu

Choose this option to open the Source Examples Menu, which allows you to see examples of RPGLE and CL source code.

7. IFS Encryption Menu

Choose this option to open the IFS Encryption Menu, which contains all the menu options to manage IFS encryption entries and related functionality.

9. Field Analysis Menu

Choose this option to open the Field Analysis Menu.

10. Product Information

Choose this option to open the Work with Misc Settings panel, where you can setup and run the IFS Encryption processes.

20. External Key Manager Menu

Choose this option to open the External Key Manager menu.

Function Keys

F3 (Exit): Ends the current task and returns to the display from which the task was started.
F4 (Prompt): Provides assistance in entering or selecting a command.
F9 (Retrieve): Displays the last command you ran from the command line, and any parameters you selected. By pressing this key once, you will see the last command you ran. By pressing this key twice, you will see the next-to-last command that you ran, and so on.
F12 (Cancel): Returns to the previous menu or display.
F13 (Information Assistant): Shows a menu with several types of assistance available.
F16 (System main menu): Goes to the system main menu.

Master Encryption Key Menu

A Master Encryption Key (MEK) is an AES 256 bit Symmetric Key used to protect (encrypt) the Data Encryption Keys (DEKs) contained in a Key Store. An organization can create up to 8 MEKs per environment on the IBM i. For instance, a MEK could be created to encrypt the Order Entry DEKs contained in a Key Store, and a second MEK could be created to encrypt the Payroll DEKs contained in another Key Store.

A MEK is generated by Powertech Encryption for IBM i using passphrases entered by designated Key Officers. Depending on the organization's key policy, up to 8 different passphrases can be required (by different users) in order to generate a MEK.

The Master Encryption Keys (MEK) are stored in a product-supplied validation list (*VLDL) object. The MEKs are encrypted with the Product Encryption Key (PEK).

Master Encryption Key (MEK) Versions

Each MEK can have up to three versions which are named *NEW, *CURRENT and *OLD:

*NEW Version

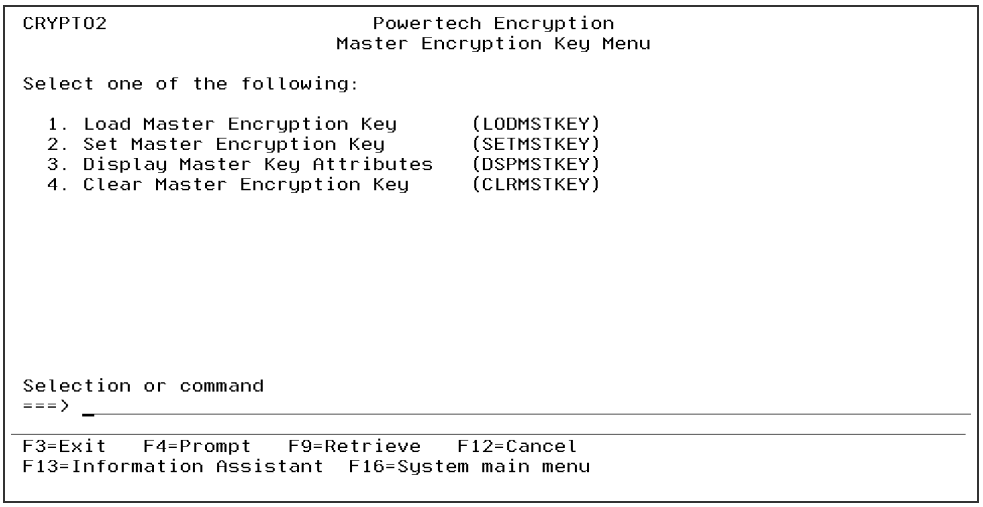
The *NEW version of a MEK is the version in which passphrases are being entered (loaded) by users with the LODMSTKEY (Load Master Key) command. The *NEW version cannot be used to encrypt DEKs within Key Stores. In order to convert the *NEW version into the *CURRENT version, an authorized Key Officer must set the Master Key using the CRYPTO/SETMSTKEY command.

*CURRENT Version

The *CURRENT version of a MEK is the current version which can be associated with Key Stores.

*OLD Version

The *OLD version of a MEK is the prior *CURRENT version of the MEK. The *OLD version cannot be associated with new Key Stores. However, DEKs in current Key Stores may still be encrypted under the *OLD version until they are translated (using the TRNKEYSTR command).



How to Get There

From the Main Menu, choose option **2**, Master Key Menu.

Options

1. Load Master Encryption Key (LODMSTKEY)

Choose this option to open the Load Master Encryption Key (LODMSTKEY) panel, where authorized users can specify the passphrase parts for a *NEW version of the Master Encryption Key (MEK).

2. Set Master Encryption Key (SETMSTKEY)

Choose this option to open the Set Master Key (SETMSTKEY) panel, which allows authorized users to set the Master Encryption Key (MEK).

3. Display Master Key Attributes (DSPMSTKEY)

Choose this option to open the Display Master Key Attributes (DSPMSTKEY) panel, which allows authorized users to display attributes for a Master Encryption Key (MEK).

4. Clear Master Encryption Key (CLRMSTKEY)

Choose this option to open the Clear Master Encryption Key (CLRMSTKEY) panel, which allows authorized users to clear a version of the Master Encryption Key (MEK).

Powertech Encryption License Setup

Use this panel to enter the information that allows you to use Powertech Encryption on your system.

System Serial Number

The serial number of the system where the product is installed.

System Model Number

The model number of the system where the product is installed.

Processor Feature Code

The processor feature code of the system where the product is installed.

Logical Partition

The partition number on which the product is installed.

ASP Group

The iASP Group on which the product is installed.

System Processors

The number of processors on your system.

On-Demand Processors

The number of On-Demand processors on your system.

Partition Processors

The number of processors on this partition.

OS Processors

The number of processors you are entitled to use with your copy of OS.

Shared Processors

The number of processors in the shared processor pool.

Licensed for

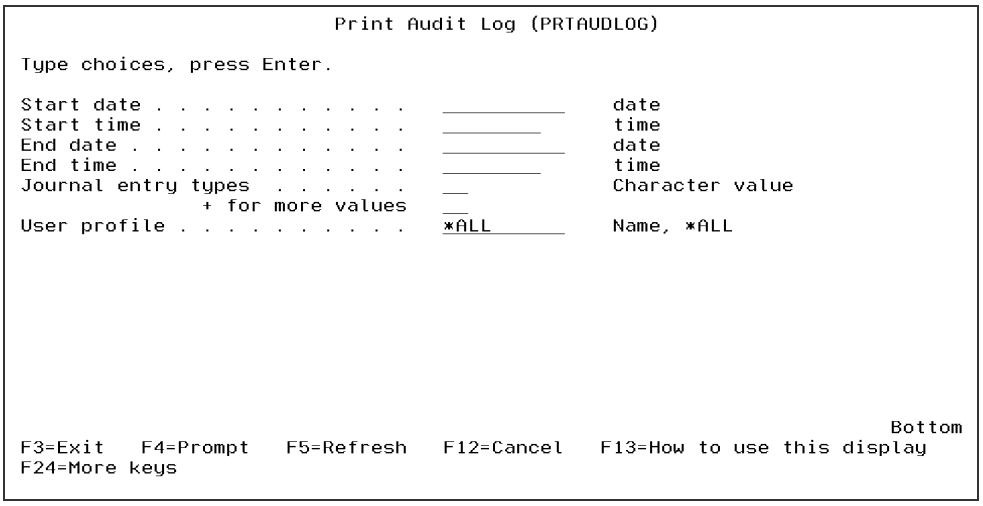
This is the number and type of processors for the code you have entered.

License Code

Enter the code exactly as it is printed. This is the code that you received from Fortra. Press function key 11 to toggle the License Code entry field between Cut and Paste Mode and Data Entry Mode. Using Cut and Paste Mode allows you to cut and paste the license code directly from the e-mail you received from Fortra onto your iSeries.

Print Audit Log (PRTAUDLOG)

The Print Audit Log (PRTAUDLOG) command allows authorized users to print the Powertech Encryption for IBM i audit log entries. This command provides selection criteria of date and time ranges, audit types and user IDs.



How to Get There

From the Key Policy and Security Menu, choose option **20**, Print Audit Log Report. Or, prompt (**F4**) the command **CRYPTO/PRTAUDLOG**.

Options

Start date (STRDATE)

Specify the starting date. All audit entries in the range from the starting date and time will be included in the report. The date should be specified in mmddyy or mmddccyy format, with or without date separators.

Start time (STRTIME)

Specify the starting time. All audit entries in the range from the starting date and time will be included in the report.

The time should be specified in 24-hour format, with or without a time separator.

End date (ENDDATE)

Specify the ending date. All audit entries in the range up to the ending date and time will be included in the report. The date should be specified in mmddyy or mmddccyy format, with or without date separators.

End time (ENDTIME)

Specify the ending time. All audit entries in the range up to the ending date and time will be included in the report.

The time should be specified in 24-hour format, with or without a time separator.

Journal entry type (JRNTYPE)

If the journal entry type parameter is left blank, then all Powertech Encryption for IBM i audit entries will be selected. Otherwise, if any journal entry types are entered, then only the entries which match those journal entry types will be selected.

You can specify up to 25 journal entry types to filter on.

Below is a list of Powertech Encryption for IBM i journal entry types.

01 - Key Policy setting(s) changed
02 - Key Officer added
03 - Key Officer changed
04 - Key Officer removed
05 - Master Key passphrase part loaded
06 - Master Key was Set
07 - Master Key cleared
08 - Key Store created
09 - Key Store translated
10 - Symmetric Key created
11 - Symmetric Key changed
12 - Symmetric Key copied
13 - Symmetric Key deleted
14 - Field Encryption Registry - Entry added
15 - Field Encryption Registry - Encryption Key changed
16 - Field Encryption Registry - Entry removed
17 - Field Encryption Registry - Entry activated
18 - Field Encryption Registry - Entry changed
19 - Field Encryption Registry - Entry deactivated
21 - Symmetric Key exported
22 - Field Encryption Registry - Unable to Activate Entry
23 - Field Encryption Registry - Unable to Deactivate Entry
24 - Field Encryption Registry - Entry copied
25 - Field Encryption Registry - SQL Triggers added to file
26 - Field Encryption Registry - SQL Triggers removed from fil
27 - Field Encryption Registry - Field keys translated
30 - Unable to encrypt/decrypt field using stored procedure
31 - Trigger exit program - Error occurred or return code of 'E'rror
32 - Trigger exit program - Return code of 'I'gnore
33 - Trigger exit program - Return code of 'P'rocess with message
34 - Unable to send Security Alert
35 - Alert added
36 - Alert changed

37 - Alert deleted
40 - Data encrypted with Key that requires logging
41 - Data decrypted with Key that requires logging
50 - Authority error

User profile (USERPRF)

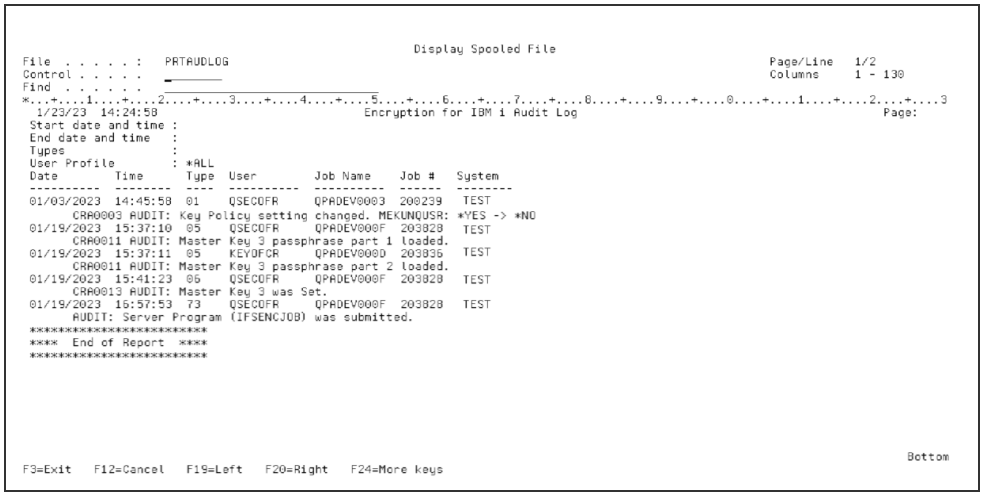
Specify the user profile(s) to select in the audit log.

The possible values are:

user-profile-name Specify the user profile name to select.
***ALL** All users will be selected.

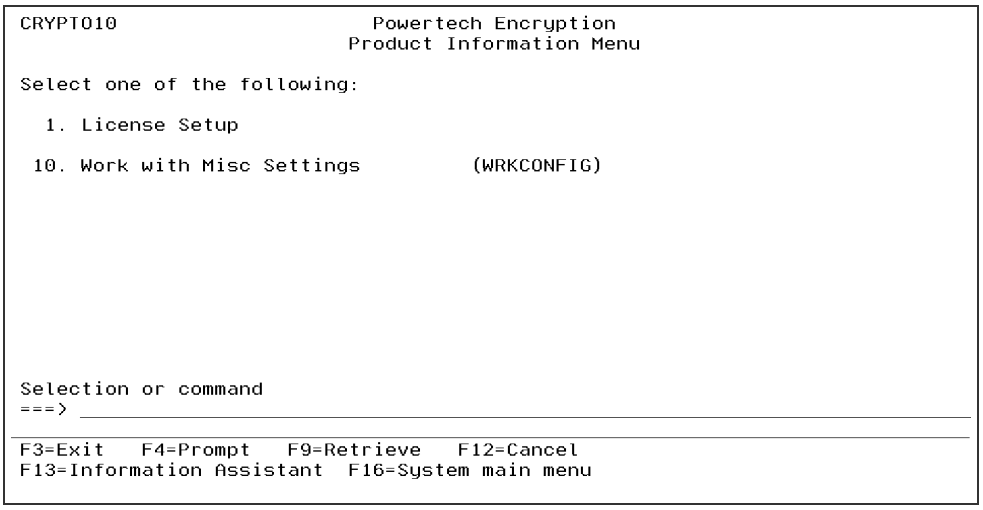
Example

A formatted report will be generated with the audit log entries. For each audit entry printed, it will include the audit date, time, user, job name, job number, audit type and message.



Product Information Menu

This menu allows you to work with licensing and product settings.



How to Get There

On the Main Menu, choose option **10**. Or, execute the command **GO CRYPTO/CRYPTO10**.

Options

1. License Setup

Choose this option to open the Powertech Encryption License Setup panel, which allows you to enter the information that allows you to use Powertech Encryption for IBM i on your system.

10. Work with Misc Settings (WRKCONFIG)

Choose this option to open the Work with Misc Settings panel, where you can set up and run the IFS Encryption processes.

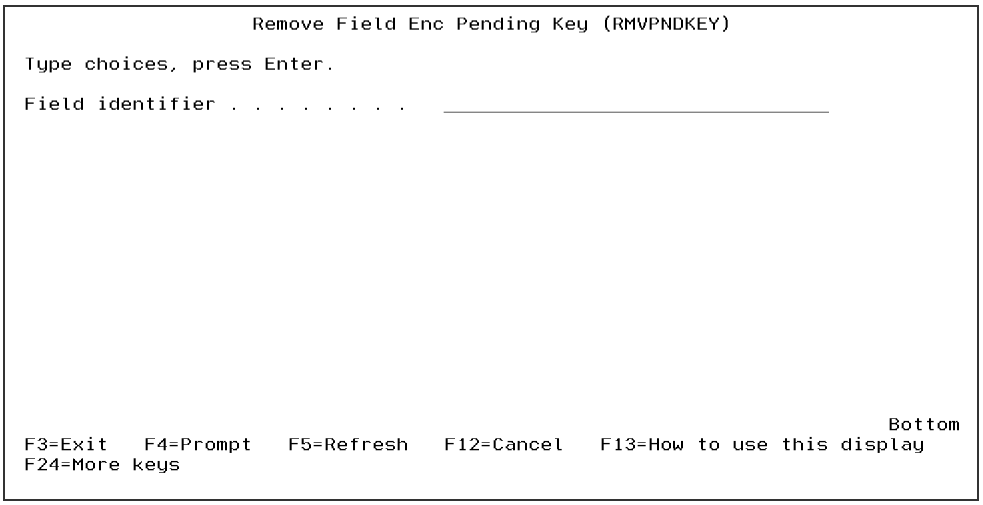
Remove a Field Encryption Pending Key (RMVPNDKEY)

The Remove a Field Encryption Pending Key (RMVPNDKEY) command allows authorized users to remove an existing Pending key.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.



How to Get There

In the File Field Encryption Menu, choose option **8**.

Options

Field identifier (FLDID)

Indicate the unique name of the field entry to remove the Pending key.

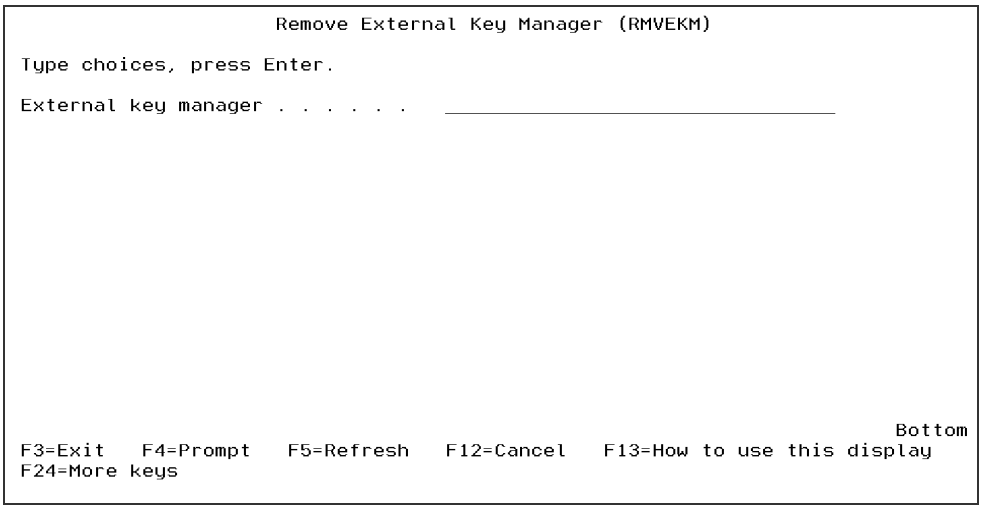
Remove External Key Manager (RMVEKM)

The RMVEKM command allows an authorized user to remove an External Key Manager entry from the environment.

> **NOTE:** Any maintenance to the External Key Managers is logged into an audit file.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key policy and alerts” authority setting



How to Get There

In the External Key Manager Menu, choose option **5**. Or, from the Work with External Key Managers panel, choose option **4** for a Key Manager. Or, submit the command **RMVEKM**.

Field Descriptions

Manager id (EKMGRID)

Indicate the unique name of the external key manager entry.

Remove Field Encryption Entry (RMVFLDENC)

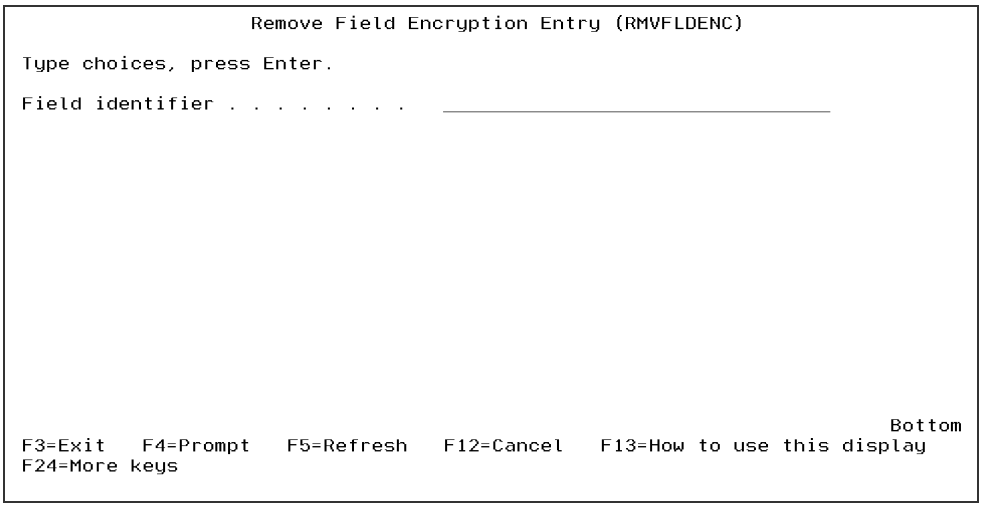
The RMVFLDENC command allows authorized users to remove an *INACTIVE field entry from the Encryption Registry.

> **NOTE:** The RMVFLDENC command will only remove the field entry from the Encryption Registry. It will not cause any action to be performed on the actual database field in the file.

The following users can use the RMVFLDENC command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain Field Enc. Registry” authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object, which contains the Field Encryption Registry.



Do the following steps to remove a field entry from the Encryption Registry:

1. Prompt (**F4**) the command **CRYPTO/RMVFLDENC**.
2. Enter the Field identifier to remove, then press Enter.

Remove Field Triggers (RMVFLDTRG)

The Remove Field Triggers (RMVFLDTRG) command will remove any SQL triggers that were created by Powertech Encryption on the database file for the specified Field Identifier (when it was activated with ACTFLDENC).

RMVFLDTRG is useful for temporarily removing the triggers when special maintenance operations need to be performed on the database file, such as adding a new field to the file through a change management tool.

RMVFLDTRG can only be used for *ACTIVE field entries which were configured in the Registry to use SQL triggers.

NOTE:

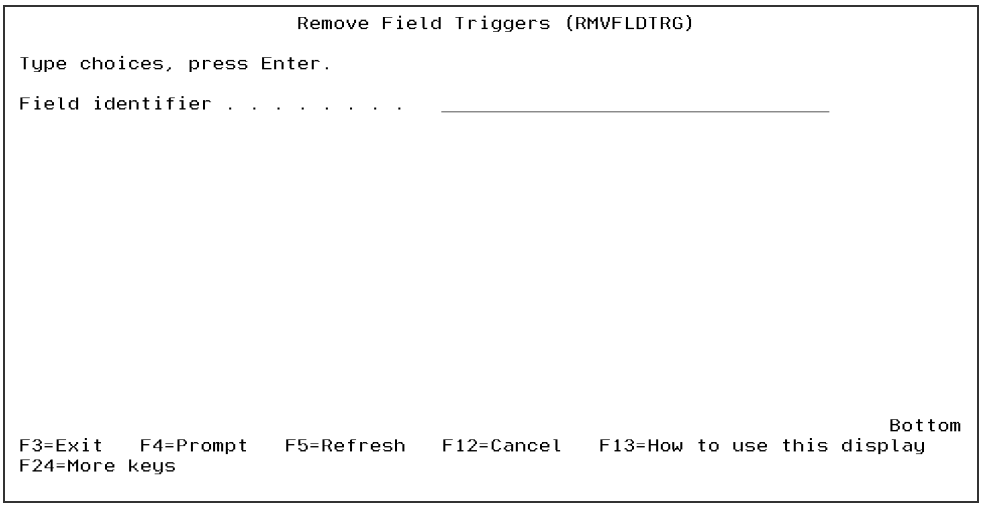
- You should not allow other users or applications to access the database file while the SQL Triggers are removed. Otherwise, field values may be added/updated in the file in unencrypted format.
- RMVFLDTRG only removes the SQL Triggers from the database file. Any field values, which existed before RMVFLDTRG was executed, will remain encrypted. The field entry will keep an *ACTIVE status in the Field Encryption Registry.
- If you want to decrypt the existing field values and remove the SQL triggers, then the DCTFLDENC (Deactive Field Encryption) command should be used.
- Use the ADDFLDTRG command to recreate the SQL Triggers on the database file.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have the following object authorities:

- *USE authority for the CRVL002 *VLDL object which contains the Field Encryption Registry.
- Alter authority for the database file specified on the field entry.



NOTE: Use the ADDFLDTRG command to recreate the SQL Triggers on the database file.

How to Get There

From the Field Triggers Menu, choose option **1**, Remove Field Triggers. Or, execute the command **RMVFLDTRG**.

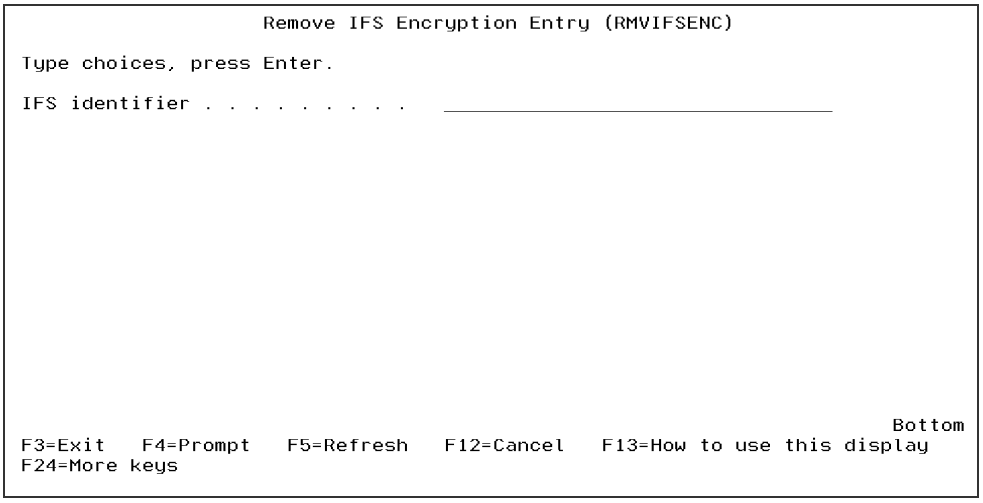
Options

Field identifier (FLDID)

Enter the Field identifier to remove the SQL triggers for.

Remove IFS Encryption Entry (RMVIFSENC)

The Remove IFS Encryption Entry (RMVIFSENC) command allows authorized users to remove an existing entry from the IFS Encryption Registry.



This command is only allowed for removing an entry with an *INACTIVE status.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

This command requires the user to have *CHANGE authority to the CRVL003 Validation List (*VLDL) object which contains the IFS Encryption Registry.

NOTE: The RMVIFSENC command only removes the entry from the registry. It will not cause any action to be performed on the actual IFS files.

How to Get There

From the IFS Encryption Menu, choose option **5**. Or, prompt (**F4**) the command **CRYPTO/RMVIFSENC**.

Options

IFS identifier (IFSID)

Indicate the unique name of the entry to remove.

Remove IFS Exit Point Programs (RMVIFSEXTP)

The Remove Exit Point Programs (RMVIFSEXTP) command will remove the exit programs for the QIBM_QP0L_SCAN_CLOSE, QIBM_QP0L_SCAN_OPEN and QIBM_QPWFS_FILE_SERV exit points.

This command requires that you have *ALLOBJ (all object) and *SECADM (security administrator) special authorities to remove exit programs from the registration facility.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

IMPORTANT: Fortra has integrated exit point sharing among select Fortra IBM i products. You will continue to see exit programs remain on the exit points, even after issuing the RMVIFSEXTP command. You do not need to restart QSERVER jobs to recognize the removal of exit programs. See Getting Started with IFS Encryption.

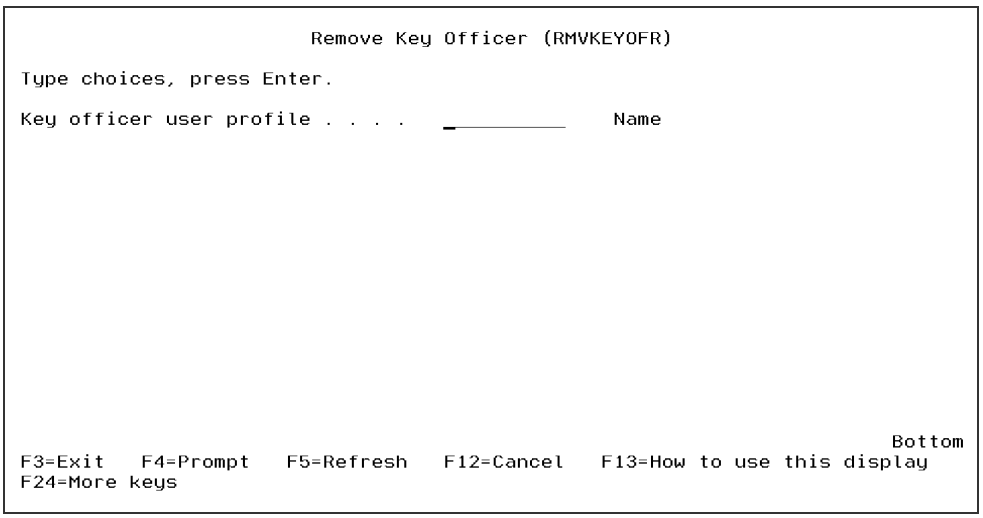
How to Get There

From the IFS Utility Menu, choose option **4**, Remove IFS Exit Point Programs. Or, prompt (**F4**) the command **CRYPTO/RMVIFSEXTP**.

Remove Key Officer (RMVKEYOFR)

The RMVKEYOFR command allows an authorized user to remove a Key Officer from the Symmetric Key environment.

> **NOTE:** Any maintenance to the Key Officers is logged into an audit file.



The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key officers” authority setting

The RMVKEYOFR will not delete the actual user profile. It will only remove this user profile as a Key Officer from the Key Management System.

How to Get There

From the Key Policy and Security Menu, choose option **14**, Remove Key Officer. Or, prompt (**F4**) the command **CRYPTO/RMVKEYOFR**.

Field Description

Key officer user profile

Specify the Key Officer’s user profile on the IBM i.

Set Master Encryption Key (SETMSTKEY)

The SETMSTKEY command performs the following:

- The CRVL001 validation list (*VLDL) object, which contains the encrypted MEKs, is backed up into a Save File object (sequentially named).
- The *NEW version of the MEK is generated (using all the passphrase parts entered)
- The *OLD version of the MEK is cleared
- The *CURRENT version of the MEK is copied into the *OLD version of the MEK
- The *NEW version of the MEK is copied into the *CURRENT version of the MEK

WARNING: The SETMSTKEY command will replace the *OLD version of the MEK with the *CURRENT version. Before running this command, you should first use the TRNKEYSTR command to translate (re-encrypt) any DEKs in the Key Stores which are still encrypted with the *OLD version of the MEK.

See also Generating the MEK using the Loaded Passphrase Parts in Getting Started.

```
                    Set Master Encryption Key (SETMSTKEY)

Type choices, press Enter.

MEK id number  . . . . . .  _            1-8

                                                                  Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

The following users can use the SETMSTKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Set and clear MEKs” authority setting

How to Get There

From the Master Encryption Key Menu, choose option **2**, Set Master Encryption Key. Or, prompt (**F4**) the command **CRYPTO/SETMSTKEY**.

Options

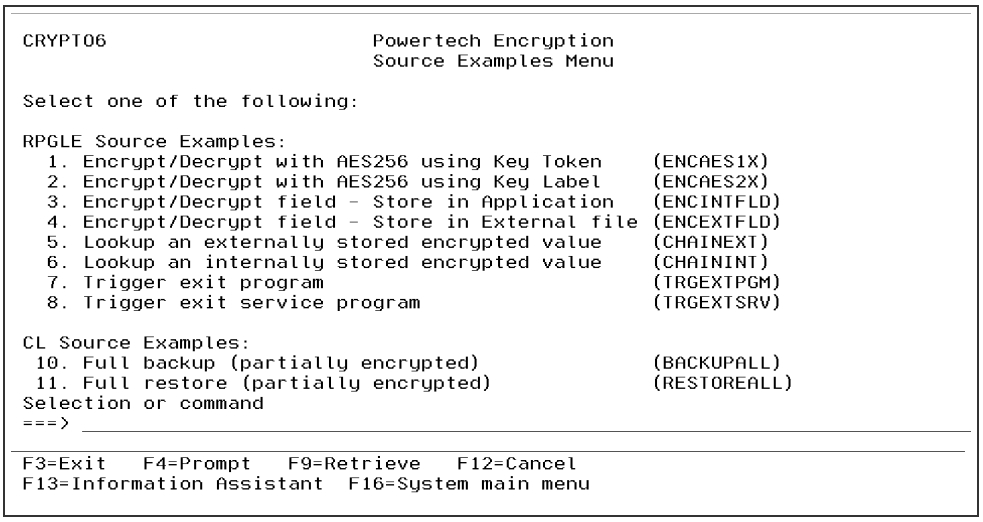
MEK id number (MEKID)

Indicate the id number of the Master Encryption Key (MEK) to set.

> **NOTE:** After running SETMSTKEY… If existing DEKs in Key Stores are encrypted with the MEK, then you should execute the TRNKEYSTR command to translate (re-encrypt) the DEKs in the Key Stores.

Source Examples Menu

This menu allows you to see examples of RPGLE and CL source code.



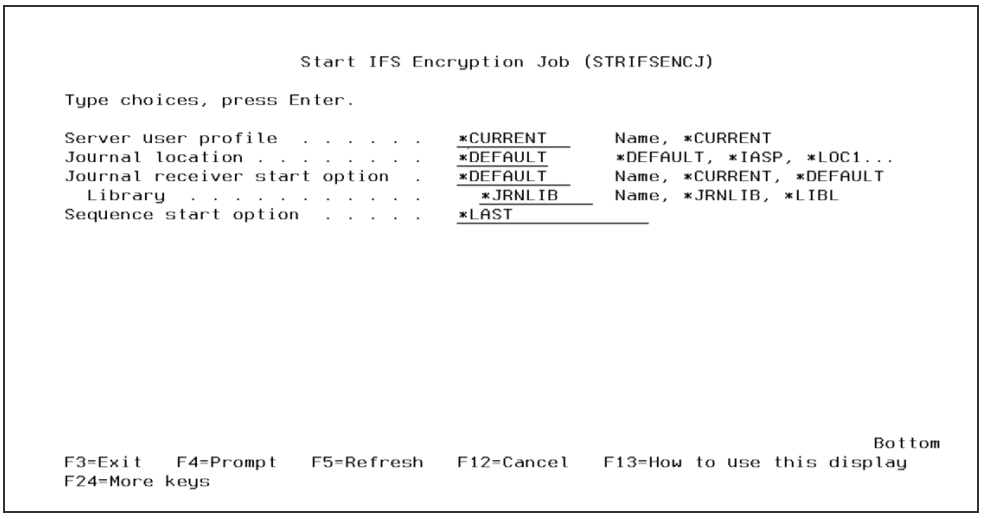
How to Get There

From the Main Menu, choose option **6**, Source Examples Menu. Or, execute the command **GO CRYPTO/CRYPTO6**.

Start IFS Encryption Job (STRIFSENCJ)

The Start IFS Server Job (STRIFSENCJ) command will submit the IFSENCJOB server job to batch. This job will monitor system journals for changes made to objects within IFS folders that are targeted for encryption.

This command will use the CRYPTO Job Description in the CRYPTO Library for submitting the IFSENCJOB.



The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

How to Get There

From the IFS Utility Menu, choose option **1**, Start IFS Server Job. Or, prompt (**F4**) the command **CRYPTO/STRIFSENCJ**.

Options

Server user profile (SRVUSRPRF)

The user profile under which the IFSENCJOB server job will run.

This user profile should have the following authorities.

- *ALL authority to the Directory(s) to encrypt and all the files in the directory(s).
- *ALL authority to the Directory(s) that hold the encrypted files and the files in the directory(s).
- *CHANGE authority to the files CRPFIFS, CRPFIFSL1, CRPFIFSL2, CRPFIFSL3, CRPFIFSL4, CRPFIFS2, CRPFIFSLOG Files which are located in the CRYPTO library.
- *CHANGE authority to the data areas CRDEBUG, CRLSTSEQ and CRSRVRUN, which are located in the CRYPTO library.
- *Use Authority to the CRJNI001 Journal and all Journal Receivers, which are located in the CRYPTO library.
- *USE authority to the CRCL414 and CRRP040 programs in the CRYPTO Library

The possible values are:

user-profile-name Specify a valid user profile name to run the IFSENCJOB server job under. This user profile must be enabled.
***CURRENT** The current job's user profile is used to run the IFSENCJOB server job.

Journal location (JRNLOC)

Indicate which Encryption job to start. IFSENCJOB, IFSENCJOBA, IFSENCJOB1, IFSENCJOB2, IFSENCJOB3, IFSENCJOB4 or IFSENCJOB5.

The possible values are:

***DEFAULT** The server job (IFSENCJOB) which runs over the journal in the CRYPTO Library will be started. All entries in the IFS Encryption Registry that use *DEFAULT for the Journal Loc will use this job (IFSENCJOB).

***IASP** The server job (IFSENCJOBA) which runs over the journal in the Journal Library defined for *IASP will be started. All entries in the IFS Encryption Registry that use *IASP for the Journal Loc will use this job (IFSENCJOBA).

***LOC1** The server job (IFSENCJOB1) which runs over the journal in the Journal Library defined for *LOC1 will be started. All entries in the IFS Encryption Registry that use *LOC1 for the Journal Loc will use this job (IFSENCJOB1).

***LOC2** The server job (IFSENCJOB2) which runs over the journal in the Journal Library defined for *LOC2 will be started. All entries in the IFS Encryption Registry that use *LOC2 for the Journal Loc will use this job (IFSENCJOB2).

***LOC3** The server job (IFSENCJOB3) which runs over the journal in the Journal Library defined for *LOC3 will be started. All entries in the IFS Encryption Registry that use *LOC3 for the Journal Loc will use this job (IFSENCJOB3).

***LOC4** The server job (IFSENCJOB4) which runs over the journal in the Journal Library defined for *LOC4 will be started. All entries in the IFS Encryption Registry that use *LOC4 for the Journal Loc will use this job (IFSENCJOB4).

***LOC5** The server job (IFSENCJOB5) which runs over the journal in the Journal Library defined for *LOC5 will be started. All entries in the IFS Encryption Registry that use *LOC5 for the Journal Loc will use this job (IFSENCJOB5).

Journal receiver start option (RCVSTROPT)

The receiver start option to use for reading the journal.

The possible values are:

journal-receiver-name Use the journal receiver name that is entered. When a name is entered the SEQSTROPT parameter is used to determine the sequence number to use. This should only be used when it is ok to skip journal records. An example of a time when it is ok to skip records would be when switching to another machine (high availability) and all the records that would be skipped have been processed on the machine you are switching from.

***DEFAULT** Use the values in the CRLSTSEQ data area for the journal receiver and the last sequence number.

***CURRENT** Use the current journal receiver that is attached to the Journal. When *CURRENT is used the SEQSTROPT parameter is used to determine the sequence number to use. This should only be used when it is ok to skip journal records. An example of a time when it is ok to skip records would be when switching to another machine (high availability) and all the records that would be skipped have been processed on the machine you are switching from.

The possible library values are:

library-name Enter the name of the library where the journal receiver is located.
***JRNLIB** Locate the journal receiver in the Journal library.
***LIBL** Locate the journal receiver within the library list.

Sequence start option (SEQSTROPT)

The sequence number to set in the data area to journal records. This parameter is only used when the RCVSEQOPT parameter is set to journal receiver name or *CURRENT.

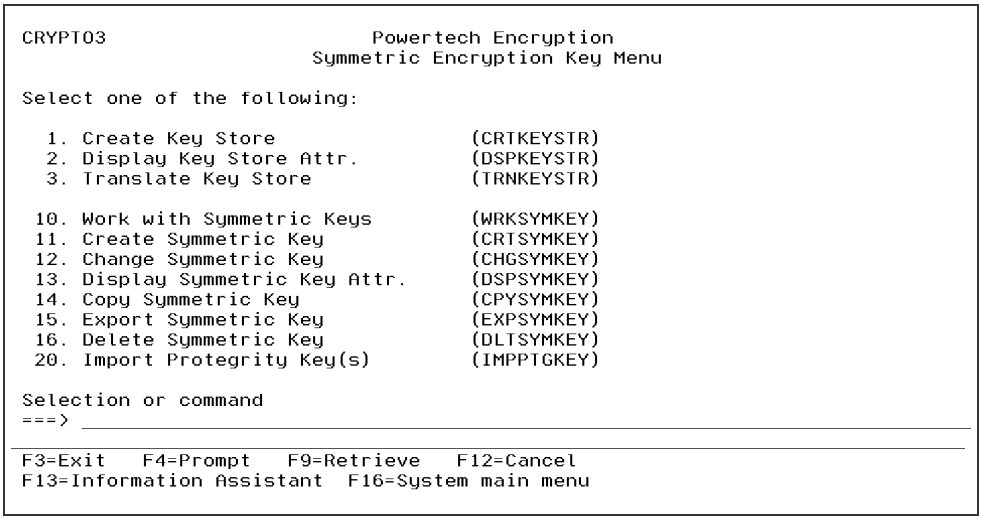
The possible values are:

sequence-number A sequence number that exists in the journal receiver entered in the RCVSTROPT parameter.

***FIRST** Uses the first sequence number found in the journal receiver entered in RCVSTROPT parameter.
***LAST** Uses the last sequence number found in the journal receiver entered in RCVSTROPT parameter.

Symmetric Encryption Key Menu

Use this menu to work with Symmetric Encryption Keys.



How to Get There

From the Main Menu, choose option **3**, Symmetric Encryption Menu. Or, submit the command **GO CRYPTO/CRYPTO3**.

Options

1. Create Key Store (CRTKEYSTR)

Choose this option to open the Create Key Store (CRTKEYSTR) panel, which allows authorized users to create a Key Store for containing Symmetric Keys.

2. Display Key Store Attr. (DSPKEYSTR)

Choose this option to open the Display Key Store Attributes (DSPKEYSTR) panel, which allows authorized users to display the attributes for a Key Store.

3. Translate Key Store (TRNKEYSTR)

Choose this option to open the Translate Key Store (TRNKEYSTR) panel, which allows authorized users to translate (re-encrypt) the Symmetric Keys within a Key Store to the *CURRENT version of a Master Encryption Key (MEK).

10. Work with Symmetric Keys (WRKSYMKEY)

Choose this option to open the Work with Symmetric Keys (WRKSYMKEY) panel, which allows authorized users to display the attributes for a Key Store.

11. Create Symmetric Key (CRTSYMKEY)

Choose this option to open the Create Symmetric Key (CRTSYMKEY) panel, which allows authorized users to create a Symmetric Key (also known as Data Encryption Key) and place it into a Key Store.

12. Change Symmetric Key (CHGSYMKEY)

Choose this option to open the Change Symmetric Key (CHGSYMKEY) panel, which allows authorized users to display the attributes for a Key Store.

13. Display Symmetric Key Attr. (DSPSYMKEY)

Choose this option to open the Display Symmetric Key Attributes (DSPSYMKEY) panel, which allows authorized users to display the attributes for a Key Store.

14. Copy Symmetric Key (CPYSYMKEY)

Choose this option to open the Copy Symmetric Key (CPYSYMKEY) panel, which allows authorized users to copy Symmetric Keys between Key Stores.

15. Export Symmetric Key (EXPSYMKEY)

Choose this option to open the Export Symmetric Key (EXPSYMKEY) panel, which allows authorized users to display the actual value of a Symmetric Key contained within a Key Store.

16. Delete Symmetric Key (DLTSYMKEY)

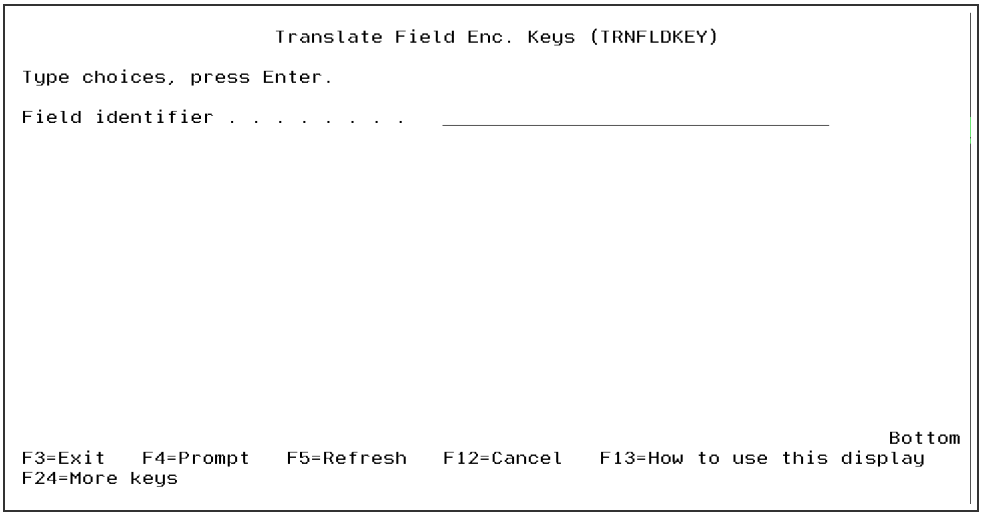
Choose this option to open the Delete Symmetric Key (DLTSYMKEY) panel, which allows authorized users to remove a Symmetric Key from a Key Store.

20. Import Protegrity Key(s) (IMPPTGKEY)

Choose this option to open the Import Protegrity Key (IMPPTGKEY) panel, which allows authorized users to remove a Symmetric Key from a Key Store.

Translate Field Encryption Keys – External Storage (TRNFLDKEY)

The Translate Field Keys (TRNFLDKEY) command allows authorized users to translate (re-encrypt) any field values, which were encrypted under older Keys, up to the most current Key for the specified Field Identifier.



NOTE:

- TRNFLDKEY can be used for *ACTIVE field entries which store the encrypted field values in an external file.
- It is recommended to submit this command to batch using the SBMJOB command.
- TRNFLDKEY can be executed while users and applications are active on the system.
- The execution time for TRNFLDKEY depends on the number of records which it must translate (re-encrypt) to the current Key.

TRNFLDKEY will find any records in the external file which are encrypted under old keys. For each record found, TRNFLDKEY will decrypt the value with the old key and re-encrypt the value using the current key.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have the following object authorities:

- *USE authority for the CRVL002 *VLDL object which contains the Field Encryption Registry.
- *READ authority for the database file specified on the field entry.
- *CHANGE authority for the external file which contains the encrypted values.

> **NOTE:**
> • TRNFLDKEY can be executed while users and applications are active on the system.
> • The execution time for TRNFLDKEY depends on the number of records which it must translate (re-encrypt) to the current Key.

How to Get There

In the Field Keys Menu, choose option **3**.

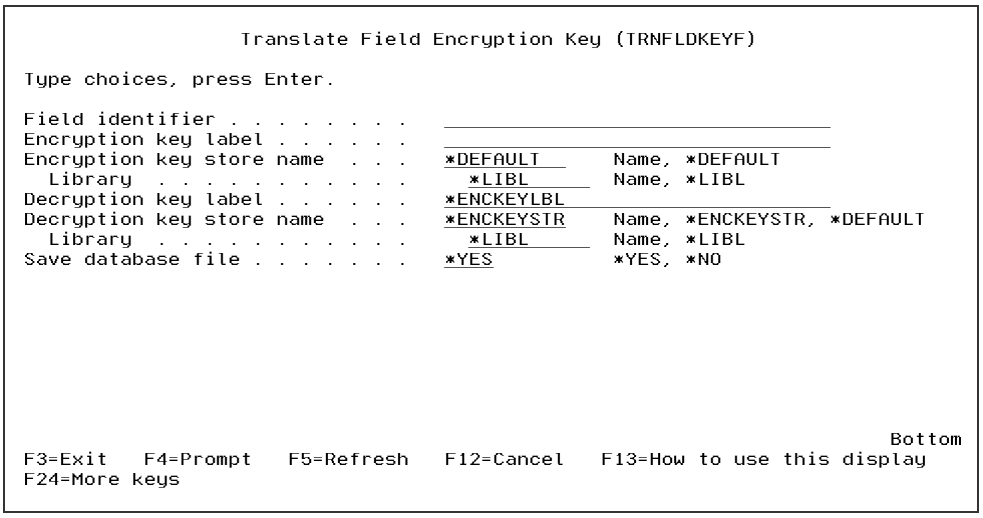
Options

Field identifier (FLDID)

Specify the Field identifier to translate the field keys for.

Translate Field Encryption Key – Field Procedure (TRNFLDKEYF)

The Translate Field Encryption Key - Field Procedure (TRNFLDKEYF) command allows authorized users to translate (re-encrypt) field values to a new key for the specified Field Identifier.



NOTE:

- TRNFLDKEYF can be used for *ACTIVE field entries which use a DB2 Field Procedure for automatic encryption/decryption.
- It is recommended to submit this command to batch using the SBMJOB command.
- TRNFLDKEYF will attempt to get an exclusive lock on the file, so no users or applications should be using the file at the time.
- The execution time for TRNFLDKEYF depends on the number of records which it must translate (re-encrypt) to the new Key.

For each record found, TRNFLDKEYF will decrypt the value with the old key and re-encrypt the value using the new key specified.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

This command requires that you have *USE authority for the Key Stores that contain both the current and new encryption/decryption Keys.

IMPORTANT: Before using the TRNFLDKEYF command to encrypt production data, do the following steps:

1. Make sure you have *ALL authority to the database file containing the field to decrypt.
2. Within a test environment, you should have tested TRNFLDKEYF.
3. No applications or users should be currently using the database file containing the field to translate.
4. The TRNFLDKEYF command will perform a mass re-encryption of the current field values. You should allocate enough downtime for the TRNFLDKEYF to execute. Execution times will vary depending on the processor speed of your system, the number of records in your database file, and other activity running on the system at the time. In order to estimate the execution time for TRNFLDKEYF, you should run the TRNFLDKEYF command over some test data first.

The TRNFLDKEYF command performs the following primary steps:

1. Obtains an exclusive (*EXCL) lock on the database file containing the field to encrypt.
2. Optional: Creates a backup of the database file (containing the field to encrypt) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
3. Changes the status of the field to *PROCESS.
4. Reads all records in the file and re-encrypts the field values to the new key entered.
5. Changes the *CURRENT keys in the field registry to the new keys entered on the command.
6. The exclusive lock will be released on the database file containing the encrypted field.
7. The status of the field entry will be changed to *ACTIVE.

How to Get There

In the Field Keys Menu, choose option **5**.

Options

Field identifier (FLDID)

Indicate the unique name of the field entry to translate the field key for.

Encryption key label (ENCKEYLBL)

Indicate the label of the Symmetric Key to use for encrypting the field values.

Encryption key store name (ENCKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for encryption of the field. The users (or user groups) which need to encrypt values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Decryption key label (DECKEYLBL)

Indicate the label of the Symmetric Key to use for decrypting the field values.

The possible values are:

decryption-key-label Indicate the label of the key to use for decryption.

> WARNING: If specifying a different key label than the label specified for encryption, then that decryption key should contain the same key value as the encryption key.

***ENCKEYLBL** Use the same label as specified on the ENCKEYLBL parameter.

Decryption key store name (DECKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the field. The users (or user groups) that need access to the decrypted values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***ENCKEYSTR** Use the same Key Store as specified on the ENCKEYSTR parameter.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

***LIBL** Locate the Key Store within the library list.

Save database file (SAVDTA)

Indicate if the database file (containing the field to encrypt) should be saved (backed up) into a Save File before the translation process begins. It is highly recommended to save the database file for error recovery purposes.

The possible values are:

***YES** Save the database file into a Save File before translation begins.

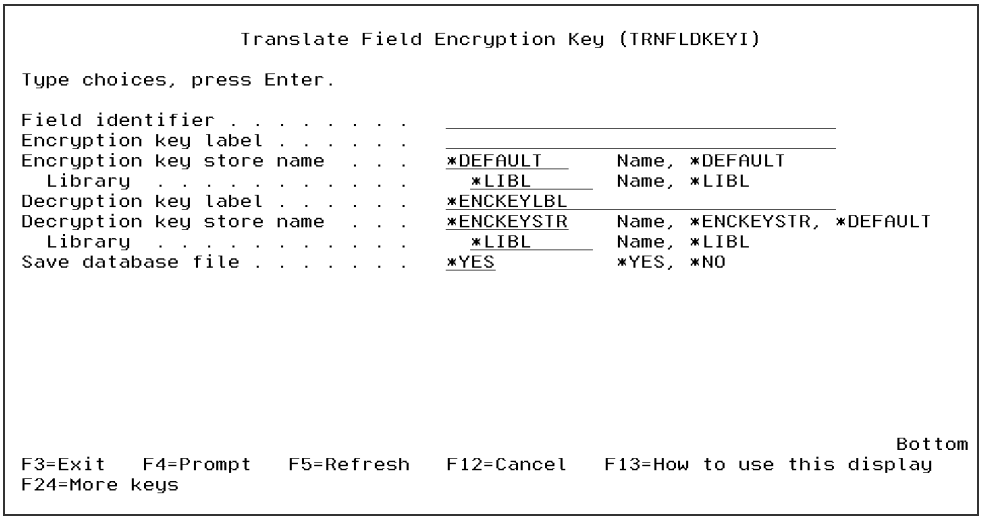
NOTE:

- The created Save File will be named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
- Before using this option, ensure that enough disk space is available for a saved copy of the database file.

***NO** Do not save the database file before the translation process begins.

Translate Field Encryption Key – Internal Storage (TRNFLDKEYI)

The Translate Field Encryption Key Internal (TRNFLDKEYI) command allows authorized users to translate (re-encrypt) field values to a new key.



NOTE:

- TRNFLDKEYI can be used for *ACTIVE field entries which store the encrypted values in the existing database field (without using a DB2 Field Procedure).
- It is recommended to submit this command to batch using the SBMJOB command.
- TRNFLDKEYI will attempt to get an exclusive lock on the file, so no users or applications should be using the file at the time.
- The execution time for TRNFLDKEYI depends on the number of records which it must translate (re-encrypt) to the new Key.

For each record found, TRNFLDKEYI will decrypt the value with the old key and re-encrypt the value using the new key specified.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

IMPORTANT: Before using the TRNFLDKEYI command to encrypt production data, do the following steps:

1. Make sure you have *ALL authority to the database file containing the field to decrypt.
2. Within a test environment, you should have tested TRNFLDKEYI.
3. No applications or users should be currently using the database file containing the field to translate.
4. The TRNFLDKEYI command will perform a mass re-encryption of the current field values. You should allocate enough downtime for the TRNFLDKEYI to execute. Execution times will vary depending on the processor speed of your system, the number of records in your database file, and other activity running on the system at the time. In order to estimate the execution time for TRNFLDKEYI, you should run the TRNFLDKEYI command over some test data first.

The TRNFLDKEYI command performs the following primary steps:

1. Obtains an exclusive (*EXCL) lock on the database file containing the field to encrypt.
2. Optional: Creates a backup of the database file (containing the field to encrypt) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
3. If triggers are used on the field, they are removed.
4. Changes the status of the field to *PROCESS.
5. Reads all records in the file and re-encrypts the field values to the new key entered.
6. Changes the *CURRENT keys in the field registry to the new keys entered on the command.
7. If triggers are used on the field, they are re-added.
8. The exclusive lock will be released on the database file containing the encrypted field.
9. The status of the field entry will be changed to *ACTIVE.

How to Get There

In the Field Keys Menu, choose option **4**.

Options

Field identifier (FLDID)

Indicate the unique name of the field entry to translate the field key for.

Encryption key label (ENCKEYLBL)

Indicate the label of the Symmetric Key to use for encrypting the field values.

Encryption key store name (ENCKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for encryption of the field. The users (or user groups) which need to encrypt values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Decryption key label (DECKEYLBL)

Indicate the label of the Symmetric Key to use for decrypting the field values.

The possible values are:

decryption-key-label Indicate the label of the key to use for decryption.

WARNING: If specifying a different key label than the label specified for encryption, then that decryption key should contain the same key value as the encryption key.

***ENCKEYLBL** Use the same label as specified on the ENCKEYLBL parameter.

Decryption key store name (DECKEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Key to use for decryption of the field. The users (or user groups) that need access to the decrypted values will need to have at least *USE authority to this Key Store object.

The possible values are:

key-store-name Enter the name of the Key Store.
***ENCKEYSTR** Use the same Key Store as specified on the ENCKEYSTR parameter.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.
***LIBL** Locate the Key Store within the library list.

Save database file (SAVDTA)

Indicate if the database file (containing the field to encrypt) should be saved (backed up) into a Save File before the translation process begins. It is highly recommended to save the database file for error recovery purposes.

The possible values are:

***YES** Save the database file into a Save File before translation begins.

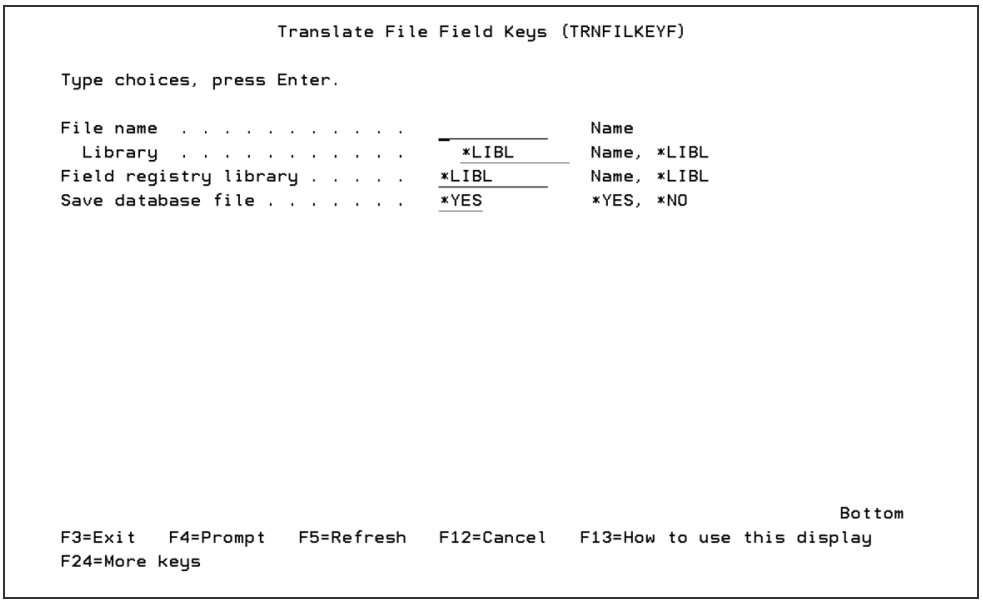
NOTE:

- The created Save File will be named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
- Before using this option, ensure that enough disk space is available for a saved copy of the database file.

***NO** Do not save the database file before the translation process begins.

Translate File Encryption Key - Field Procedure (TRNFILKEYF)

The Translate File Encryption Key - Field Procedure (TRNFILKEYF) command allows authorized users to translate (re-encrypt) field values in a file (when a Pending Key has been entered) to the new pending Keys previously entered.



NOTE:

- TRNFILKEYF can be used for *ACTIVE field entries which use a DB2 Field Procedure for automatic encryption/decryption.
- It is recommended to submit this command to batch using the SBMJOB command.
- TRNFILKEYF will attempt to get an exclusive lock on the file, so no users or applications should be using the file at the time.
- The execution time for TRNFILKEYF depends on the number of records which it must translate (re-encrypt) to the new Key.

For each record found, TRNFLDKEYF will decrypt the value with the old key and re-encrypt the value using the new key specified.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires that you have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

This command requires that you have *USE authority for the Key Stores that contain both the current and new encryption/decryption Keys.

IMPORTANT: Before using the TRNFILKEYF command to encrypt production data, do the following steps:

1. Make sure you have *ALL authority to the database file containing the field to decrypt.
2. Within a test environment, you should have tested TRNFILKEYF.
3. No applications or users should be currently using the database file containing the field to translate.
4. The TRNFILKEYF command will perform a mass re-encryption of the current field values. You should allocate enough downtime for the TRNFLDKEYF to execute. Execution times will vary depending on the processor speed of your system, the number of records in your database file, and other activity running on the system at the time. In order to estimate the execution time for TRNFILKEYF, you should run the TRNFILKEYF command over some test data first.

The TRNFILKEYF command performs the following primary steps:

1. Obtains an exclusive (*EXCL) lock on the database file containing the field to translate.
2. Optional: Creates a backup of the database file (containing the field to encrypt) into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
3. Changes the status of the field to *PROCESS.
4. Reads all records in the file and re-encrypts the field values to the new key entered.
5. Changes the *CURRENT keys in the field registry to the Pending keys previously entered.
6. The exclusive lock will be released on the database file containing the encrypted fields.
7. The status of the field entry will be changed to *ACTIVE.

How to Get There

In the Field Keys Menu, choose option **6**. Or, in the File Field Encryption Menu, choose option **5**.

Options

File name (TRNFILE)

Specify the name of the file that contains the field(s) to translate.

The possible values are:

file-name The name of the file that contains the field(s) to translate.

The possible library values are:

library-name Enter the name of the library where the file is located.
***LIBL** Locate the file within the library list.

Save database file (SAVDTA)

Indicate if the database file (containing the field to encrypt) should be saved (backed up) into a Save File before the translation process begins. It is highly recommended to save the database file for error recovery purposes.

The possible values are:

***YES** Save the database file into a Save File before translation begins.

NOTE:

- The created Save File will be named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
- Before using this option, ensure that enough disk space is available for a saved copy of the database file.

***NO** Do not save the database file before the translation process begins.

Translate Key Store (TRNKEYSTR)

The Translate Key Store (TRNKEYSTR) command allows authorized users to translate (re-encrypt) the Symmetric Keys within a Key Store to the *CURRENT version of a Master Encryption Key (MEK).

Notes

- TRNKEYSTR can be executed while users and applications are active on the system.
- TRNKEYSTR will not modify any existing data contained within your database files.
- Existing data will not need to be re-encrypted after a TRNKEYSTR is performed.
- After executing the TRNKEYSTR command, you should verify that the Key verification values (KEYVV) match between the Key Store and the Master Key by viewing those values with the DSPKEYSTR and DSPMSTKEY commands.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain key stores" authority setting.

The user must have *CHANGE authority to the Key Store Validation List (*VLDL) object and *USE authority to the library that contains the Key Store.

How to Get There

From the Symmetric Encryption Key Menu, choose option **3**.

Options

Key store name (KEYSTR)

Indicate the Key Store name and Library to translate.

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

To MEK id number (TOMEKID)

Indicate the id number of the Master Encryption Key (MEK) which will be used to translate (re-encrypt) any entries contained in the Key Store. The *CURRENT version of the specified MEK must exist.

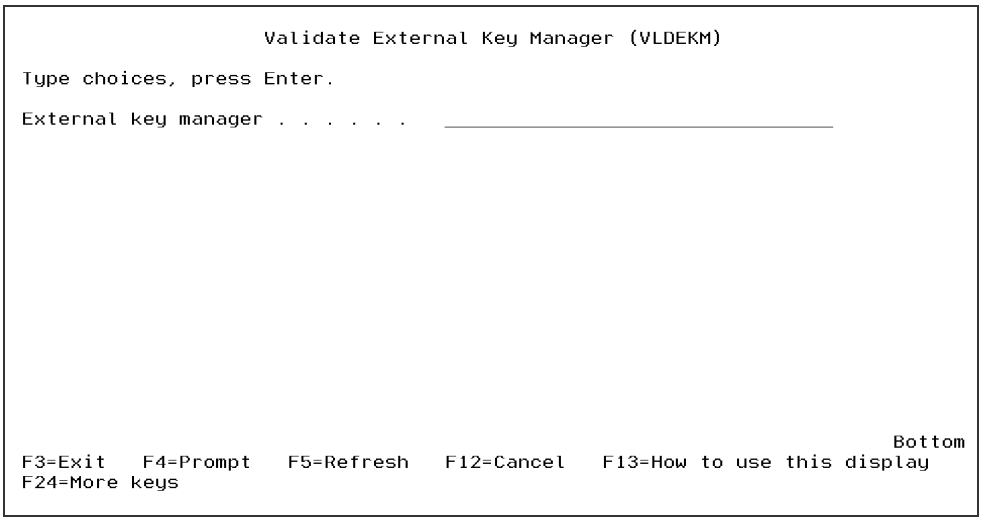
The possible values are:

mek-id-number Indicate a number from 1-8.

Validate External Key Manager (VLDEKM)

The VLDEKM command allows an authorized user to test the connection to an External Key Manager.

The user must have *USE authority to the CRVL001 Validation List (*VLDL) object. Also ensure the user has at least *USE authority to the library that contains the Key Store.



How to Get There

In the External Key Manager Menu, choose option **6**. Or, submit the command **VLDEKM**.

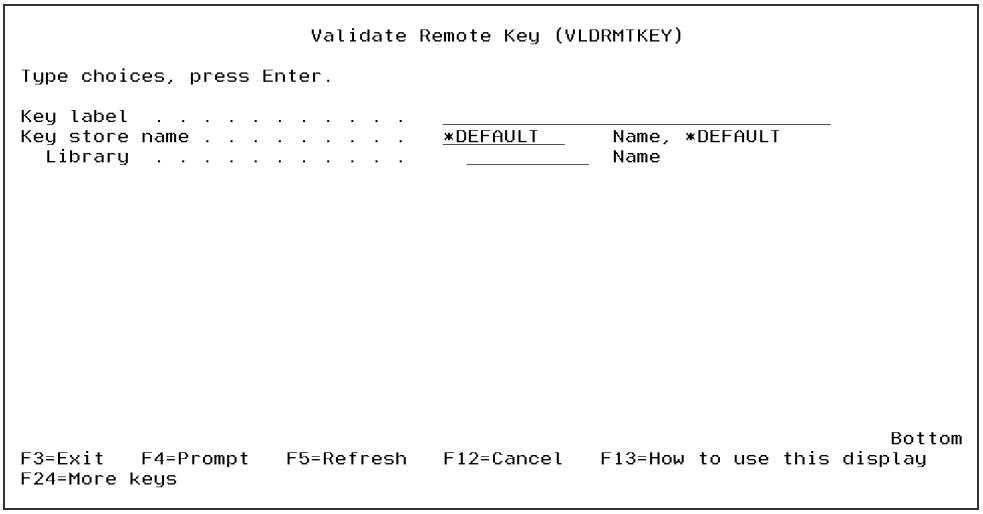
Field Descriptions

Manager id (EKMGRID)

Indicate the unique name of the external key manager entry.

Validate Remote Key (VLDRMTKEY)

The VLDRMTKEY command allows authorized users to verify that a key can be accessed in an External Key Manager.



Do the following steps to validate a Remote Key:

1. Prompt (F4) the command of **CRYPTO/VLDRMTKEY**.
2. Type in the Key label and Key Store Name.
3. Press Enter to validate the key.

Field Descriptions

Key label

Indicate the name (label) of the Key to validate.

Key store name and Library

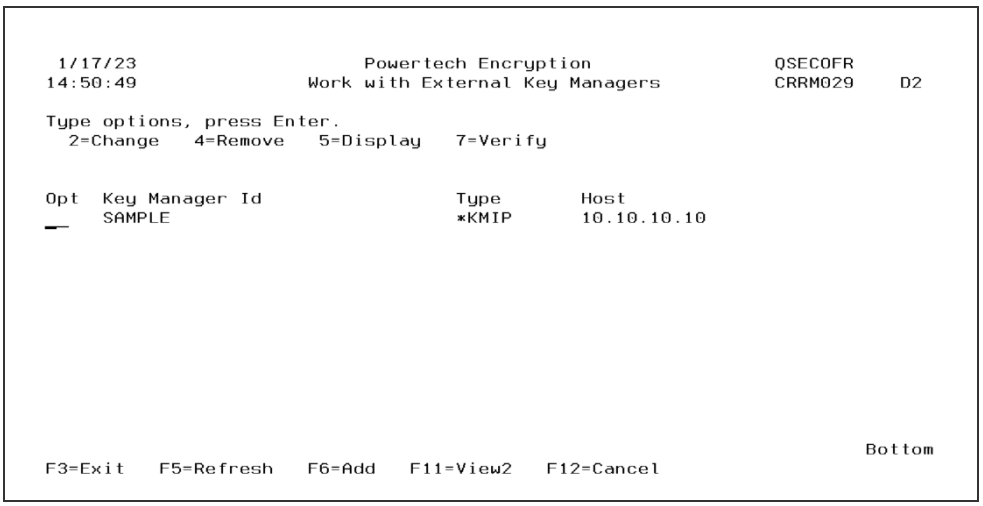
Indicate the name and library of the Key Store containing the Key to validate. Specify *DEFAULT to use the default key store name specified in the Key Policy. The user must have *USE authority to the Key Store *VLDL object and *USE authority to the library that contains the Key Store.

Work with External Key Managers (WRKEKM)

The WRKEKM command allows an organization to work with the external key managers in the environment.

The following users can use this command:

- QSECOFR user profile
- A user profile with *SECADM authority
- A Key Officer that has a *YES specified for the “Maintain key policy and alerts” authority setting



How to Get There

From the External Key Manager Menu, choose option **1**. Or, submit the command **WRKEKM**.

Options

For each External Key Manager listed on the screen, you can use one of the following options.

2=Change

Displays the Change External Key Manager (CHGEKM) panel, which allows you to change the settings for the External Key Manager.

4=Remove

Displays the Remove External Key Manager (RMVEKM) panel, which prompts you to confirm the removal of the External Key Manager.

5=Display

Displays the Display External Key Manager (DSPEKM) panel, which displays the current settings for the External Key Manager.

7=Verify

Displays the Validate External Key Manager (VLDEKM) panel, which verifies the connection to the External Key Manager.

Function Keys

F3 (Exit): Exits the WRKEKM screen.
F5 (Refresh): Refreshes the list of External Key Managers.
F6 (Add): Displays the Add External Key Manager Entry (ADDEKM) panel, which allows you to add a new External Key Manager using the command.

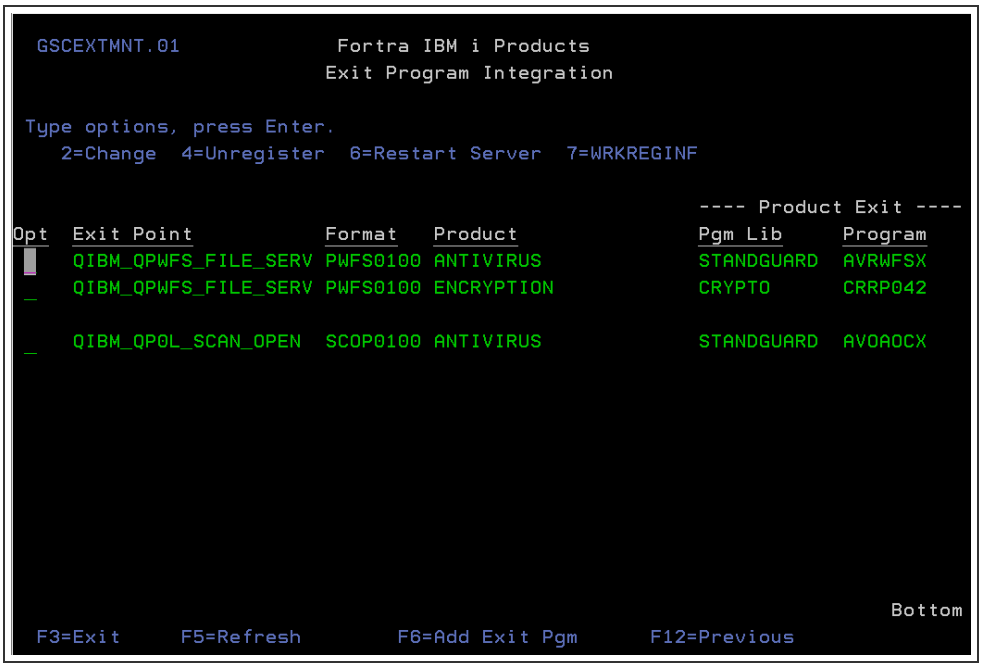
Work with Exit Program Integration (WRKEXTPGM) Command

The Work with Exit Program Integration (WRKEXTPGM) command is used to configure exit program integration. See Overview of Exit Program Integration.

IMPORTANT:
The command should be used only in the following two situations:

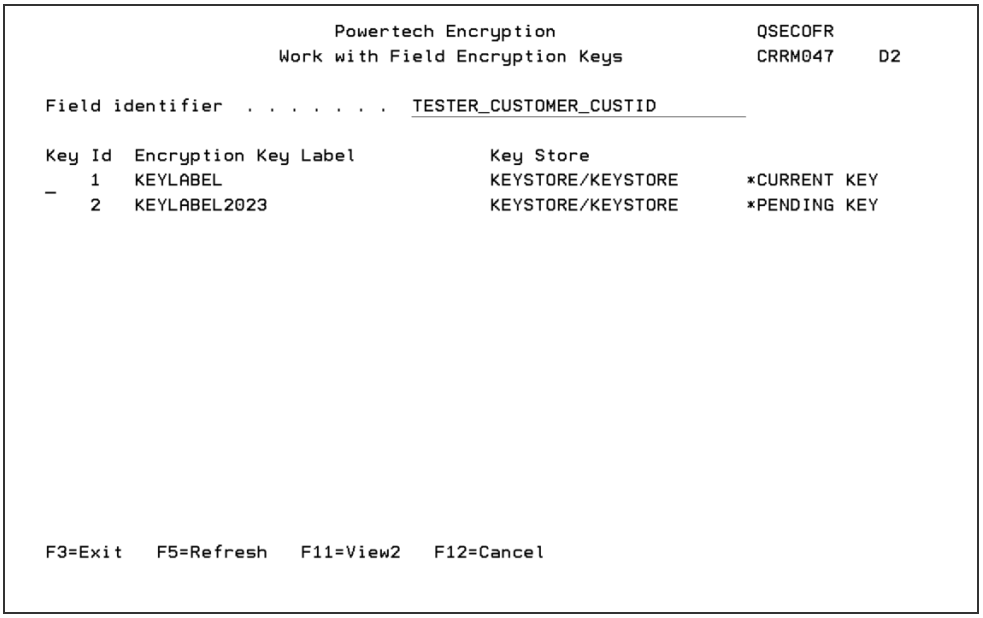
1. To add an exit program to one of the exit points supported by the integration, if the product does not support automatic exit point integration.
2. At the direction of Powertech Technical Support.

This command has no parameters and displays the Exit Program Integration screen.



Work with Field Encryption Keys (WRKFLDKEY)

The Work with Field Enc. Keys (WRKFLDKEY) command allows authorized users to display the keys used for encrypting/decrypting a field entry in the Registry.



This command will show the history of the keys used, as well as the current keys being used to encrypt and decrypt data.

This command requires the user to have *USE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

The key shown with the value of "*CURRENT KEY" is the current Key id used to encrypt and decrypt field values.

This screen will show *PENDING Keys, if entered.

How to Get There

From the Field Keys Menu, choose option **1**, Work with Field Encryption Keys. Or, prompt (**F4**) the command of **CRYPTO/WRKFLDKEY**.

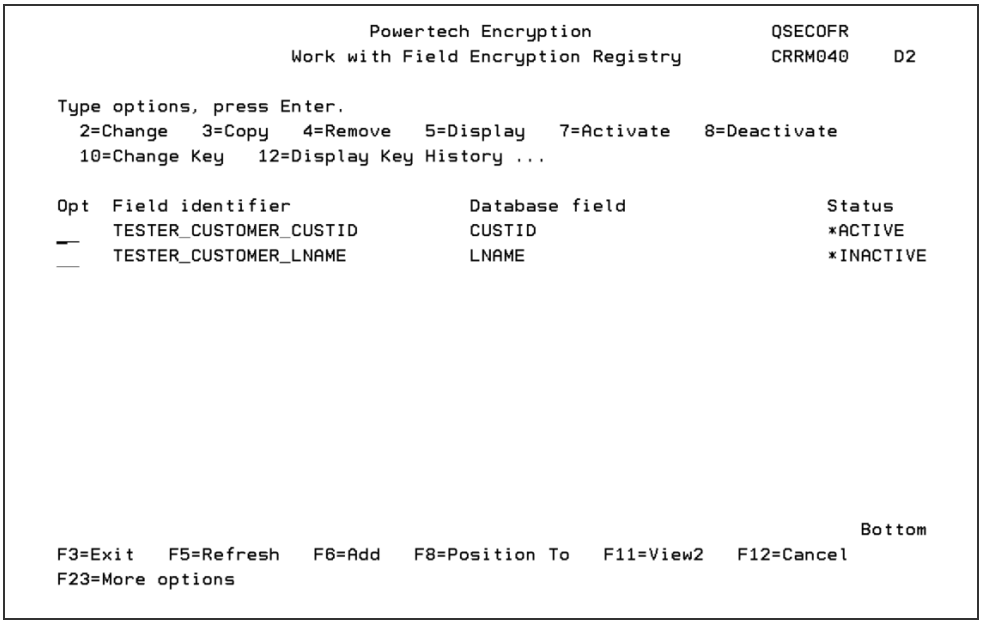
Options

Field identifier (FLDID)

Indicate the unique name of the entry to display the keys used.

Work with Field Encryption Registry (WRKFLDENC)

The WRKFLDENC command allows authorized users to work with the entries in the Field Encryption Registry. This command's screen includes functions to add, change, activate, deactivate and remove field entries.



How to Get There

From the Field Encryption Menu, choose option **1**, Work with Field Encryption (WRKFLDENC). Or, execute the command **CRYPTO/WRKFLDENC**.

For each field entry listed, the WRKFLDENC screen shows the user-assigned field identifier, the actual database field name and the status. Press **F11** to view more information about each entry.

Status Codes

*ACTIVE

The field entry is activated for encryption.

*INACTIVE

The field entry is not activated for encryption.

*PROCESS

The field entry is currently being processed for activation or deactivation.

*ERROR

The activation or deactivation process failed and requires Fortra support.

Options

For each entry listed on the screen, you can utilize one of the following options.

2=Change

Displays the Change Field Encryption Entry (CHGFLDENC) panel, where you can change the field entry using the CHGFLDENC command.

3=Copy

Displays the Copy Field Encryption Entry (CPYFLDENC) panel, where you can copy the field entry using the CPYFLDENC command.

4=Remove

Displays a prompt to confirm the removal of the field entry using the RMVFLDENC command.

5=Display

Displays the Display Field Encryption Entry (DSPFLDENC) panel, which shows the values for the field entry using the DSPFLDENC command.

7=Activate

Displays a prompt to activate the field entry for encryption using the ACTFLDENC command.

8=Deactivate

Displays a prompt to deactivate the field entry from encryption using the DCTFLDENC command.

10=Change Key

Displays the Change Field Encryption Key (CHGFLDKEY) panel, which allows you to change the field entry's encryption/decryption keys using the CHGFLDKEY command.

12=Display Key History

Displays the history of the Keys (used to encrypt/decrypt the field entry values) using the WRKFLDKEY command.

14=Edit Full Auth List

Displays the Edit Authorization List command for Authorization List for the Full Value.

15=Edit Mask Auth List

Displays the Edit Authorization List command for Authorization List for the Masked Value.

16=Change Field Mask

Displays the Change Field Mask (CHGFLDMSK) panel, where you can change the field mask using the CHGFLDMSK command.

17=Change Field Auth. Lists

Displays the Change Field Authorization Lists (CHGFLDAUTL) panel, where you can change the field Authority lists using the CHGFLDAUTL command.

18=Add Pending Key

Displays the Add a Field Encryption Pending Key (ADDPNDKEY) panel, where you can add a pending key using the ADDPNDKEY command.

19=Change Pending Key

Displays the Change a Field Encryption Pending Key (CHGPNDKEY) panel, where you can change a pending key using the CHGPNDKEY command.

20=Remove Pending Key

Displays a prompt to allow you to remove a pending key using the RMVPNDKEY command.

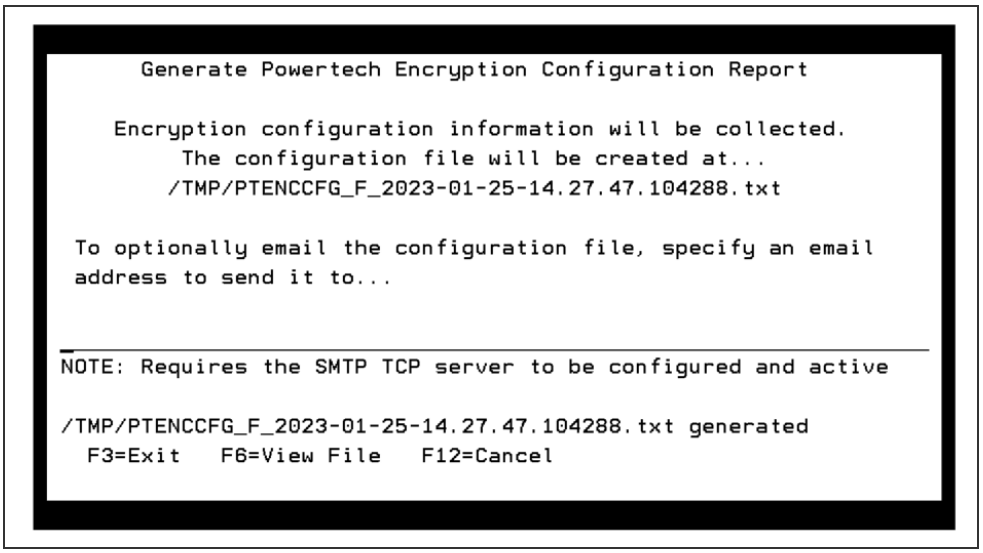
21=Display Pending Key

Displays a prompt to allow you to display a pending key using the DSPPNDKEY command.

22=Generate Config Report

Displays a message detailing the location of where the encryption configuration file will be created. You can also enter a valid email address to where the configuration report can be sent. Note that this option requires the SMTP TCP server to be both configured and active.

Press Enter to generate the report.



Function Keys

F3 (Exit): Exits the WRKFLDENC screen.

F5 (Refresh): Refreshes the list of field entries in the Registry.

F6 (Add): Opens the Add Field Encryption Entry (ADDFLDENC) panel, where you can add a new field entry in the Registry (using the ADDFLDENC command).

F8 (Position To): Displays filters to select the list entries on the screen.

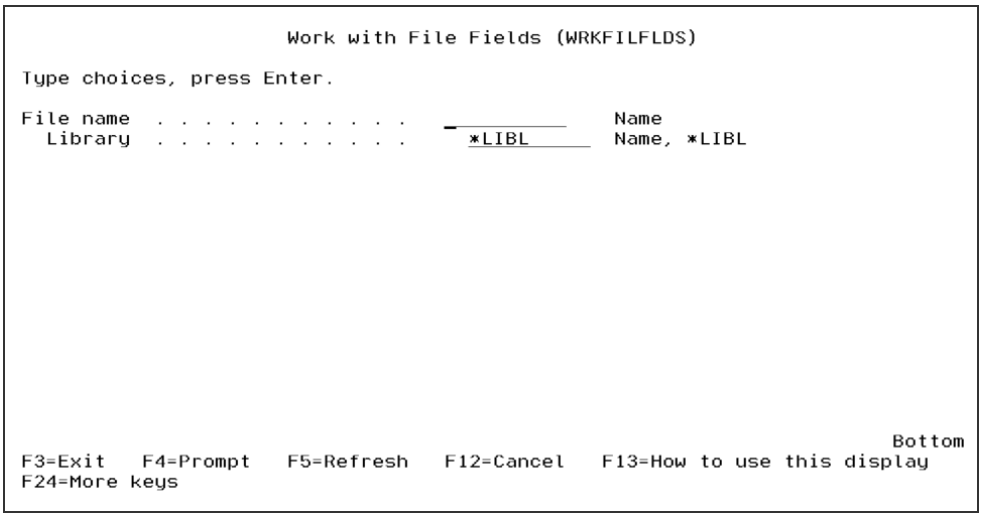
F11 (View2): Additionally shows the field entry's database file name, external storage setting and trigger setting.

F12 (Cancel): Cancel the current screen and go to the previous screen.

F23 (More options): Show more options.

Work with File Fields (WRKFILFLDS)

The Work with File Fields (WRKFILFLDS) command allows authorized users to work with fields in a file and the Field Encryption Registry.



The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires the user to have *CHANGE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

> **NOTE:** The WRKFILFLDE command only adds the field entry settings into the registry. It will not cause any action to be performed on the actual database field in the file. The field will not be activated for encryption until the ACTFLDENC (Activate Field Encryption) command is executed.

How to Get There

On the File Field Encryption Menu, choose option **2**, Work with File Fields.

Options

The possible values are:

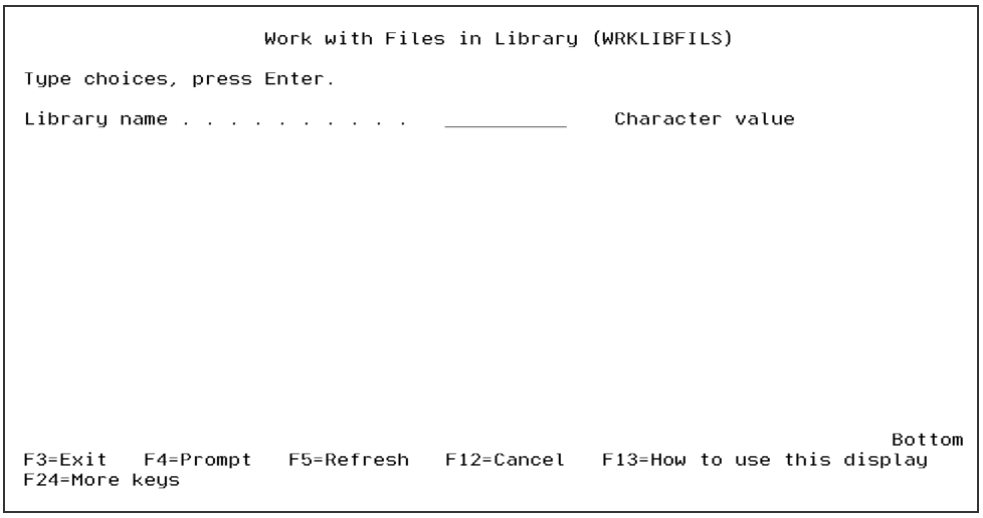
file-name The name of the file that contains the field(s) to work with.

The possible library values are:

library-name Enter the name of the library where the file is located.
***LIBL** Locate the file within the library list.

Work with Files in Library (WRKLIBFILS)

The Work with Files in Library (WRKLIBFILS) command allows authorized users to work with files in a library and show which files may have encrypted fields.



The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer whom has a *YES specified for the "Maintain Field Enc. Registry" authority setting

This command requires the user to have *USE authority to the CRVL002 Validation List (*VLDL) object which contains the Field Encryption Registry.

> **NOTE:** The WRKLIBFILS command will show files in a library and how many fields have entries in the Field Registry and how many are encrypted.

How to Get There

On the File Field Encryption Menu, choose option **1**, Work with Library Files.

Options

Library name (LIBRARY)

Specify the name of the library to search for files.

The possible values are:

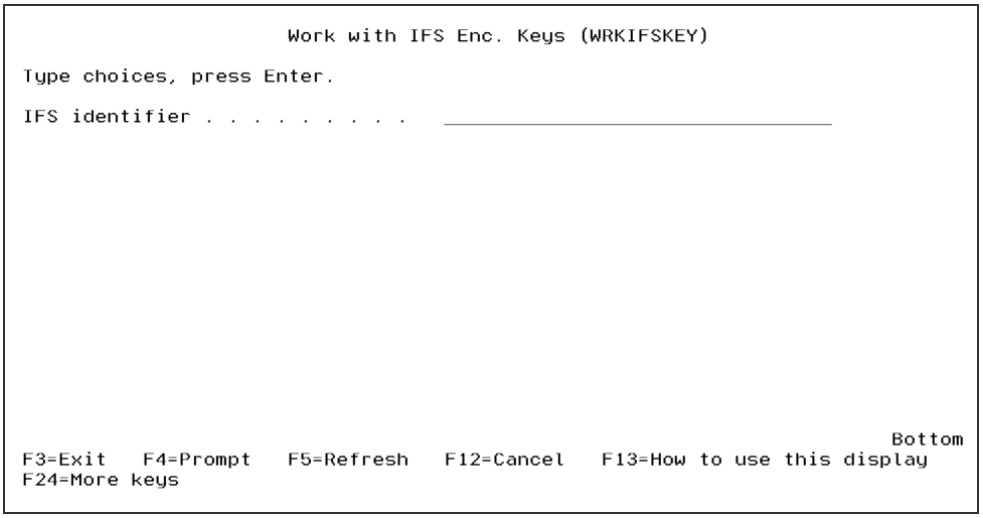
library-name The name of the library to search for files.

Work with IFS Encryption Keys (WRKIFSKEY)

The Work with IFS Encryption Keys (WRKIFSKEY) command allows authorized users to display the keys used for an entry in the IFS encryption registry.

This command will show the history of the keys used, as well as the current keys being used to encrypt and decrypt the files associated with the registry entry.

This command requires the user to have *USE authority to the CRVL003 Validation List (*VLDL) object which contains the IFS Encryption Registry.



How to Get There

From the IFS Keys Menu, choose option **1**. Or, submit the command **WRKIFSKEY**.

Options

IFS identifier (IFSID)

Indicate the unique name of the entry to display the keys used.

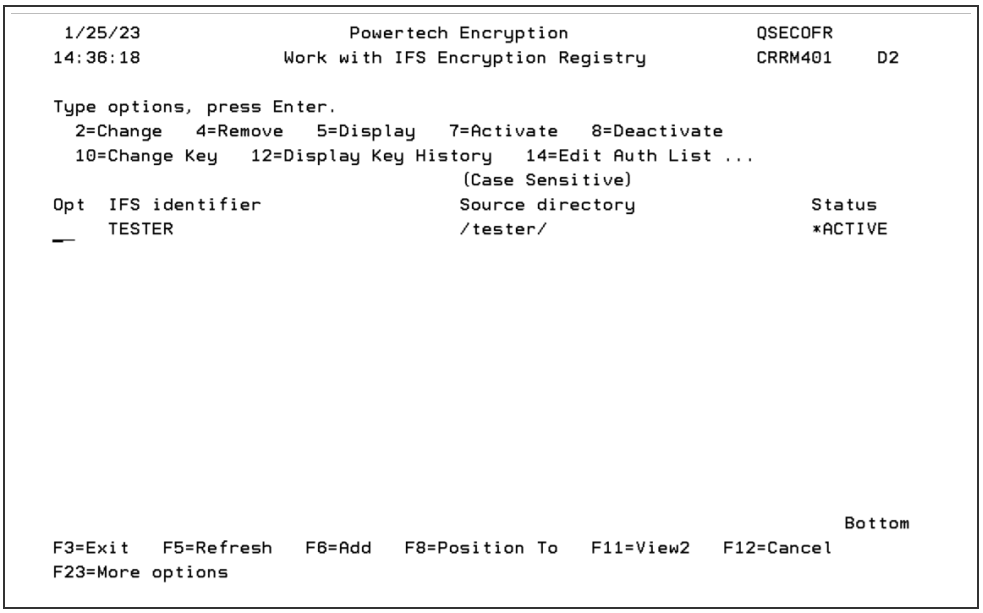
Function Keys

F3 (Exit): Exits the WRKIFSKEY screen.
F5 (Refresh): Refreshes the list of keys for the IFS entry.
F11 (View2): Additionally shows the decryption key label and Key Store, as well as the user/timestamp recorded when the key was changed.

Work with IFS Encryption Registry (WRKIFSENC)

The WRKIFSENC command allows authorized users to work with the entries in the IFS Encryption Registry. This command's screen includes functions to add, change, activate, deactivate and remove IFS entries.

See also Getting Started with IFS Encryption.



How to Get There

From the IFS Encryption Menu, choose option **1**. Or, execute the command **CRYPTO/WRKIFSENC.** The current entries within the IFS Encryption Registry will be displayed.

Status Codes

*ACTIVE

The IFS entry is activated for encryption.

*INACTIVE

The IFS entry is not activated for encryption.

*PROCESS

The IFS entry is currently being processed for activation or deactivation.

*ERROR

The activation or deactivation process failed and requires Fortra support.

Options

For each IFS entry listed, the WRKIFSENC screen will show the user-assigned IFS identifier, the source directory and the status. Press **F11** to view more information about each entry.

2=Change

Displays a prompt to change the IFS entry using the CHGIFSENC command. See Change IFS Encryption Entry (CHGIFSENC) panel.

4=Remove

Displays a prompt to confirm the removal of the IFS entry using the RMVIFSENC command.

5=Display

Displays the values for the IFS entry using the DSPIFSENC command.

7=Activate

Displays a prompt to activate the IFS entry for encryption using the ACTIFSENC command.

8=Deactivate

Displays a prompt to deactivate the IFS entry from encryption using the DCTIFSENC command.

10=Change Key

Displays a prompt to change the IFS entry's encryption/decryption keys using the CHGIFSKEY command. See Change IFS Encryption Key (CHGIFSKEY) panel.

12=Display Key History

Displays the history of the Keys (used to encrypt/decrypt the IFS entry values) using the WRKIFSKEY command.

14=Edit Auth List

Goes to the Edit Auth List (EDTAUTL) command for the Authorization List in the entry. If an entry does not contain an Authorization List, then this will go to the Work with Authorization Lists screen.

Function Keys

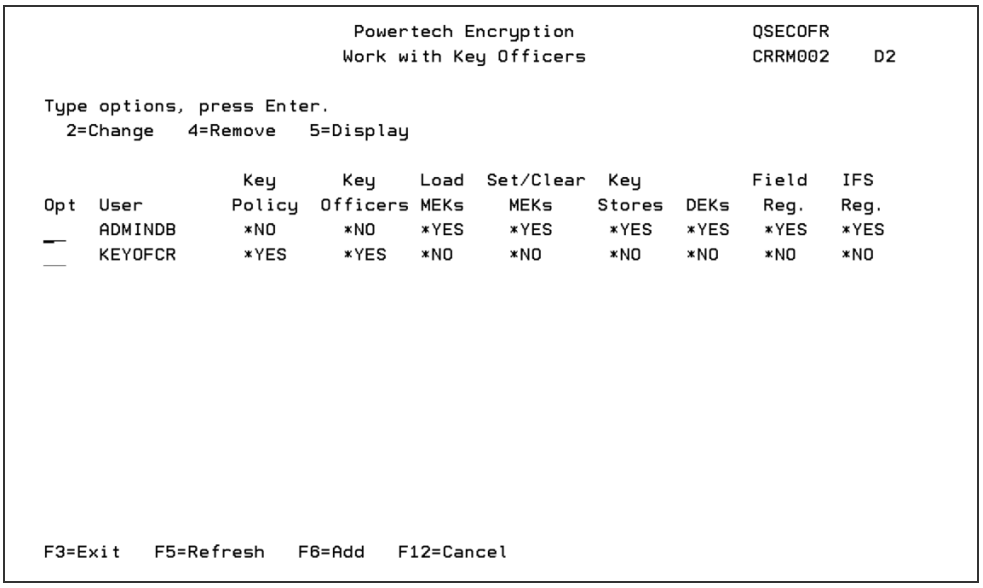
F3 (Exit): Exits the WRKIFSENC screen.
F5 (Refresh): Refreshes the list of IFS entries in the Registry.
F6 (Add): Displays a prompt to add a new IFS entry in the Registry using the ADDIFSENC command.
F8 (Position To): Displays filters to select the list entries on the screen.
F11 (View2): Additionally shows the IFS entry's directory to store the encrypted values, the include subdirectories value and the Decryption Authorization List.
F12 (Cancel): Cancel the current screen and go to the previous screen.
F23 (More options): Show more options.

Work with Key Officers (WRKKEYOFR)

Key Officers are those users that are authorized to create and manage Master Encryption Keys (MEKs), Key Stores, Data Encryption Keys (DEKs) and the Field Encryption Registry.

The WRKKEYOFR command allows an organization to work with the key officers within the Symmetric Key environment.

The Key Officers and their authority settings are stored in the CRYPTO library by default. These settings are encrypted with the Product Encryption Key (PEK).



NOTE: A user does not need to be a Key Officer to encrypt and decrypt data.

The following users can use this command:

- QSECOFR user profile
- A user profile with *SECADM authority
- A Key Officer that has a *YES specified for the “Maintain key officers” authority setting

How to Get There

From the Key Policy and Security Menu, choose option **10**, Work with Key Officers. Or, execute the command **CRYPTO/WRKKEYOFR**.

Options

For each Key Officer listed on the screen, you can use one of the following options.

2=Change

Displays a prompt to change the authority settings for the Key Officer using the CHGKEYOFR command.

4=Remove

Displays a prompt to confirm the removal of the Key Officer using the RMVKEYOFR command.

5=Display

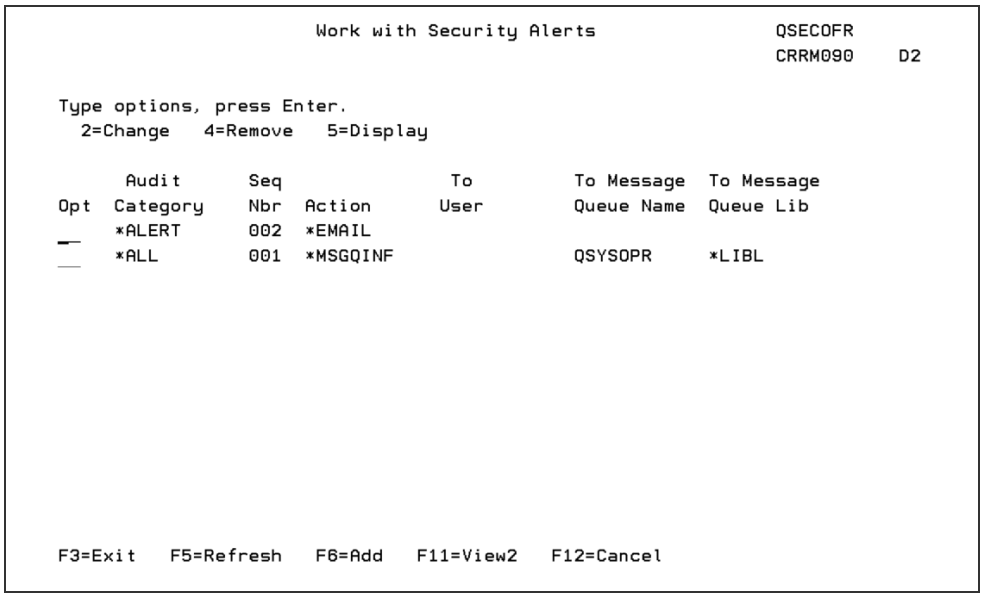
Displays the current authority settings for the Key Officer using the DSPKEYOFR command.

Function Keys

F3 (Exit): Exits the screen.
F5 (Refresh): Refreshes the list of Key Officers.
F6 (Add): Displays a prompt to add a new Key Officer using the ADDKEYOFR command.
F12 (Cancel): Cancel the current screen and go to the previous screen.

Work with Security Alerts (WRKCCALR)

The WRKCCALR command allows authorized users to configure and view the Security Alerts settings.



The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain key policy and alerts” authority setting

How to Get There

From the Key Policy and Security Menu, choose option **3**, Work with Security Alerts. Or, execute the command **CRYPTO/WRKCCALR**.

Options

For each Alert listed on the screen, you can use one of the following options.

2=Change

Displays a prompt to change the settings for the Alert using the Change Alert command (CHGCCALR).

4=Remove

Displays a prompt to confirm the deletion of the Alert using the Delete Alert command (DLTCCALR).

5=Display

Displays the current settings for the Alert using the Display Alert command (DSPCCALR).

Function Keys

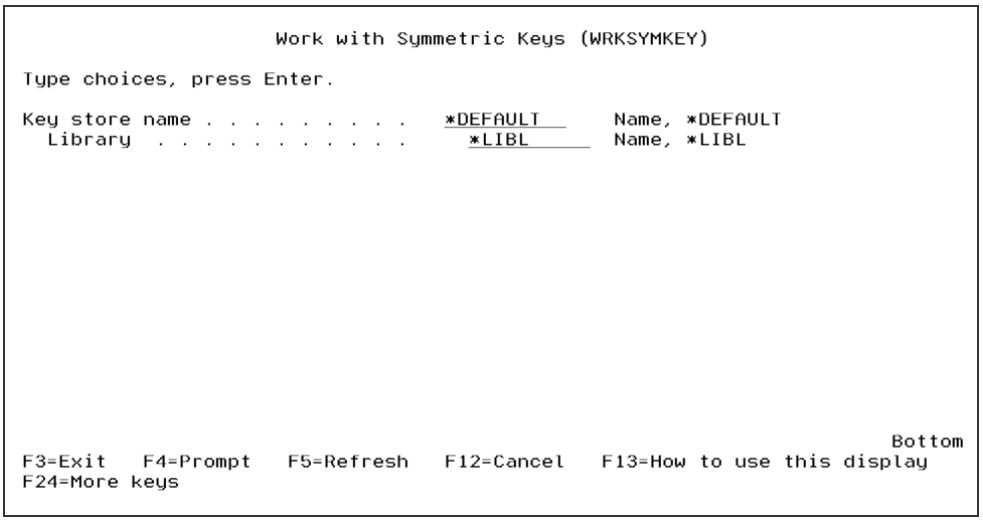
F3 (Exit): Exits the WRKCCALR screen.
F5 (Refresh): Refreshes the list of Alerts.
F6 (Add): Displays a prompt to add a new Alert using the ADDCCALR command .

See also Display Alert (DSPCCALR).

Work with Symmetric Keys (WRKSYMKEY)

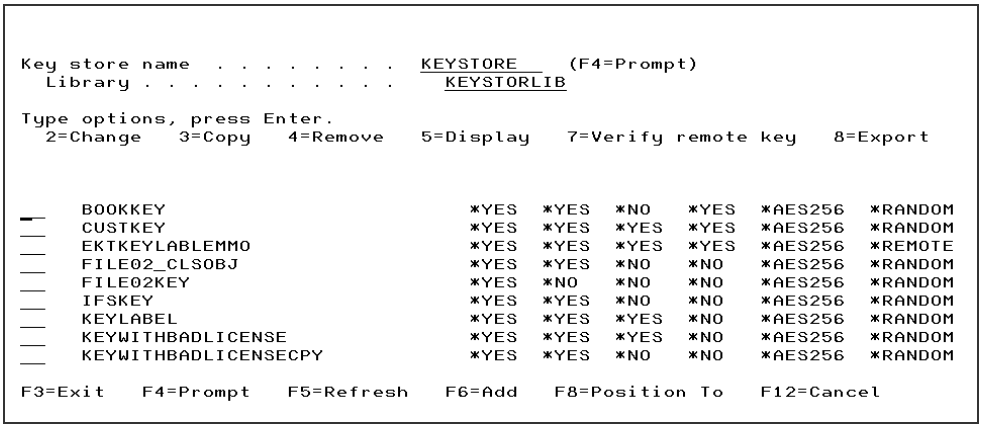
The Work with Symmetric Keys (WRKSYMKEY) command allows authorized users to work with the Symmetric Keys contained in a Key Store.

This command will display a list of existing Symmetric Keys in the Key Store. Functions will be available to create new keys, change keys, display keys and delete keys.



How to Get There

From the Symmetric Encryption Key Menu, choose option **10**.



Field Descriptions

Key store name (KEYSTR)

Indicate the object name and library of the Key Store which contains the Symmetric Keys to work with.

key-store-name Enter the name of the Key Store.
***DEFAULT** Use the default Key Store name specified at the Key Policy level.

The possible library values are:

library-name Enter the name of the library where the Key Store is located.

***LIBL** Locate the Key Store within the library list.

Options

2=Change

Displays a prompt to change the Key attributes using the CHGSYMKEY command.

3=Copy

Displays a prompt to copy the Key attributes using the CPYSYMKEY command.

4=Remove

Displays a prompt to confirm the removal of the Key using the DLTSYMKEY command.

5=Display

Displays the Key attributes. The actual Key value will not be displayed.

7=Verify Remote Key

Verifies a remote key that is located in an External Key Manager.

8=Export

Exports the actual Key value using the EXPSYMKEY command (if the Key Policy allows).

Function Keys

F3 (Exit): Exits the WRKSYMKEY screen.
F4 (Prompt): Provides assistance in entering or selecting a command.
F5 (Refresh): Refreshes the list of Keys in the Key Store.
F6 (Create): Displays a prompt to create a new Key using the CRTSYMKEY command.
F8 (Position To): Type the starting characters of a key to which to position the start of the list in the display.
F12 (Cancel): Cancel the current screen and go to the previous screen.

Work with IFS Encryption (WRKIFSENC)

The Work with IFS Encryption (WRKIFSENC) command allows authorized users to work with the entries in the IFS Encryption Registry. This command's screen includes functions to add, change, activate, deactivate and remove IFS entries.

Add IFS Encryption Entry (ADDIFSENC)

The ADDIFSENC command allows authorized users to add a new entry into the IFS Encryption Registry.

> **NOTE:** The ADDIFSENC command only adds the IFS entry settings into the registry. It will not cause any action to be performed on the actual files in the directory(s). The IFS will not be activated for encryption until the ACTIFSENC (Activate IFS Encryption) command is executed.

The following users can use the ADDIFSENC command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

This command requires that you have *CHANGE authority to the CRVL003 Validation List (*VLDL) object, which contains the IFS Encryption Registry.

Do the following steps to add a new entry in the IFS Encryption Registry:

1. Prompt (F4) the command of **CRYPTO/ADDIFSENC**.
2. Press F1 on any parameter for complete online help text.
3. Press Enter after the parameter values are entered.

```
                    Add IFS Encryption Entry (ADDIFSENC)

Type choices, press Enter.



IFS identifier . . . . . . . . BANK_DATA   _______________

IFS directory (to encrypt) . . /BankData_____ ____________________

Include sub directories . . . *YES         *YES, *NO

Encrypted files storage folder /EncryptedData/BankData_________

Encryption key label . . . . . . CREDITCARDKEY____________________

Encryption key store name  . . .  *DEFAULT__   Name, *DEFAULT

  Library  . . . . . . . . . . .    *LIBL_____  Name, *LIBL

Decryption key label . . . . . .  *ENCKEYLBL_____________________

Decryption key store name  . . .  *ENCKEYSTR    Name, *ENCKEYSTR, *DEFAULT

  Library  . . . . . . . . . . .    *LIBL_____  Name, *LIBL

Decryption authorization list  .  CCDECRYPT    Name, *NONE

Journal location . . . . . . . .  *DEFAULT     *DEFAULT, *ASP, *LOC1...
```

Screen Example: ADDIFSENC Command with Sample Values

ADDIFSENC IFS Descriptions:

IFS identifier	Indicate the unique identifier (name) of the entry up to 30 characters. This name cannot contain spaces or certain special characters. Underscore characters can be used in the name (i.e. ACCOUNT_NUMBER). The name is not case sensitive - it will be stored in upper case.
IFS directory (to encrypt)	Specify the path of the IFS directory containing the files to be encrypted.
Include subdirectories	Encrypt files that exist in the subdirectories of the Source directory.

Encrypted files storage folder	Specify the path of the IFS directory to store the encrypted versions of the files.
Encryption key label	Indicate the label of the initial key to use for encrypting the IFS files.
Encryption key store name Library	Indicate the name and library of the Key Store which contains the Encryption key label. Specify *DEFAULT to use the default Key Store name specified in the Key Policy.
Decryption key label	Indicate the label of the initial key to use for decrypting the IFS files. Specify *ENCKEYLBL to use the same label name that is entered for the Encryption key label. Caution: If specifying a different key label than the label specified for encryption, then that decryption key should contain the same key value as the encryption key.
Decryption key store name Library	Indicate the name and library of the Key Store that contains the Decryption key label. Specify *DEFAULT to use the default key store name specified in the Key Policy. Specify *ENCKEYSTR to use the same value which is entered for the Encryption key store name.
Decryption authorization list	Indicate the IBM i Authorization List that should be used by the IFS decryption APIs for checking the user's permissions to decrypt for the IFS files. Specify *NONE to not use an Authorization List. * see additional note below
Journal location	Indicate the location of the journal and related objects. Specify *DEFAULT to use the CRYPTO library to store the objects. Specify *IASP to use an IASP library to store the objects. Specify *LOC1 through *LOC5 when an external journal is used to journal the IFS directory and files.

NOTE: An Authorization List can be created with the CRTAUTL command. The users (or user groups) which need access to the decrypted files will need at least *USE authority to the Authorization List. Additionally the users which need access to the decrypted files are required to have at least *USE authority to the Key Store object which holds the Decryption Key.

Change IFS Encryption Entry (CHGIFSENC)

The CHGIFSENC command allows authorized users to change the settings for an *INACTIVE IFS entry in the Encryption Registry.

> **NOTE:** The CHGIFSENC command will only change the settings for an IFS entry in the Encryption Registry. It will not cause any action to be performed on the actual files in the directory(s).

The following users can use the CHGIFSENC command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

This command requires that you have *CHANGE authority to the CRVL003 Validation List (*VLDL) object, which contains the IFS Encryption Registry.

Do the following steps to change an IFS entry in the Encryption Registry:

1. Prompt (F4) the command of **CRYPTO/CHGIFSENC**.
2. Enter the IFS identifier to change, and then press Enter.
3. The current IFS entry settings (parameter values) will be displayed.
4. Press F1 on any parameter for complete online help text.
5. Press Enter after the parameter values are changed.
6. For an explanation of the parameters, refer to the documentation for the ADDIFSENC command.

```
      Change IFS Encryption Entry (CHGIFSENC)


IFS identifier . . . . . . . . BANK DATA
IFS directory (to encrypt) . . /BankData

_____

Include sub directories . . . *YES         *YES, *NO
Encrypted files storage folder /EncryptedData/BankData
Decryption authorization list  CCDECRYPT    Name, *NONE
Journal location . . . . . . .  *DEFAULT     *DEFAULT, *ASP, *LOC1...
```

Screen Example: CHGIFSENC Command with Sample Values

Display IFS Encryption Entry (DSPIFSENC)

The DSPIFSENC command allows authorized users to display an IFS entry's settings within the IFS Encryption Registry.

This command requires that you have *USE authority to the CRVL003 Validation List (*VLDL) object, which contains the IFS Encryption Registry.

Do the following steps to display an IFS entry's settings within the Encryption Registry:

1. Prompt (F4) the command of **CRYPTO/DSPIFSENC**.
2. Enter the IFS identifier to display, and then press Enter.
3. The current IFS entry settings (parameter values) will be displayed, along with the user and timestamp recorded when the IFS entry was added or last changed.
4. Press F1 on any parameter for complete online help text.
5. For an explanation of the parameters, refer to the documentation for the ADDIFSENC command.

Display IFS Encryption Entry (DSPIFSENC)

Type choices, press Enter.

IFS identifier . . . . . BANK_DATA

IFS directory (to encrypt) . . . /BankData

Include sub directories . . . . *YES

Encrypted files storage folder . /EncryptedData/BankData

Encryption key label . . . . . CREDITCARDKEY

Encryption key store name . . . *DEFAULT

Library . . . . . *LIBL

Decryption key label . . . . . *ENCKEYLBL

Decryption key store name . . . *ENCKEYSTR

Library . . . . . *LIBL

Decryption authorization list . CCDECRYPT

Journal location . . . . . *DEFAULT

Screen Example: DSPIFSENC Command with Sample Values

Activate IFS Encryption (ACTIFSENC)

The ACTIFSENC command allows authorized users to activate an IFS entry in the Encryption Registry.
This command will perform a mass-encryption of the current IFS files in the directory(s) to encrypt. You should only run this command when no applications are currently using the IFS files.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

IMPORTANT: The following specific authorities are required for users running this command:

- *RWX data authority and *ALL object authority to the directories and files to encrypt.
- *CHANGE authority to the CRVL003 (Validation List object which contains the IFS Encryption Registry), CRPFIFS, CRPFIFSL1, CRPFIFSL2, CRPFIFSL3, CRPFIFSL4, CRPFIFS2 and CRPFIFSLOG files, which will be updated during this process.
- *USE authority to the Authorization List assigned to this entry. “

WARNING: Before using the ACTIFSENC command to encrypt production data, make sure you have performed the following steps:

1. Verified you have all the previously listed authorities.
2. Within a test environment, you should have tested ACTIFSENC, and tested your applications thoroughly with encrypted files.
3. No applications or users should be currently using the directory(s) and files to encrypt.

The ACTIFSENC command will perform a mass encryption of the current IFS files in the directory(s). You should allocate enough application downtime for the ACTIFSENC to execute. Execution times will vary depending on the processor speed of your system, the number of files, and other activity running on the system at the time. In order to estimate the execution time for ACTIFSENC, you should run the ACTIFSENC command over some test data first.

Check (and double check) the IFS entry settings using the DSPIFSENC command. Especially make sure the Source directory name, Target directory name and include subdirectories settings are correct.

Recommendations for ACTIFSENC command:

- Run ACTIFSENC in batch using the SBMJOB command.
- Specify *YES for the “Save directory” parameter to save a copy of the directory(s) to Encrypt and the files into a Save File before the activation process. This option is important for error recovery purposes.
- Ensure that enough disk space is available for a saved copy of the directory and files.

Do the following steps to activate a IFS entry in the Encryption Registry:

1. Prompt (F4) the command of **CRYPTO/ACTIFSENC**.
2. Type in the IFS identifier to activate and press Enter.

```
      Activate IFS Encryption (ACTIFSENC)

Type choices, press Enter.


IFS identifier . . . . . BANK_DATA      _____________

Save directory(s). . . . . *YES         *YES, *NO
```

Screen Example: ACTIFSENC Command with Sample Values

The ACTIFSENC command performs the following steps:

1. Optional: Creates a backup of the Source directory structure and files into a Save file named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999. This file is placed in the CRYPTO Library.
2. Performs a mass encryption of the current IFS files in the source directory and optionally subdirectories.
3. Journaling will be started over the directory and if Include subdirectories is set to *YES then journaling will be started over the subdirectories as well.
4. The status of the IFS entry will be changed to *ACTIVE.

Notes on ACTIFSENC:

- After the ACTIFSENC command completes: Once you have determined that your applications are working properly with the encrypted files, you can remove the Save file (created in step 1 above) containing a backup of the Source directory structure and Files.
- To activate an iASP entry, you should run this command while in the iASP.

Change IFS Encryption Key (CHGIFSKEY)

The CHGIFSKEY command allows authorized users to change (rotate) the keys used to encrypt and decrypt data for an IFS entry in the Encryption Registry. Up to 99,999 keys can be rotated for an IFS entry.

This command can be used for *INACTIVE IFS entries, as well as *ACTIVE IFS entries. All Encrypted files will use the original Key that was used to encrypt them when they are Decrypted.

The following users can use the CHGIFSKEY command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

This command requires that you have *CHANGE authority to the CRVL003 Validation List (*VLDL) object, which contains the IFS Encryption Registry.

This command requires that you have *CHANGE authority to the CRPFIFS2 file which will be updated during this process.

Do the following steps to change the keys for a IFS entry in the Encryption Registry:

1. Prompt (F4) the command of **CRYPTO/CHGIFSKEY.**
2. Press F1 on any parameter for complete online help text.
3. Press Enter after the parameter values are specified.

```
        Change IFS Encryption Key (CHGIFSKEY)

Type choices, press Enter.



IFS identifier . . . . . BANK_DATA       _______________

Encryption key label . . . . . BANKDATA_KEY_2012_10__________

Encryption key store name  . . .  *DEFAULT__   Name, *DEFAULT

 Library  . . . . . .  *LIBL_____  Name, *LIBL

Decryption key label . . . . . *ENCKEYLBL_____________________

Decryption key store name  . . .  *ENCKEYSTR    Name, *ENCKEYSTR, *DEFAULT

 Library  . . . . . .  *LIBL_____  Name, *LIBL
```

Screen Example: CHGIFSKEY Command with Sample Values

IFS descriptions:

IFS identifier	Indicate the unique identifier of the IFS entry in the encryption registry.
Encryption key label	Indicate the label of the key to use for encrypting the IFS values.
Encryption key store name Library	Indicate the name and library of the Key Store that contains the Encryption key label. Specify *DEFAULT to use the default Key Store name specified in the Key Policy.
Decryption key label	Indicate the label of the key to use for decrypting the IFS values. Specify *ENCKEYLBL to use the same value which is entered for the Encryption key label. Caution: If specifying a different key label than the label specified for encryption, then that decryption key should contain the same key value as the encryption key.
Decryption key store name Library	Indicate the name and library of the Key Store that contains the Decryption key label. Specify *DEFAULT to use the default Key Store name specified in the Key Policy. Specify *ENCKEYSTR to use the same value which is entered for the Encryption key store name.

The Decryption Key Label, Key Store Name and Library are stored in the Encrypted File and are used to Decrypt the file.

When Key Labels are changed with the CHGIFSKEY command, the new Key information is saved in any newly encrypted IFS file. However the Key Information remains the same for any existing encrypted IFS files, which will allow Powertech Encryption for IBM i to decrypt those IFS values using the prior Key Labels. This technique allows you to rotate the keys frequently without having to immediately re-encrypt existing IFS files.

Work with IFS Encryption Keys (WRKIFSKEY)

The WRKIFSKEY command allows authorized users to view the current key, as well as the history of keys used to encrypt and decrypt data for an IFS entry in the Encryption Registry.

Do the following steps to view the keys for an IFS entry in the Encryption Registry:

1. Prompt (F4) the command of CRYPTO/WRKIFSKEY.
2. Specify the IFS identifier and press enter.
3. The keys for the IFS entry will be displayed.

```
7/18/21        Work with IFS Encryption Keys



IFS identifier . . . . . DATA_____________________



Key Id  Encryption Key Label       Key Store

  1  BANKDATA_KEY_2006_10      KEYLIB/KEYSTORE
  2  BANKDATA_KEY_2007_10      KEYLIB/KEYSTORE
  3  BANKDATA_KEY_2008_10      KEYLIB/KEYSTORE
  4  BANKDATA_KEY_2009_10      KEYLIB/KEYSTORE
  5  BANKDATA_KEY_2010_10      KEYLIB/KEYSTORE
  6  BANKDATA_KEY_2011_10      KEYLIB/KEYSTORE
  7  BANKDATA_KEY_2012_10      KEYLIB/KEYSTORE  *CURRENT KEY



F3=Exit   F5=Refresh   F11=View2   F12=Cancel
```

Screen Example: WRKIFSKEY Command with Sample Values

The key shown with the value of "*CURRENT KEY" is the current Key id used to encrypt IFS files. The file will be decrypted using the Key Label and Key Store that is saved in the Encrypted file.

Deactivate IFS Encryption (DCTIFSENC)

The DCTIFSENC command allows authorized users to deactivate an IFS entry in the Encryption Registry.

The following users can use this command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer who has a *YES specified for the "Maintain IFS Enc. Registry" authority setting

IMPORTANT: The following specific authorities are required for users running this command:

- *RWX data authority and *ALL object authority to the directories and files in the Source and Target that are to be decrypted.
- *CHANGE authority to the CRVL003 (Validation List object which contains the IFS Encryption Registry), CRPFIFS, CRPFIFSL1, CRPFIFSL2, CRPFIFSL3, CRPFIFSL4, CRPFIFS2 and CRPFIFSLOG files, which will be updated during this process.
- *USE authority to the Authorization List, Keys and Keystore that are assigned to this entry. “

WARNING: Before using the DCTIFSENC command to decrypt production data, do the following steps:

1. Make sure you have *ALL authority to the Source and Target directories containing the IFS files to decrypt.
2. Make sure you have at least *USE authority to the Key Store(s) which hold the Data Encryption Keys (DEKs) that will be used to decrypt the data. You can use the WRKIFSKEY command to find out which Key Store(s) and DEKs are used to decrypt the IFS values. If you created any of these DEKs yourself, in which you are considered the owner of these DEK(s), then the “DEK decrypt usage by owner” setting (viewable in the DSPKEYPCY command) must be a *YES.
3. Make sure you have at least *USE authority to the Authorization List used to allow for decryption of the data.
4. Within a test environment, you should have tested DCTIFSENC and tested your applications thoroughly with decrypted values.
5. No applications or users should be currently using the directory containing the IFS files to decrypt.
6. The DCTIFSENC command will perform a mass decryption of the current IFS files. You should allocate enough downtime for the DCTIFSENC to execute. Execution times will vary depending on the processor speed of your system, the number of files, and other activity running on the system at the time. In order to estimate the execution time for DCTIFSENC, you should run the DCTIFSENC command over some test data first.

Recommendations for DCTIFSENC command:

- Run DCTIFSENC in batch using the SBMJOB command.
- Specify *YES for the “Save directories” parameter to save a copy of the source directory structure and files and the target directory structure and files into a Save File before the deactivation process. This option is important for error recovery purposes.
- Ensure that enough disk space is available for a saved copy of the directory structures.

Do the following steps to deactivate an IFS entry in the Encryption Registry:

1. Prompt (F4) the **CRYPTO/DCTIFSENC** command.
2. Enter the IFS identifier to deactivate, and then press Enter.

```
        Deactivate IFS Encryption (DCTIFSENC)

Type choices, press Enter.


IFS identifier . . . . . DATA     _____________

Save directory(s). . . . . *YES        *YES, *NO
```

Screen Example: DCTIFSENC Command with Sample Values

The DCTIFSENC command performs the following steps:

1. Optional: Creates a backup of the IFS directory and subdirectories if INCSUBDIR is *YES (containing the source files) into a Save File named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
2. Optional: Creates a backup of the IFS directory and subdirectories if INCSUBDIR is *YES (containing the encrypted files) into a Save File named BACKUPxxxxx, where xxxxx is a sequential number from 1 to 99999.
3. Journaling will be stopped for the directory(s).
4. Performs a mass Decryption of IFS files in the directory(s).
5. The status of the IFS entry will be changed to *INACTIVE.

Notes on DCTIFSENC:

- After the DCTIFSENC command completes: Once you have determined that your applications are working properly with the decrypted values, you can remove the Save files (created in steps 1 and 2 above) containing the backup of the directory(s)

containing the source files and the directory(s) containing the encrypted files.

- To deactivate an iASP entry you should be run this command while in the iASP.

Remove IFS Encryption Entry (RMVIFSENC)

The RMVIFSENC command allows authorized users to remove an *INACTIVE IFS entry from the Encryption Registry.

The RMVIFSENC command will only remove the IFS entry from the Encryption Registry. It will not cause any action to be performed on the IFS files in the directory(s).

The following users can use the RMVIFSENC command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

This command requires that you have *CHANGE authority to the CRVL003 Validation List (*VLDL) object, which contains the IFS Encryption Registry.

Do the following steps to remove an IFS entry from the Encryption Registry:

1. Prompt (F4) the command of CRYPTO/RMVIFSENC.
2. Enter the IFS identifier to remove, and then press Enter.

Start IFS Server Job (STRIFSENCJ)

The STRIFSENCJ command submits the server job to batch. This job will monitor the journal records created when files are accessed and run processes after a file has been accessed.

The following users can use the STRIFSENCJ command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

This command uses the CRYPTO Job Description that is shipped in the CRYPTO Library when submitting the Server Job (IFSENCJOB) to Batch. You can change this Job Description to run the IFSENCJOB where and how you want. This command should be added to the system's start-up program.

```
      Start IFS Encryption Job (STRIFSENCJ)

 Type choices, press Enter.



Server user profile . . . . . *CURRENT _____________  Name, *CURRENT

Journal location . . . . . *DEFAULT     *DEFAULT, *ASP, *LOC1...
```

Screen Example: STRIFSENCJ Command

The Journal Location parameter will determine which journal to monitor. The job names used will be IFSENCJOB(*DEFAULT), IFSENCJOBA(*IASP), IFSENCJOB1(*LOC1), IFSENCJOB2(*LOC2), IFSENCJOB3(*LOC3), IFSENCJOB4(*LOC4) or IFSENCJOB5 (*LOC5)

The Server User Profile must have the following authorities:

- ALL authority to the Directory(s) to encrypt and all the files in the directory(s).
- ALL authority to the Directory(s) that hold the encrypted files and the files in the directory(s).
- CHANGE authority to the CRPFIFS, CRPFIFSL1, CRPFIFSL2, CRPFIFSL3, CRPFIFSL4, CRPFIFS2, CRPFIFSLOG Files.
- CHANGE authority to the CRDEBUG, CRLSTSEQ and CRSRVRUN Data Areas.
- USE authority to the CRJNI001 Journal and all Journal Receivers.
- USE authority to the CRCL414 and CRRP040 programs in the CRYPTO Library.

End IFS Server Job (ENDIFSENCJ)

The ENDIFSENCJ command sends a message to stop the Server Job (IFSENCJOB) and should be added to the system's end routine or at least be run before taking the system to a restricted state.

The following users can use the ENDIFSENCJ command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)

- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

```
        End IFS Encryption Job (ENDIFSENCJ)

Type choices, press Enter.



Journal location . . . . . .   *DEFAULT      *DEFAULT, *ASP, *LOC1...
```

Screen Example: ENDIFSENCJ Command

The Journal Location parameter will determine which server job to end. The job ended will be IFSENCJOB(*DEFAULT), IFSENCJOBA(*IASP), IFSENCJOB1(*LOC1), IFSENCJOB2 (*LOC2), IFSENCJOB3(*LOC3), IFSENCJOB4(*LOC4) or IFSENCJOB5(*LOC5).

Add Exit Programs (ADDIFSEXTP)

The ADDIFSEXTP command will add the QIBM_QPWFS_FILE_SERV, QIBM_QP0L_ SCAN_CLOSE and QIBM_QP0L_SCAN_OPEN exit programs to the system.

> **NOTE:** The ADDIFSEXTP command will add the exit programs to the Registry however any job that was active at the time will need to be restarted.

The following users can use the ADDIFSEXTP command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

The Add Exit Programs (ADDIFSEXTP) command will just register the exit programs in the system. Any Job that needs to use those exit programs will need to be restarted. There are a couple of ways of doing this.

1. IPL the System.
2. Or end any restart any Job that will be using the exit programs. The instructions below will stop and restart many of the servers. There may be others that are not listed.
    a. End Processes
        i. ENDTCPSVR *NETSVR
        ii. ENDTCPSVR SERVER(*FTP)
        iii. ENDTCPSVR SERVER(*HTTP) HTTPSVR(*ALL)
        iv. ENDHOSTSVR *FILE
        v. ENDHOSTSVR *DATABASE
        vi. ENDSBS QSERVER
        vii. End any Batch Jobs that access the IFS Data
    b. Restart processes
        a. STRSBS QSERVER
        b. STRTCPSVR *NETSVR
        c. STRTCPSVR SERVER(*FTP)
        d. STRTCPSVR SERVER(*HTTP) HTTPSVR(*ALL)
        e. STRHOSTSVR *FILE
        f. STRHOSTSVR *DATABASE
        g. Restart any Batch Jobs that access the IFS Data

The user profile must have the following authorities:

1. *USE authority to the CRRP041 exit program in the CRYPTO Library.
2. *EXECUTE authority to the CRYPTO library.
3. *ALL authority to the Directory(s) to encrypt and all the files in the directory(s).
4. *ALL authority to the Directory(s) that hold the encrypted files and the files in the directory(s).
5. *CHANGE authority to the CRPFIFS, CRPFIFSL1, CRPFIFSL2, CRPFIFSL3, CRPFIFSL4, CRPFIFS2, CRPFIFSLOG Files.
6. *CHANGE authority to the CRDEBUG, CRLSTSEQ and CRSRVRUN Data Areas.
7. *Use Authority to the CRJNI001 Journal and all Journal Receivers.

WARNING: If the user profile is not valid or accessible at the time the exit program is called, the action on the IFS file will be ignored, which may cause the file to NOT be encrypted or decrypted at the appropriate time.

Remove IFS Exit Point Programs (RMVIFSEXTP)

The RMVIFSEXTP command will remove the QIBM_QPWFS_FILE_SERV, QIBM_QP0L_SCAN_CLOSE and QIBM_QP0L_SCAN_OPEN exit programs from the system.

> **NOTE:** The RMIFSVEXTP command will remove the exit programs from the Registry however any job that was active at the time will need to be restarted.

The following users can use the RMVIFSEXTP command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

The Remove IFS Exit Programs (RMVIFSEXTP) command will remove the IFS Exit Programs from the Registry. Any Jobs that are using the exit programs will need to be restarted. There are a couple of ways of doing this.

1. IPL the System.
2. Or end any restart any Job that will be using the exit programs. The instructions below will stop and restart many of the servers. There may be others that are not listed.
    a. End Processes
        i. ENDTCPSVR *NETSVR
        ii. ENDTCPSVR SERVER(*FTP)
        iii. ENDTCPSVR SERVER(*HTTP) HTTPSVR(*ALL)
        iv. ENDHOSTSVR *FILE
        v. ENDHOSTSVR *DATABASE
        vi. ENDSBS QSERVER
        vii. End any Batch Jobs that access the IFS Data
    b. Restart processes
        a. STRSBS QSERVER
        b. STRTCPSVR *NETSVR
        c. STRTCPSVR SERVER(*FTP)
        d. STRTCPSVR SERVER(*HTTP) HTTPSVR(*ALL)
        e. STRHOSTSVR *FILE

f. STRHOSTSVR *DATABASE
g. Restart any Batch Jobs that access the IFS Data

Display IFS Debug Mode (DSPIFSDBG)

The DSPIFSDBG command allows users to view the Debug Mode. .

This command requires that you have *USE authority to the CRDEBUG Data Area.

Change IFS Debug Mode (CHGIFSDBG)

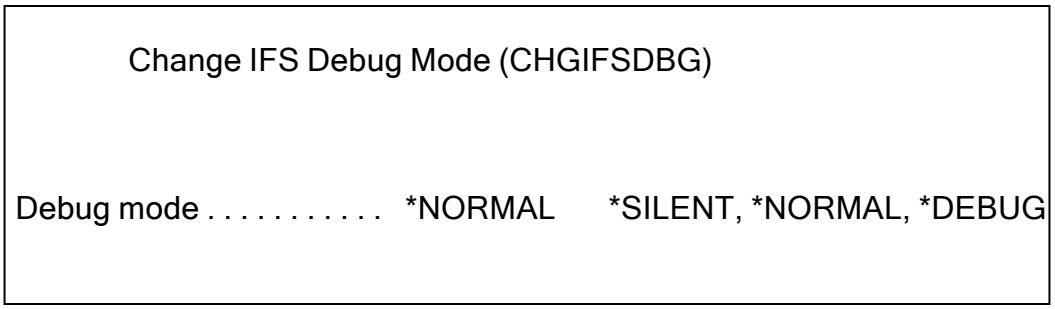
The CHGIFSDBG command allows authorized users to change the Debug Mode.
.

The following users can use the CHGIFSDBG command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

This command requires that you have *CHANGE authority to the CRDEBUG Data Area.

This command requires that you have *CHANGE authority to the CRPFIFS2 File.



Screen Example: CHGIFSDBG Command

Clear IFS Debug Log (CLRIFSLOG)

The CLRIFSLOG command will clear all debug records from the CRPFIFSLOG file.

The following users can use the CLRIFSLOG command:

- QSECOFR user profile (unless excluded in the Key Officer settings)
- A user profile with *SECADM authority (unless excluded in the Key Officer settings)
- A Key Officer that has a *YES specified for the “Maintain IFS Enc. Registry” authority setting

This command requires that you have *USE authority to the CRVL003 Validation List (*VLDL) object which contains the IFS Encryption Registry.

This command requires that you have *CHANGE authority to the CRPFIFSLOG File.

IFS Registry Authorization List

Within your applications, you may want to control if IFS files are available for each user to access, based on their authorities. You can control this application-level security through IBM i Authorization Lists and Powertech Encryption for IBM i’s IFS Encryption Registry.

Listed below is an example of the steps needed to create Authorization Lists and then associate them to a IFS in the IFS Encryption Registry:

1. Create an IBM i Authorization List to control authority to DECRYPT values for IFS files. Example:

   > CRTAUTL AUTL(CCFULL) TEXT(‘Auth. List of Users allowed to decrypt’)

2. For the CCDATA Authorization List, grant *USE authority only to those users (or user groups) that should have access to decrypt the values. Example:

   > EDTAUTL AUTL(CCDATA)

Appendix

The topics in this section include additional information about Powertech Encryption for IBM i.

Appendix A: All-Object Authority

By default, all-object (*ALLOBJ) special authority allows the user to access any resource on the system whether (or not) private authority exists for the user. Even if the user has *EXCLUDE authority to an object, *ALLOBJ special authority still allows the user to access the object. A user with *ALLOBJ can view, change, or delete any object. These users can also grant object authority to other users on the system.

NOTE:
Listed below are the steps which IBM performs when checking a user's authority on an object:

1. Does the user profile have *ALLOBJ authority specified in his user profile? If so, authority is granted to use the object.
2. Does the user profile have individual authority to the object? The user will be granted or denied access based on whatever object authorities he/she possesses in the object.
3. Does the user profile have authority through an external authorization list that is associated with the object? If present, the user is granted or denied access based on those authorities in the authorization list.
4. Is the user a member of a group profile that has *ALLOBJ authority? If so, the user is granted access to the object.
5. Does the user profile's associated group profile have authority to the object? The user will be granted or denied access based on whatever object authorities the group profile has.
6. Does the user profile's associated group profile have authority through an external authorization list that is associated with the object? If present, the user is granted or denied access based on those group authorities in the authorization list.
7. Does the object have *PUBLIC authority sufficient enough to allow the user to access the object? The user is granted or denied access based on the public user's authority to the object.

When encrypting or decrypting data, normal IBM authority checks are used to determine if the user has rights to the Key Store (*VLDL) object, which holds the encryption/decryption Key. If the user has *ALLOBJ, then IBM will indicate the user is authorized, which will allow the user to access the Key to encrypt or decrypt data.

Additionally, when determining if the user is authorized to the full or masked value (on a decryption operation), IBM authority checks are used to check the user's permission to the Authorization Lists that may be specified for the field in the Powertech Encryption for IBM i registry. If the user has *ALLOBJ authority, then IBM will indicate the user is authorized to the Authorization List, which will allow the user to access the full decrypted value.

For best security practices, there should be no users on the system with *ALLOBJ special authority. However, this may not be seen as an acceptable option if legacy customer applications "break" when *ALLOBJ authority is removed. Therefore there are two recommended options for helping customers to deal with *ALLOBJ authority.

OPTION #1 – Move the user's *ALLOBJ authority to a Group Profile

Because of how IBM checks object authority (indicated above), you could move the *ALLOBJ special authority from the individual user profile level to a group profile (which you assign the user to). The user will then get its *ALLOBJ authority from the group profile, giving the user (by default) access to all objects on the system. Then you can *EXCLUDE this user from certain objects such as the Key Stores and Authorization Lists used in Powertech Encryption for IBM i.

Follow the steps below to implement this option:

1. Create a Group User Profile with *ALLOBJ special authority using the CRTUSRPRF command.
2. Using the CHGUSRPRF command, change the individual's User Profile to remove *ALLOBJ special authority and associate them (using the GRPPRF parameter) with the Group Profile created in step 1. The individual will still have *ALLOBJ authority, but only through their Group Profile.
3. Edit the authorities on the Key Store object using the EDTOBJAUT command. Add the individual User Profile (that you want to exclude) to the object and specify *EXCLUDE for the Object Authority.

OPTION #2 – Have Powertech Encryption for IBM i perform its own authority check for *ALLOBJ users

An option is available on the Key Policy level in Powertech Encryption for IBM i, which allows you to indicate how to handle users with *ALLOBJ authority. This option is labeled “Limit all-object authority” with the parameter keyword of LMTALLOBJ. Valid options are *NO and *YES.

If LMTALLOBJ(*YES) is specified at the Key Policy level, then Powertech Encryption for IBM i will perform its own authority check on any requested Key Store or Authorization List for users with *ALLOBJ authority. The user profile (or group profile which it belongs) must be specifically listed as an authority entry (with at least *USE authority) on the Key Store or Authorization List.

KeyStore Authorization Check

1. If LMTALLOBJ(*YES) is specified at the Key Policy level and if the user has *ALLOBJ special authority, then the following steps will be performed to determine if the user has authority to a Key Store (*VLDL) object containing the Key requested for encryption/decryption operations:
2. Does the user profile have an authority entry on the object? If the entry is at least *USE, then the User is authorized to the object. Otherwise if the entry is not at least *USE, then the user will be considered as not authorized.
3. Does the user have a group profile or one or more supplemental group profiles that have an authority entry on the object? If the entry is at least *USE, then the user is authorized to the object. Otherwise if the entry is not at least *USE, then the user will be considered as not authorized.
4. Check the *PUBLIC authority entry on the object. If the entry is at least *USE, then the user is authorized to the object. Otherwise if the entry is not at least *USE and an Authorization List is not attached to the Object, then the user will be considered as not authorized.
5. Is an Authorization List attached to the Key Store object, and is the user profile in that List? If the entry is at least *USE, then the user is authorized to the object. Otherwise if the entry is not at least *USE, then the user will be considered as not authorized.
6. Is an Authorization List attached to the Key Store object, and does the user have a group profile or one or more supplemental group profiles that exist in that List? If the authority is at least *USE for any of the group entries, then the user is authorized to the object. If none of those entries are at least *USE, then the user will be considered as not authorized.
7. If an Authorization List is attached to the Key Store object, then check the *PUBLIC entry in that List. If the entry is at least *USE, then the user is authorized to the object. Otherwise if the entry is not at least *USE, then the user will be considered as not authorized.

Field Registry Authorization List

Authorization Lists can currently be assigned to a field in the Field Encryption Registry to indicate which users are authorized to the full or masked values. If LMTALLOBJ(*YES) is specified at the Key Policy level and if the user has *ALLOBJ special authority and authorization list(s) are specified for the field, then the following steps will be performed to determine if the user is authorized.

1. Does the user profile exist in the Authorization List? If the authority is at least *USE, the user is authorized. Otherwise if the entry is not at least *USE, then the user will be considered as not authorized.
2. Does the user have a group profile or one or more supplemental group profiles, and one or more group profile entries exist in the Authorization List? If the authority is at least *USE for any of the group entries, the user is authorized. Otherwise if the entry is not at least *USE, then the user will be considered as not authorized.
3. Check the *PUBLIC entry in the Authorization List? If the authority is at least *USE, then the user is authorized. Otherwise if the entry is not at least *USE, then the user will be considered as not authorized.

NOTES / RISKS:

- A user with *ALLOBJ authority can change the authority on any object on the system including Key Store (*VLDL) objects and Authorization Lists. So if an *ALLOBJ user becomes restricted by LMTALLOBJ(*YES) in Powertech Encryption for IBM i, they could simply grant themselves authority to the Key Store or the Authorization List. Since you cannot restrict an *ALLOBJ user from changing authorities, the only option you have is to audit authority changes with the CHGOBJAUD command.
- Program adopted authority will not be recognized for users with *ALLOBJ authority when LMTALLOBJ(*YES) is specified. For instance, if an *ALLOBJ user is running a program that adopts authority from the owner of the program, this *ALLOBJ user will not adopt authority to a Key Store if the program owner is authorized to the Key Store.

Appendix B: DB2 Field Procedures

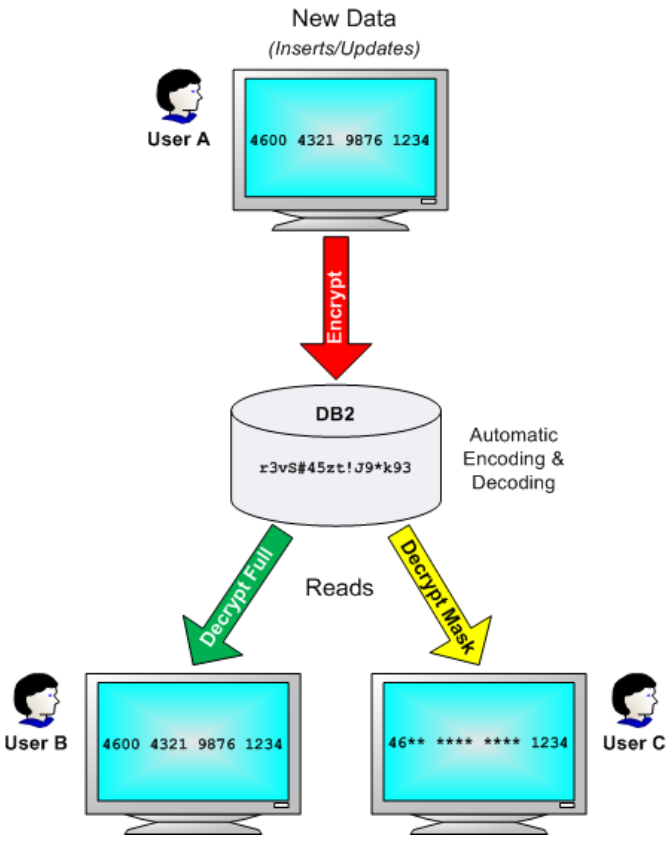
Introduction

DB2 Field Procedures (FieldProcs) were introduced in IBM i version 7.1 in order to help simplify field encryption. A FieldProc can be placed on a database field, which will call a user-specified “exit” program whenever data is read from, inserted into, or updated in the field. FieldProcs are somewhat similar to database triggers; however there are two distinct advantages:

1. FieldProcs allow data to be modified (by the exit program) on a "Read" operation. This will allow the exit program to automatically decrypt the field value before it is returned to the user or application. No application changes are therefore needed to decrypt the data, which can dramatically reduce the implementation time for field-level encryption.
2. FieldProcs provide a separate internal space in the existing file to store the 'encoded' (encrypted) version of the field values. This internal space allows you to encrypt other field types (e.g. numeric, date, time) without having to store the encrypted alphanumeric values in a separate file.

While IBM provided this "hook" into the database with FieldProcs, they leave it up to 3rd party solutions (e.g. Powertech Encryption for IBM i) to create the FieldProc exit programs and perform the encryption/decryption functions.

Powertech Encryption for IBM i simplifies the creation and management of FieldProcs through its innovative Field Encryption Registry commands and screens. It encrypts and decrypts the field values (within the FieldProcs) while providing full integration to Powertech Encryption for IBM i's policy and security controls, key management and audit trails for meeting strict compliance requirements.



Based on authorization lists assigned to the fields, users can be granted access to the fully decrypted field values, restricted to the masked values or can be denied access to any values.

Implementing DB2 Field Procedures

Follow the steps below to add a DB2 Field Procedure (FieldProc) to a database field for encryption:

1. Add the database field to the Powertech Encryption for IBM i Field Encryption Registry using the ADDFLDENC command. Prompt the ADDFLDENC command (in the CRYPTO library) and specify the name of the field, the file it's located in, its length, the encryption key, mask format (if an alpha field), as well as the authorization lists to control access to the decrypted values. Then choose to use a FieldProc by specifying *YES on the USEFLDPROC keyword. Press F1 on any parameter for more help.
   Example of ADDFLDENC command:

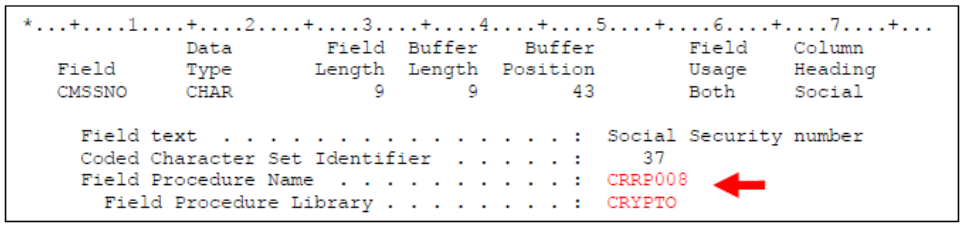
```
CRYPTO/ADDFLDENC
  FLDID(SOCIAL_SECURITY_NUMBER)
  DBFLD(CMSSNO)
  DBFILE(PRDATA/EMPLOYEE)
  DBFLDTYP(*CHAR)
  DBFLDLEN(9)
  ENCKEYLBL(SS_KEY)
  ENCKEYSTR(LIB/KEYSTORE)
  FLDMASK('*****9999')
  AUTLDEC(SSFULL)
  AUTLMASK(SSMASK)
  USEFLDPROC(*YES)
  FLDPROCOPT(*AUTH)
```

   > **NOTE:** The ADDFLDENC command will only add the field entry to the Field Encryption Registry. The FieldProc will not be created until the "activate" process is performed in step 2.

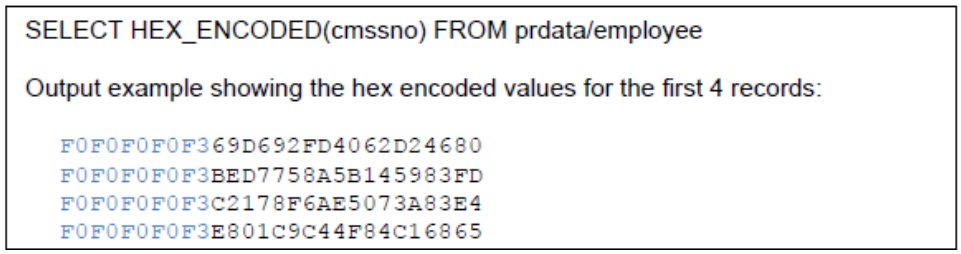
2. When you are ready to add the FieldProc to the field and perform a mass encryption of the existing field values, you should run the ACTFLDENC (Activate Field Encryption) command in batch. *There should be no locks on the file at the time.* This command will add the FieldProc to the file using the SQL "Alter Table Alter Column

Set FieldProc" statement, which will add the internal storage area to the field (for storing the encrypted values) and will then encrypt (encode) the existing field values.

3. To verify that the FieldProc was added to the field, you can perform a DSPFFD command on the file. The field should show the name of the FieldProc in the CRYPTO library. Example:



4. You can verify that the field values were encoded (encrypted) by using the HEX_ ENCODED function in SQL to view the hex values.
Example:



The key identifier occupies the first 5 bytes (in blue) and the encrypted values are in the remaining bytes.

DB2 Field Procedures Facts

Listed below are some things to know about DB2 Field Procedures (FieldProcs):

- FieldProcs are supported for both DDS-described physical files and SQL-defined tables.
- Several fields in the same file can have FieldProcs on them.
- FieldProcs are supported for files with multiple members.
- No one else can be using the file when a FieldProc is being added to it, since an exclusive lock is required to perform the field encoding (encryption).
- Adding a FieldProc will not change the format level identifier on the file, so you do not need to recompile the programs that use that file. It will not generate "level check" errors in the programs.

- If you duplicate a file (e.g. CRTDUPOBJ), it will also duplicate any FieldProcs that are on the file.

DB2 Field Procedure Performance

A DB2 Field Procedure (FieldProc) exit program will be called whenever the field is updated, read or searched (for lookups). Below is a detailed list of events (known at the time of this writing) that will cause a FieldProc exit program to be called:

Events that call the FieldProc program to encrypt (encode) the field:

- When a new record is added to the file, regardless of the program or method used (e.g. RPG, COBOL, SQL, DFU, etc.).
- When a record is updated, even if the field value does not change.
- When a SQL lookup needs to be performed against a field (which is encrypted with a FieldProc). For example, if a user issues the SQL statement of
  Select NAME where SSNO = '508998888'
  then the FieldProc will be called to encrypt the lookup value of '508998888' so it can then search the encrypted SSNO field with that value.
- When a lookup needs to be performed on a keyed field (which is encrypted) using record-level operations (e.g. SETLL , SETGT, CHAIN, READE). For example, if the user issues the RPG operation of
  SSNO CHAIN EMPMAST
  then the FieldProc will be called to encrypt the key field SSNO so it can look up the field with that encrypted value.

Events that call the FieldProc program to decrypt (decode) the field:

- When a native record-level read of that field is performed from RPG (e.g. READ, READE, CHAIN) and COBOL.
- When the field is read in SQL Select and Fetch statements.
- When reading the field in a Query or through a Report Writer.
- When the field is downloaded through a File Transfer utility (e.g. Client Access, Surveyor/400).
- When reading the file with a CL command like DSPPFM or CPYF (if not creating the file).

IMPORTANT: For any of these events (listed above), CPU cycles will be consumed to call the FieldProc exit program(s). If a field (containing a FieldProc) is used frequently throughout your applications, these repeated program calls may increase response times for interactive jobs and extend the duration of batch jobs. Performance will be further impacted if the file contains multiple fields with FieldProcs.

DB2 Field Procedure Cautions

Before implementing DB2 Field Procedures (FieldProcs) in a production environment, you should be aware of the potential problems (as listed below) that may arise from their usage. It is also highly recommended to apply the latest PTFs from IBM.

Logical and Keyed Physical Files – Not Sorted Correctly

If the field (having a FieldProc) is a key on a logical or physical, then it will be sorted by the encoded (encrypted) value on any READ operations. For instance, if the records are normally read in key order of 1,2,3,4,5, the FieldProc may cause them to be read in the order of 5,1,3,2,4. These “unordered” results may cause problems in your applications if they are expecting it to be sorted by the decrypted values.

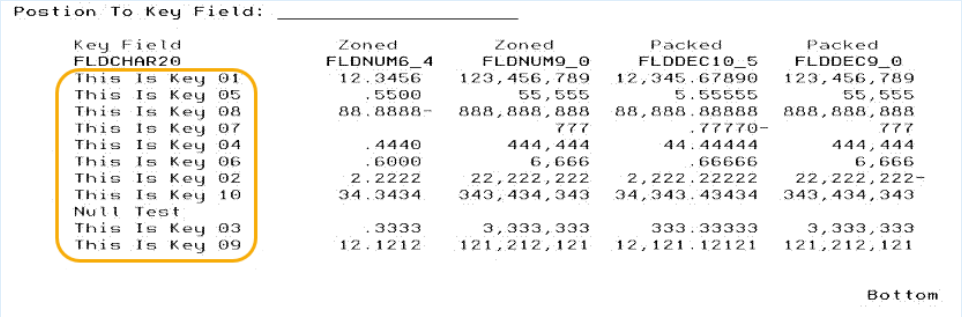
Solution A (recommended)

Fortra offers a utility to assist with encrypted key sorting when processed by native I/O (SETLL/READ).

Powertech Encryption for IBM i includes two commands that can be strategically embedded within your applications that provide a solution to the encrypted key sorting issue: STRRLASRT and ENDRLASRT. The commands can be applied against all native DB2 I/O operations. They offer a universal solution for RPG and COBOL applications, as well as other programs that use native I/O as opposed to SQL to access data.

1. Call **STRRLASRT** just prior to opening the file that the SETLL/READ will be executed on.
2. Call **ENDRLASRT** just after closing the file that the SETLL/READ was executed on.

EXAMPLE:
If STRRLASRT is not called prior to the RPG program that builds a subfile listing from the file with the encrypted key field, records will be out of sequence.

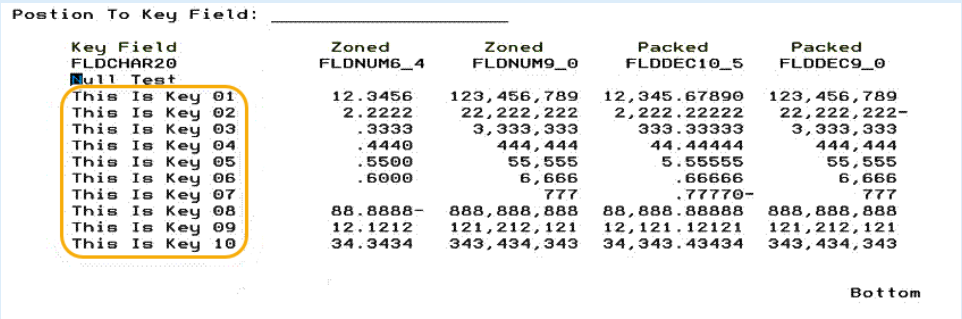


```
Postion To Key Field: _____________________

     Key Field              Zoned        Zoned          Packed         Packed
     FLDCHAR20          FLDNUM6_4     FLDNUM9_0     FLDDEC10_5     FLDDEC9_0
     This Is Key 01       12.3456   123,456,789   12,345.67890   123,456,789
     This Is Key 05         .5500        55,555         5.55555        55,555
     This Is Key 08      88.8888-   888,888,888   88,888.88888   888,888,888
     This Is Key 07                        777         .77770-           777
     This Is Key 04         .4440       444,444        44.44444       444,444
     This Is Key 06         .6000         6,666          .66666         6,666
     This Is Key 02        2.2222    22,222,222     2,222.22222    22,222,222-
     This Is Key 10       34.3434   343,434,343    34,343.43434   343,434,343
     Null Test
     This Is Key 03         .3333     3,333,333       333.33333     3,333,333
     This Is Key 09       12.1212   121,212,121    12,121.12121   121,212,121

                                                                        Bottom
```

From the command line prior to calling RPG program that builds subfile listing from file with encrypted key field, call the command:

STRRLASRT P_FILE(*TESTTBL*)
P_OPTION(*INP)

CALL PGM(ENCDEMOR)



```
Postion To Key Field: _____________________

     Key Field              Zoned        Zoned          Packed         Packed
     FLDCHAR20          FLDNUM6_4     FLDNUM9_0     FLDDEC10_5     FLDDEC9_0
     Null Test
     This Is Key 01       12.3456   123,456,789   12,345.67890   123,456,789
     This Is Key 02        2.2222    22,222,222    2,222.22222    22,222,222-
     This Is Key 03         .3333     3,333,333       333.33333     3,333,333
     This Is Key 04         .4440       444,444        44.44444       444,444
     This Is Key 05         .5500        55,555         5.55555        55,555
     This Is Key 06         .6000         6,666          .66666         6,666
     This Is Key 07                        777         .77770-           777
     This Is Key 08      88.8888-   888,888,888   88,888.88888   888,888,888
     This Is Key 09       12.1212   121,212,121   12,121.12121   121,212,121
     This Is Key 10       34.3434   343,434,343   34,343.43434   343,434,343

                                                                        Bottom
```

The key field is sorted without modifying the application's native i/o processing of SETLL/READ.
When the program and file closes, issue the command:

ENDRLASRT P_FILE(TESTTBL)

> **NOTE:** The command can be called within a CLP just prior to the RPG call or within the RPG itself as long as the command is called prior to when the file that is being overridden is opened.

> **NOTE:** ENCDEMOR is a program that simply displays the usage of the commands included within these instructions, and is not part of Powertech Encryption for IBM i.

> **WARNING:** STRRLASRT is not recommended for use with large files where system performance is a concern. See the tip below for best performance.

TIP: For best performance it is recommended to use Solution B (below) and subset the selection criteria for the data needed by using a 'where' clause on a non-encrypted field then using the 'order by' clause to sort on the encrypted field.
For example:
Select column_1, key_1 from crypt.testkeys where column_1 like 'TEST%' order by key_1;
(where column_1 is an unencrypted field and key_1 is the encrypted key field that needs to be sorted).

Solution B

Alternatively, to sort the records properly (by their decrypted values), you can use an SQL SELECT statement with the ORDER BY clause on the field(s) you want to sort. The ORDER BY clause will call the associated FieldProc exit program(s) to decrypt (decode) the values for proper sorting.

Changing the Current User of the Job

When you need to change the authority for a user in order to allow them to see the full value of a field when they are normally only authorized to the masked value (or not authorized), you must change the Current User of the job. The CRRP015 program can be used to change the current user and notify the Field Procedure that the current user has changed.

NOTE: The Field Procedure programs are called in a secondary thread most of the time and adopt authority will not work.

Using the CRRP015 program

- The Program CRRP015 is compiled with User profile *OWNER.
- The owner of the CRRP015 program must be changed to a user that has *USE authority to the newly swapped User Profile.
- The newly 'swapped to' User Profile must be on the 'Full Authorization List' for whichever field entry is being used.

CRRP015 Parameters:

Name	Description	Type	Length	In/Out	Required
Process	Plain Text	Alpha	10	In	Yes
UserId	Current User	Alpha	10	In	Yes
MsgId	Message Id	Alpha	7	Out	
MsgText	Message Text	Alpha	80	Out	

Name	Description	Type	Length	In/Out	Required
Errors	Errors occurred (Y=Yes)	Alpha	1	Out	

Example program call to swap the current user:

```
C* Change the current user
C           CALL 'CRRP015'
C           PARM '*SWAP'      PROCESS          10
C           PARM 'RBYRD'      UserId           10
C           PARM              MSGID             7
C           PARM              MSGTEXT          80
C           PARM              ERRORS            1
C* Success-> No errors occurred. Check if External Index was found
(not 0)
C IF ERRORS <> 'Y'
C … Success logic …
C ELSE
C* Errors-> Display MSGID and MSGTEXT
C … Error logic …
C ENDIF
```

Example program call to reset the current user back:

```
C* Change the current user
C           CALL 'CRRP015'
C           PARM '*RESET'     PROCESS          10
C           PARM ' '          UserId           10
C           PARM              MSGID             7
C           PARM              MSGTEXT          80
C           PARM              ERRORS            1
C* Success-> No errors occurred. Check if External Index was found
(not 0)
C IF ERRORS <> 'Y'
C … Success logic …
C ELSE
C* Errors-> Display MSGID and MSGTEXT
C … Error logic …
C ENDIF
```

Example program call to release the profile handles:

There is no need to release the handles until finished.

There is a limit in a job of how many profile handles you can create. The maximum number of profile handles that can be created in a job is approximately 20,000.

```
C* Change the current user
C           CALL 'CRRP015'
C           PARM '*RELEASE'   PROCESS          10
C           PARM ' '          UserId           10
C           PARM              MSGID             7
C           PARM              MSGTEXT          80
C           PARM              ERRORS            1
C* Success-> No errors occurred. Check if External Index was found
(not 0)
C IF ERRORS <> 'Y'
C … Success logic …
C ELSE
C* Errors-> Display MSGID and MSGTEXT
C … Error logic …
C ENDIF
C
C* Do not set on LR
C RETURN;
```

LF and SQL View Limitations

- Within an LF (DDS created), a Select/Omit statement related to an encrypted field is not allowed.
- In an SQL View, a WHERE clause that conditions the View on the value of an encrypted field is not allowed. This is due to the fact that the Select/Omit, or the WHERE clause, is based on a decoded value, and the data at rest in the file is encoded after encryption. An example would be a situation in which the field SSN must be encrypted, but the view is conditioned on ‘WHERE SSN <> 0’.
- There is a potential for a delay when fully processing certain SQL requests, since Powertech Encryption for IBM i must decode the entire file prior to being able to retrieve the data. For example, this may be the case while performing an SQL statement such as the following:

  SELECT title FROM books WHERE title LIKE ‘%RPG’

  In this case, if the field ‘title’ is encrypted, and there are 2 million records in the table, all 2 million values of ‘title’ must be decrypted before Powertech Encryption for IBM i can know what is to be included in the selection. So, the execution of this SQL statement, post-encryption, could be far more time consuming than it was pre-encryption.

- Be sure to thoroughly test your SQL when implementing Encryption and Field Procedures.

Join LF Considerations

When a logical file is joining based on an Encrypted Key Field, you must first remove the Join LF, then Encrypt both physical files. Then you are able to recreate the Join LF over the top of the newly encrypted physical files. Within a Join LF, the JDFTVAL keyword is not supported if you are joining on an Encrypted field.

LF, PF, and SQL-created Table Limitations

When a key field on a file/table is being encrypted, the following attribute keywords are not valid: ALTSEQ, ABSVAL, ZONE, or DIGIT.

CHGPF – Dropping FieldProcs

The CHGPF command (using the parameters of SRCFILE and SRCMBR) has been commonly used to add new fields or change existing field definitions on files. Unfortunately, this use of CHGPF will drop any existing FieldProcs on the file and return any encoded (encrypted values) back to their decrypted state, which may cause the unintended exposure of sensitive data in the file.

Solution: In order to add a field to a file or change an existing field definition, it is recommended to instead use the SQL ALTER TABLE statement since it will retain any existing FieldProcs. Before you use ALTER TABLE on a DDS-described file, you should convert the file's DDS syntax to SQL DDL syntax using a tool like Surveyor/400^{tm} from Fortra.

Duplicating Files – Field Encryption Registry Library

Since the Field Encryption Registry library name is "hard-coded" into any fields with FieldProcs, then duplicating the file (using CRTDUPOBJ or CPYF CRTFILE(*YES)) into another environment will also copy these hard-coded library names into the new file. This may cause the new file's FieldProcs to point to the Registry in the wrong environment library, causing unpredictable results.

Solution: First create the file in the target environment from scratch using DDS or SQL. Then add the field to the Field Encryption Registry in the target environment and activate the field. Then use the CPYF command to copy the data from the file in the source environment to the target environment. The encode and decode functions will run for each record and field with FieldProcs.

Appendix C: Adding a Client Certificate to an External Key Manager

A Client Certificate is required when using KMIP. Use the following instructions (for External Key Manager you are using) to create and add a Client Certificate.

IBM Security Key Lifecycle Manager

1. Create a Client Certificate in the Digital Certificate Manager (DCM) and Export the Certificate Authority. See Appendix D: Creating a Certificate using the Digital Certificate Manager (DCM) for details.

   > **NOTE:**
   > You may need to convert the certificate file and its contents to ASCII:
   > `dd conv=ascii  if=PWR742_CA.crt of=PWR742_CA_ascii.crt`

2. Send the Certificate Authority to the person in charge of the SKLM server so they can import the certificate and associate it with your user id.
3. In the Digital Certificate Manager, create an Application that points to the Certificate.
4. Enter the Application ID in the External Key Manager entry.

Safenet

1. Create and export a Certificate request in the Digital Certificate Manager (DCM). The user ID that is created for you on the Safenet server must be in the Common name of the certificate. See Appendix D for details.
2. Send the certificate request to the Safenet server.
3. Have the person responsible for the Safenet server sign the certificate with a CA certificate on the Safenet machine , export the signed request from the server, and send it to you.

   Also, export the CA certificate that was used to sign the certificate. The CA certificate needs to be installed on the Digital Certificate Manager (DCM) first, before the Certificate Response.

4. Import the newly signed certificate. Server or Client.
5. In the Digital Certificate Manager, create an application that points to the Certificate.
6. Enter the Application ID in the External Key Manager entry.

Appendix D: Creating a Certificate using the Digital Certificate Manager (DCM)

Use the following instructions to create and export a Digital Certificate (also called a Certificate Authority) using IBM's Digital Certificate Manager (DCM).

Creating a Certificate Signed by a Local Certificate Authority (CA)

1. Open the Digital Certificate Manager by going to http://your server name:2001/QIBM/ICSS/Cert/admin/qycucm1.ndm/main0.

   EXAMPLE:
   http://*your server name*:2001/QIBM/ICSS/Cert/admin/qycucm1.ndm/main0

   NOTE: To open this URL, the http server must be running on your IBM i system. To start the http server, use the following command:

   ```
   STRTCPSVR SERVER(*HTTP) HTTPSVR(*admin)
   ```

2. Click **Select a Certificate Store**.
3. Choose ***SYSTEM** and click **Continue**.
4. Enter the Certificate Store password and click **Continue**.
5. Choose **Create Certificate** from the list on the left.
6. Choose **Server or Client Certificate** and click **Continue**.
7. Choose **Local Internet Certificate Authority (CA)** and click **Continue**.
8. Enter the requested information and click **Continue**.
9. Send the Certificate Authority to the person in charge of the SKLM server. See IBM Security Key Lifecycle Manager (SKLM) in Appendix C: Adding a Client Certificate to an External Key Manager for details.

Creating a Certificate Signed by VeriSign or Another Internet Certificate Authority (CA)

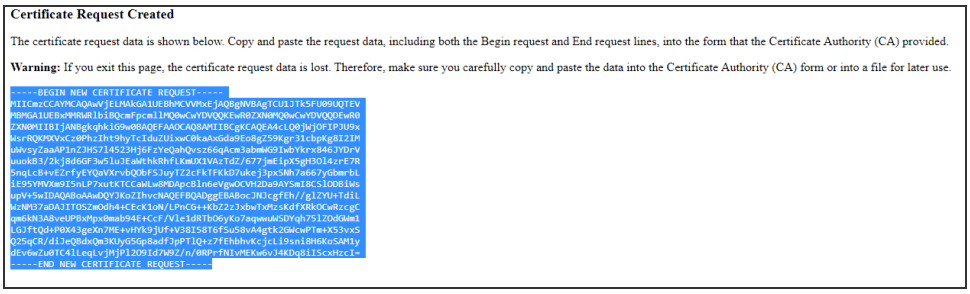
1. Open the Digital Certificate Manager by going to http://your server name:2001/QIBM/ICSS/Cert/admin/qycucm1.ndm/main0.

   EXAMPLE:
   http://*your server name*:2001/QIBM/ICSS/Cert/admin/qycucm1.ndm/main0

   NOTE: To open this URL, the http server must be running on your IBM i system. To start the http server, use the following command:

   ```
   STRTCPSVR SERVER(*HTTP) HTTPSVR(*admin)
   ```

2. Click **Select a Certificate Store**.
3. Choose ***SYSTEM** and click **Continue**.
4. Enter the Certificate Store password and click **Continue**.
5. Choose **Create Certificate** from the list on the left.
6. Choose **Server or Client Certificate** and click **Continue**.
7. Choose **VeriSign or Other Internet Certificate Authority (CA)** and click **Continue**.
8. Enter the requested information and click **Continue**.
9. Copy and paste the data, including both the Begin request and End request lines, into a file and send it to the server.



10. Import the signed certificate back in. See Safenet or Vormetric in Appendix C: Adding a Client Certificate to an External Key Manager for details.

Appendix E: Controlling Access using Authorization Lists

Powertech Encryption for IBM i includes authorization lists that allow you to control access to the product. This feature is off by default.

The following instructions describe how to activate and configure Powertech Encryption for IBM i's authorization lists so that only authorized users are granted access to Powertech Encryption for IBM i.

PCRADMIN

Most Powertech Encryption for IBM i commands are protected by the PCRADMIN authorization list. This authorization list controls access to the product menus and commands. It is shipped with *PUBLIC *USE.

To use the PCRADMIN authorization list protection:

1. Use the command **WRKAUTL PCRADMIN** to display the Work with Authorization Lists panel.
2. Enter option **2** (Edit) for PCRADMIN to open the Edit Authorization List panel.
3. Change the Object Authority of the *PUBLIC user to *EXCLUDE.
4. Press **F6** and specify the user profile(s) to be granted access. Set added users to at least *USE authority.

> **NOTE:** If you set up PCRADMIN with *PUBLIC as *EXCLUDE, when a user is added as a Key Officer, the user profile is automatically added to the PCRADMIN authorization list with *USE authority. When the user is removed from the Key Officers, the profile is removed from the PCRADMIN authorization list.

To manually add a user to the PCRADMIN authorization list, use the command:

ADDAUTLE AUTL(PCRADMIN) USER(*MYUSER*) AUT(*USE)

> **NOTE:** In addition to being authorized to the PCRADMIN authorization list, product administrators must also have either *SECADM special authority or be entered as a Key Officer within Powertech Encryption for IBM i.

> **NOTE:** The backup encryption commands are not controlled by the use of an authorization list.

PCRREPORT

The commands to print reports in Powertech Encryption for IBM i are protected by the PCRREPORT authorization list.

> **NOTE:** A user with *ALLOBJ authority is able to run the reports without special authorization.

To use the PCRREPORT authorization list to control access to the print commands:

1. Use the command **WRKAUTL PCRREPORT** to display the Work with Authorization List panel.
2. Enter option **2** (Edit) for PCRREPORT to open the Edit Authorization List panel.
3. Change the Object Authority of the *PUBLIC user to *EXCLUDE.
4. Press **F6** and specify the user profile(s) to be granted access. Set added users to *USE and Object operational authority.

To manually add a user to the PCRREPORT authorization list, use the command:

ADDAUTLE AUTL(PCRREPORT) USER(*MYUSER*) AUT(*USE)

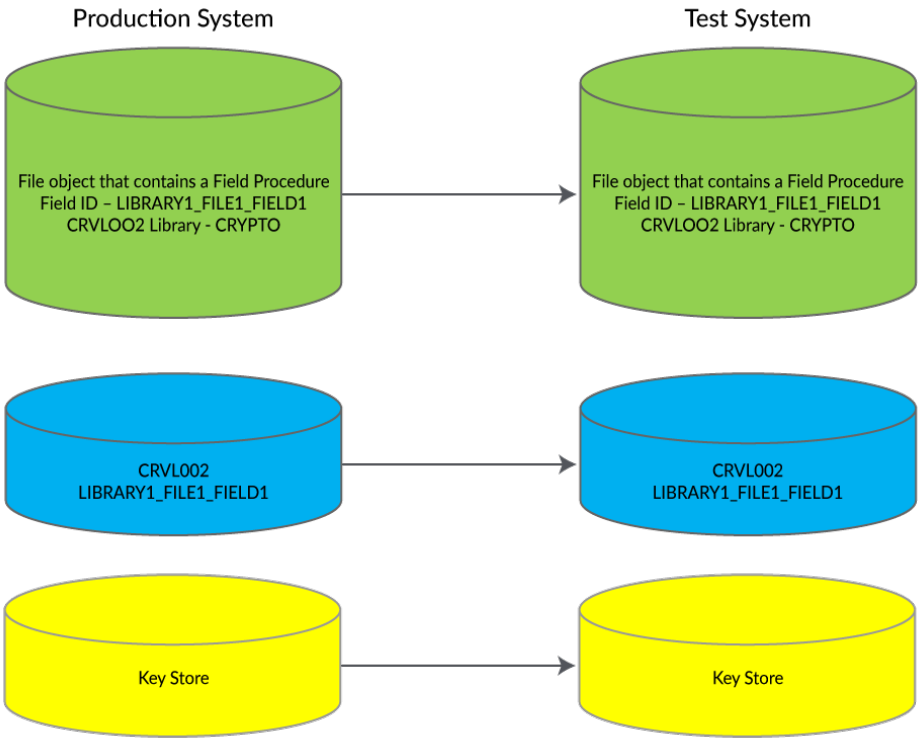
Users may also need 'Object Exists' Object Authority. To grant this:

1. Use the **WRKAUTL PCCRREPORT** command to display the Work with Authorization List panel.
2. Enter option **2** (Edit) for PCRREPORT to open the Edit Authorization List panel.
3. Use **F11** to display the object authority options.
4. Mark the 'Exist' column with an 'X' to select it.

Appendix F: Copying Files from Production to Test Environments (Field Procedures)

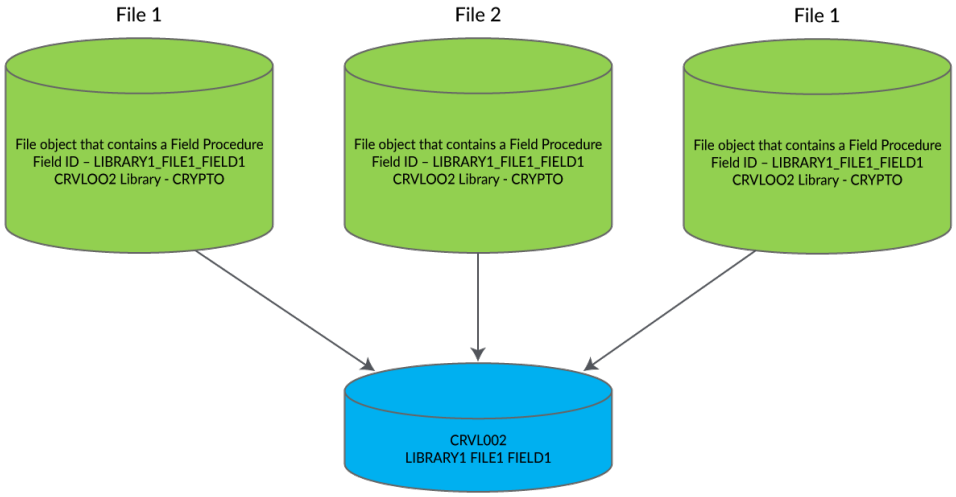
When a field is encrypted in a file using Field Procedures, the Field Identifier and Library that holds the Field Registry (CRVL002) are saved into the file. This tells Powertech Encryption for IBM i in which library to find the Field Registry (CRVL002) and which entry to use in the registry to get the information about the encryption, such as key stores, masking information, authorization lists, which method was used to encrypt (Field Procedures with AUTH method), and the library where the file exists.

1. When the file is saved and then restored to another system:
    a. The Field Registry (CRVL002) must exist in the Library that is saved with the File.
    b. An entry in the Field Registry (CRVL002) must exist with the same name that was stored in the file object.
    c. The entry saves the Key Store name and Library for both Encryption and Decryption Keys. The Keys must be the same Key that was created on the source machine. WARNING: You cannot just create a new random generated Key with the same label as what was used on the source machine. This generates a new key that will not decrypt the data correctly.



2. When a file is copied using either CRTDUPOBJ or CPYF with parameter CRTFILE (*YES) the two values stay with the file. The new object will use the same entry in the Field Registry (CRVL002) that the original file uses. If the entry has a status of *ACTIVE the encryption and decryption of the field will work normally in the new file.
    a. WARNING: If the original file is deactivated and the status is changed to *INACTIVE the Field Procedures will still exist on the newly created file and the encryption and decryption will not work on the file.

b. WARNING: The new file cannot be deactivated.



3. Copying a file from one file to another with an encrypted field *not using* CPYF WITH CRTFILE(*YES):

   When you read the field you get either the Fully decrypted value, the partially masked data, or the fully masked data depending on what you are authorized to.

   a. If you are authorized to the full value, you may get a value like ‘8765238790874325’. This value would be inserted into the destination file.
   b. If you are authorized to a partially masked value, you may get a value like ‘XXXX23879087XXXX’. This value would be inserted into the destination file.
   c. If you are authorized to a fully masked value, you may get a value like ‘XXXXXXXXXXXXXXXX’. This value would be inserted into the destination file.

   If the field in the destination file is encrypted, the value you read out of the source file gets encrypted as it is written to the destination file.

Appendix G: Decryption Accelerator Prerequisites and Limitations

Prerequisites

To take advantage of the Decryption Accelerator functionality, your systems and files must meet the following requirements:

- If the file that data is being encrypted in uses triggers, the triggers must be "secure triggers." To view your triggers and verify if they are "secure triggers," use the following SQL query:

SELECT * FROM QSYS2/SYSTRIGGER

The SECURE column will have a value of **Y** if the trigger is secure.

- The IBM Licensed Program "Advanced Data Security for i" must be installed on the system to prevent any loss of functionality when the file objects that contain encrypted data are duplicated. If you do not have this program installed, you can bypass this requirement by prompting the CRTDUPOBJ command, then setting the "Duplicate access control" parameter to *NONE. Additionally, if you use the CPYF command and set the "Create File" parameter to *YES for this duplication, you do not need to have this licensed program installed.
- Specific PTFs are needed to use the Decryption Accelerator feature. See the PTFs listed under the System Requirements in the *Powertech Encryption for IBM i Installation Guide*.

Decryption Accelerator and High Availability Environments

When the Decryption Accelerator is used in a High Availability, journal-based environment, you must specifically configure the user profile who applies journal changes on the High Availability system. For more details, see High Availability Setup.

Limitations

If the Decryption Accelerator is used on an encryption registry entry, the file/table of that entry cannot be duplicated into library QTEMP, unless you prompt the CRTDUPOBJ command, then set "Duplicate access control" to *NONE. If you use the CPYF command with the "Create File" parameter set to *YES, you can duplicate files into QTEMP, even with the Decryption Accelerator turned on.

Contact Powertech Support with questions.

Appendix H: Live Partition Mobility (LPM)

A live partition mobility (LPM) migration is the process of moving a single active partition to a different system without disrupting the workload activity of the partition.

> **IMPORTANT:** Before you run the LPM process, ensure you have valid Powertech Encryption license keys for both the source and target partitions in place.

For your field procedure program to work correctly, Powertech Encryption will use the exit points introduced in version 7.1, QIBM_QWC_SUSPEND and QIBM_QWC_RESUME, and register exit programs to each exit point, as follows:

- QIBM_QWC_SUSPEND (CRRP660)

- QIBM_QWC_RESUME (CRRP665)

> **NOTE:** If the build is not successful, contact Powertech Support.

> **TIP:** Users signed on to the IBM i who have accessed the field procedure program prior to starting the migration program will not be affected by the migration process.

Testing LPM with Powertech Encryption

Once a system administrator has set up LPM, it can be tested to see how and if it works successfully. Use the instructions below to register new exit programs and ensure users can access encrypted data before starting and after finishing the LPM testing.

1. Register the following exit programs:
   a. ADDEXITPGM EXITPNT(QIBM_QWC_SUSPEND)
      FORMAT(SSPS0100)
      PGMNBR(1)
      PGM(CRYPTO/CRRP660)
      THDSAFE(*UNKNOWN)
      TEXT('Fortra PT Encryption for LPM')
   b. ADDEXITPGM EXITPNT(QIBM_QWC_RESUME)
      FORMAT(RSMS0100)
      PGMNBR(1)
      PGM(CRYPTO/CRRP665)
      THDSAFE(*UNKNOWN)
      TEXT('Fortra PT Encryption for LPM')
2. Before starting the LPM testing, have 1-2 users do the following:
   a. Sign on and stay signed on.
   b. Select * from yourLib.yourEncryptedFile.
3. Start the LPM testing.

   > **NOTE:** Before the partition is suspended or moved, the QIBM_QWC_SUSPEND exit program fires and saves important information that is required when QIBM_QWC_RESUME fires in step 5a below.

4. During the LPM testing, occasionally run the SQL statement in step 2b to ensure there are no issues.
5. Verify the LPM testing completes by noting the following events:
   a. After the move completes, the QIBM_QWC_RESUME exit points fire.
6. For the users who signed on in step 2, do the following:

a. Run the SQL statement in step 2b.

7. Sign on with a new user.
8. Run the SQL statement in step 2b.
9. See if the LPM testing was successful by how the data returns:
   - **Successful:** If the data returns correctly based on authorization defined in the Powertech Encryption product, and the following messages appear in the QSYSOPR msgq:

```
From . . . :  QPGMR       10/19/22  14:03:49
LPAR Migration Suspend: Phase 1 - Ok to proceed.
Partition suspend request in progress.
From . . . :  QPGMR       10/19/22  14:05:10
LPAR Migration Suspend: Phase 2 - Serial number:  7851930 saved.
Partition resumed after migration.
From . . . :  QPGMR       10/19/22  14:05:53
LPAR Migration Resume: Phase 4 - Successfully replaced CRVL001.
```

   - **Unsuccessful:** If the data does not return as expected. Review and send the joblog to Powertech Support for evaluation and help.

Contacting Fortra

Please contact Fortra for questions or to receive information about Powertech Encryption for IBM i. You can contact us to receive technical bulletins, updates, program fixes, and other information via electronic mail, Internet, or fax.

Fortra Portal

For additional resources, or to contact Technical Support, visit the Fortra Community Portal at https://community.fortra.com.