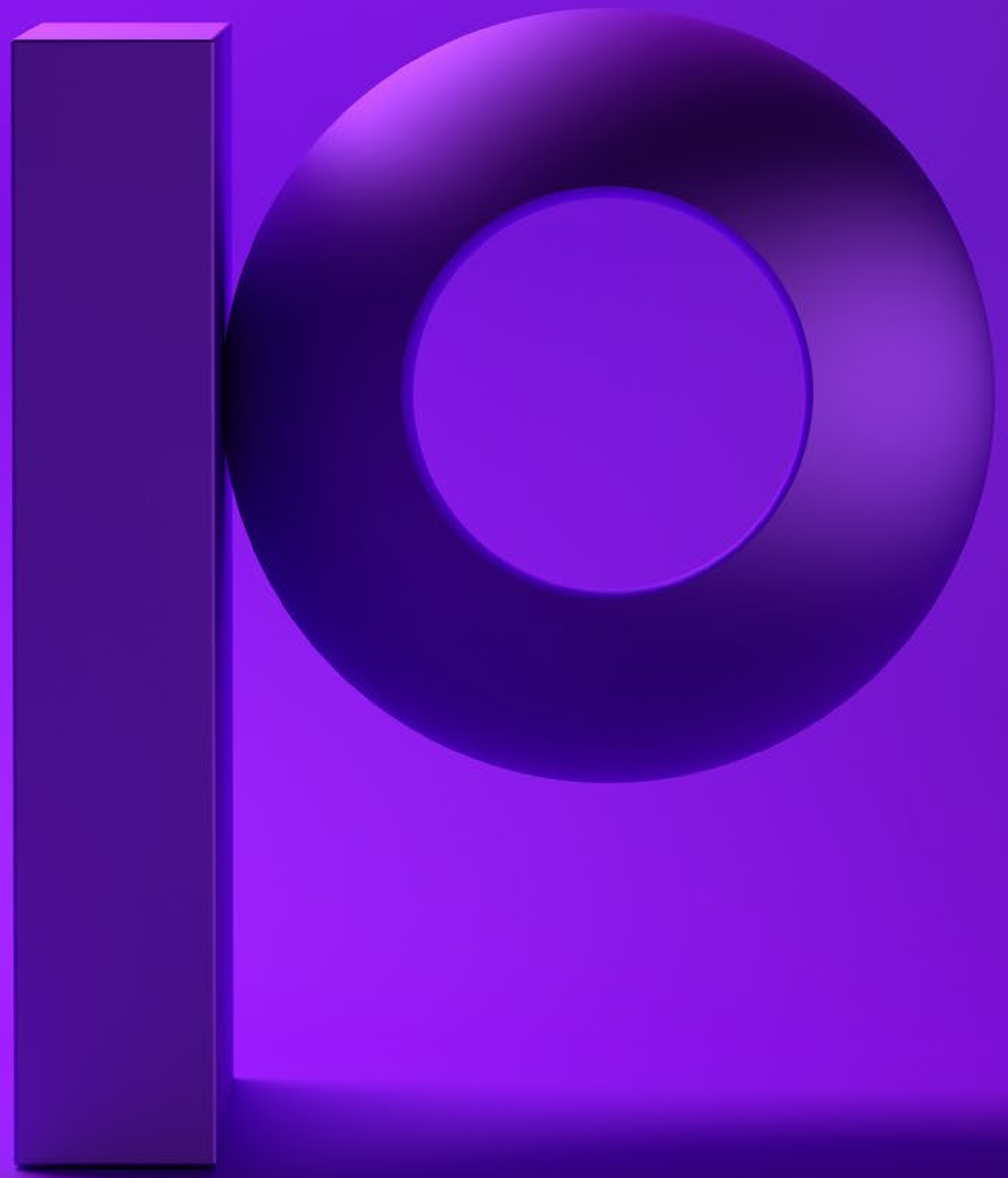
precisely

IBM i Encryption with FieldProc and Assure Encryption

Protecting Data at Rest



Abstract

Organizations of all sizes are rushing to implement encryption to protect sensitive digital assets including personally identifiable information of customers, vendors, and employees, and valuable intellectual property. IBM introduced a Db2 database column level exit point named Field Procedures, or FieldProc, in release 7.1 of the IBM i operating system. This column-level exit point is implemented directly in the Db2 database and is invisible to applications (both IBM and user) that use the database. Customers and third-party vendors are developing FieldProc exit point software to provide the encryption, key management, user control, audit and data masking features that IBM i customers need to protect sensitive data. This eBook explores the architecture, implementation, critical features and limitations of Field Procedures on the IBM i and points to some solutions to the primary challenges. Where appropriate this eBook points to Precisely's security solutions for FieldProc on the IBM i and describes how these challenges are being met.

FieldProc Architecture & Implementation

FieldProc Architecture

FieldProc is a type of column-level exit point that is implemented directly in the Db2 database. As is typical with any of the other IBM i exit points, IBM provides the architecture for the exit point to invoke a user application, but IBM does not provide that application. Customers or vendors can create a FieldProc application based on the documented architecture of the exit point. Precisely is one vendor who provides such software.

A Note on Terminology:

The term "column" is used in this paper when referring to a field in a file, and the term "table" is used to refer to a physical file. For the purposes of discussing FieldProc, these terms are interchangeable, and the more modern terms "column" and "table" will be used. In a similar way the term "index" is used to refer to a column that is either a primary or secondary key. A secondary key would include a key defined in a logical file.

The exit point architecture is very simple. There are only two commands and three functions that are supported. The two commands are:

- Start FieldProc
- End FieldProc

The three functions that are handled by a FieldProc program are:

- Initialization
- Encode (Encrypt)
- Decode (Decrypt)

When FieldProc is started on a column, the FieldProc program is called for Initialization, and then called for each row in the table to provide for the encryption of the column. When FieldProc is ended, the FieldProc program is called for each row to decrypt the data. All other normal read, change, and insert operations call the FieldProc program to provide for encryption or decryption as needed. FieldProc is not invoked for a delete operation on a row.

FieldProc Implementation

FieldProc is a SQL feature and is implemented through SQL commands. That is, FieldProc is not implemented through DDS, but only through SQL. Through SQL you create or alter a table to have the FIELDPROC attribute like this:

```
ALTER TABLE employee alter column
salary set FieldProc Userpgm
```

Likewise, you can remove a FieldProc attribute using a SQL DROP statement like this:

```
ALTER TABLE employee alter column
salary Drop FieldProc
```

As noted above, when you start FieldProc with the CREATE TABLE or ALTER TABLE commands, the effect is immediate. Starting FieldProc causes the FieldProc program to be called for each row in the table to perform encryption. Dropping the FieldProc attribute of a column causes the FieldProc program to be called for each row to decrypt the data.

Certain operations will cause FieldProc to be invoked even if the column data is not being used. For example, a legacy RPG program might read a file from beginning to end but not use a particular column that is under FieldProc control. Even though the column is not used in the RPG program, FieldProc will be invoked to decrypt the value of the column for each row. Note that a native SQL SELECT statement that does not include the encrypted column will NOT invoke FieldProc for decryption.

Some work management operations are difficult with the current IBM implementation of FieldProc. For example, save and restore operations when the restore library is different than the save library can be problematic. Additionally, change management operations where there are changes to column attributes can be difficult to manage and require decrypting the database, applying the change, and then re-encrypting the database. These limitations are especially true in legacy RPG applications.

Supported Field (Column) Types

Most column types are support for FieldProc including character, numeric, date, time, and timestamp fields. You can use FieldProc on null-capable fields and double-byte character fields. Some type of derived and counter fields do not support FieldProc, but IBM i customers will find that all normal application fields are supported. With FieldProc there is no need to change the field size or attribute, nor is there any need to change or re-compile applications.

Legacy DDS Files

Many IBM i customers have the impression that legacy DDS files will not support FieldProc encryption. This is not true! You can readily implement FieldProc encryption on files created with DDS, and it is not necessary to convert them to DDL and SQL tables. As IBM notes, there are some advantages to doing so, but it is not necessary and the large majority of FieldProc users continue to use DDS files. As noted above, you can only start and stop FieldProc using SQL commands, but these SQL commands work fine on DDS-created files.

Many IBM i customers have the impression that legacy DDS files will not support FieldProc encryption. This is not true!

Encrypting Multiple Fields in One File

Db2 FieldProc control can be started on multiple columns in one database table. There is no practical limit on the number of columns you can select for FieldProc implementation. Of course, there are performance implications for encrypting multiple columns as the FieldProc program will be called independently for each column under FieldProc control. But many IBM i customers place multiple columns in a table under FieldProc encryption control. In some cases customers place hundreds of columns under FieldProc control. FieldProc's support for multiple columns includes columns that are Primary or Secondary keys to the data. Please see the following sections for a discussion of limitations related to encrypted indexes and legacy RPG applications.

Encryption

Encryption in FieldProc

It goes without saying that your FieldProc application will need to use an encryption library to perform encryption and decryption operations. IBM provides an encryption software library as a native part of the IBM i operating system. It is available to any customer or vendor who needs to implement encryption and decryption in their FieldProc programs.

Unfortunately, the native IBM i encryption library is very slow. This might not be noticeable when encrypting or decrypting a small amount of data, but batch operations can be negatively impacted. The advent of AES encryption on the Power8 processor did little to mitigate the performance issue with encryption. IBM i customers and third party vendors of FieldProc solutions should use caution when implementing FieldProc using the native IBM i AES software libraries. They are undoubtedly accurate implementations of AES encryption, but suffer on the performance front.

Unfortunately, the native IBM i encryption library is very slow. The advent of AES encryption on the Power8 processor did little to mitigate the performance issue with encryption.

Performance and Assure Encryption

Precisely's Assure Encryption FieldProc solution relies on NIST-validated AES encryption libraries. Its optimized AES encryption is more than 100 times faster than the encryption software library in the IBM i OS on Power6 and Power7 processors, and more than 50 times faster on Power8 processors with AES encryption on the chip. This provides a significant performance advantage to customers with larger Db2 data sets.

Key Management

Encryption Key Management

An encryption strategy is only as good as the key management strategy, and it is difficult to get key management right. For companies doing encryption, the most common cause of an audit failure is improper implementation of key management. Here are a few core concepts that govern a good key management strategy:

- • Encryption keys should not be stored on the same systemas the sensitive data they protect.
- • Security administrators of the key management solution should have no access to the sensitive data, and database administrators should have no access to encryption key management (Separation of Duties). On the IBM i system this means that security administrators such as QSECOFR and any user with All Object (*ALLOBJ) should not have access to data encryption keys or key encryption keys.
- • More than one security administrator should authenticate before accessing and managing keys (Dual Control).
- • All access to encryption keys should be logged and audited. This includes use of encryption keys as well as management of keys.
- • Encryption keys should be mirrored/backed up in real time to match the organization's standards for system availability.

Encryption Key Caching

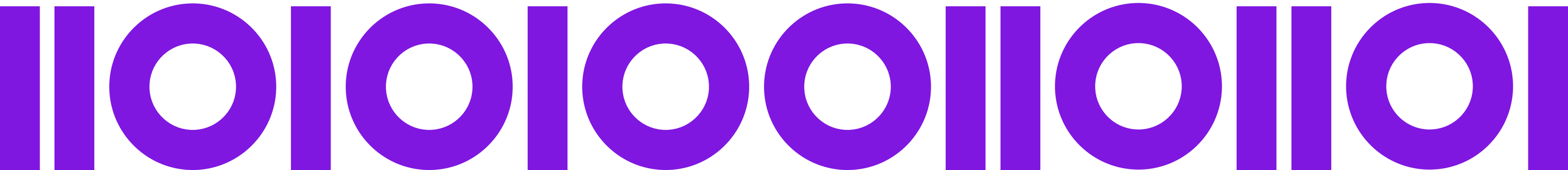
Encryption keys are often used frequently when batch operations are performed on sensitive data. It is not unusual that a batch program would need to perform millions or tens of millions of encryption and decryption operations. While the retrieval of an encryption key from the key server may be very efficient, performance may suffer when keys need to be retrieved many times. This can be addressed through encryption key caching in the local environment.

Secure key caching should be performed in separate program modules, such as a service program, and should not be cached in user programs where they are more subject to discovery and loss. Any module caching an encryption key should have debugging options disabled and visibility removed. Secure key caching is critical for system performance and care should be taken to protect storage.

Encryption Key Rotation

Periodically changing the encryption keys (sometimes called “key rotation” or “key rollover”) is important to the overall security of your protected data. Both data encryption keys (DEK) and key encryption keys (KEK) should be changed at appropriate intervals. The appropriate interval for changing keys depends on a number of variables including the amount of data the key protects and the sensitivity of that data, as well as other factors. This interval is called the cryptoperiod of the key and is defined by NIST in Special Publication 800-57 “Key Management Best Practices”. For most IBM i customers rotation of data encryption keys should occur once a year, and rotation of the key encryption keys should occur no less than once every two years.

Periodically changing the encryption keys is important to the overall security of your protected data. The appropriate interval for changing keys depends on a number of variables including the amount of data the key protects and the sensitivity of that data.



Performance

The performance of an encryption solution is one of the biggest concerns that an IBM i customer has when implementing FieldProc. There are many factors that can affect performance of a FieldProc application and it is wise to pay special attention to performance as you prepare to implement a solution. Let's look at several factors that can affect performance.

AES 128-bit and AES 256-bit Performance

All key sizes for AES encryption (128-bit, 192-bit, and 256-bit) are considered secure for protecting all commercial sensitive data. Customers naturally wonder if the smaller key size of 128-bit encryption means better encryption performance when compared to 256-bit AES keys. In fact, the performance difference is much smaller than one might imagine. The smaller 128-bit key size uses 10 rounds during AES encryption, but the 256-bit AES keys use 14 rounds during AES encryption. The IBM i RISC processor optimizes cryptographic operations so the performance penalty for 256-bit AES encryption is very small. Considering that cybersecurity guidelines now recommend the use of 256-bit AES for protecting data in long-term storage, and for protecting against advances in quantum computing, you should always consider using 256-bit AES encryption for protecting sensitive IBM i data.

AES Encryption Software Performance

Encryption software libraries can vary greatly in performance even when using exactly the same methods and key sizes. Unfortunately, the native IBM i AES encryption software library and APIs have a low performance profile and this can have a negative impact on your IBM i applications. Fortunately, there are high performance AES encryption libraries available from third parties that have a much better performance profile. The difference between the native IBM encryption libraries and third party libraries can be more than 100 times in processing speed. Use care when deploying an encryption solution to ensure that the performance meets your minimum needs for processing time.

Security Based on Standards

Precisely's high-performance encryption solutions for IBM i, including the FieldProc implementation, use NIST-validated 256-bit, optimized AES encryption. This matches current security recommendations for long term archival of sensitive data.

It is important that the FieldProc application be properly optimized as it may be invoked many times during a typical interactive or batch request.

FieldProc for Multiple Columns in a Table

It is likely that you will need to protect multiple columns in a table. For example, in a medical setting a table might contain the following information for a patient:

- Name
- Date of birth
- Address
- Social security number
- Email address
- Etc.

In this case you will be protecting multiple columns in a single table. This is fully supported by FieldProc and is very common in FieldProc implementations. When FieldProc programs are properly optimized for performance, you should find that the extra performance impacts per column are not excessive. Thanks to the re-entrant model of the IBM i operating system, the protection of multiple columns has a smaller performance impact. Encrypting two columns is NOT twice the performance impact as encryption one column.

HINT:
When protecting multiple columns in a table use the same encryption key for each column if this is acceptable to your security policy. Limiting the number of encryption keys used in a table will provide a performance benefit.

Key Management Performance Impacts

As mentioned above, your FieldProc encryption solution should implement good encryption key management practices. It is important that the key management interface impose little performance penalty. This means that the FieldProc application should use intelligent and secure key caching to minimize the number of key retrieval operations that must be performed. Additionally keys should not be exposed in user applications, but should be protected in separate modules. When implemented properly, good key management will impose extremely little performance impact.

Audit Logging Performance

One common feature of FieldProc applications is the ability to collect audit information about user activity. This might include collecting information about which users accessed decrypted information, etc. Audit logging will always exact a performance price. If you need to collect audit logs of FieldProc activity, be sure to measure the performance impact of audit log collection in your own environment.

Power8 On-Chip AES Encryption Performance

The Power8 processor from IBM provides a hardware implementation of AES encryption to improve the performance of encryption. All models of IBM i servers built with Power8 (or greater) processors include this hardware implementation of AES encryption. In concept this is similar to the Intel processors which provide AES encryption through the AES-NI implementation.

While encryption performance improved with the Power8 processor, it is not a large improvement. Encryption speeds using the native IBM i APIs are about double the speed of the previous versions of the Power processor, but still greatly lag in performance compared to third party encryption libraries. It should be noted that this performance lag disappears when the IBM APIs are used to encrypt very large blocks of information. For example, the AES implementation on the Power8 processor can encrypt a megabyte of information at incredible speeds. Unfortunately, this does not apply to the smaller blocks of data (credit card numbers, social security numbers, etc.) that are typically stored in Db2 database tables. It is likely that IBM i customers using IASP encryption will see a major performance benefit with the new Power8 systems.

FieldProc Program Performance

When FieldProc invokes a program to perform encryption or decryption, it makes a dynamic call to a program executable. There is always a certain amount of overhead when making a dynamic call to an external program, and this is true in FieldProc context, too. The more columns in a table that are under FieldProc control, the more dynamic calls you will have to the FieldProc program, as the FieldProc program is invoked independently for each column. It is important that the FieldProc program be properly optimized for performance. Some key performance measures include:

- File I/O operations by the FieldProc program are minimized.
- Program modules are optimized on compilation.
- The program executable is optimized on compilation.
- Memory management is optimized for performance.
- Visibility of program objects is removed to improve performance.
- The FieldProc program is optimized for multi-threaded operation.
- Audit logging is optimized for performance.

Limitations

Encrypted Indexes

The Db2 FieldProc implementation fully supports the encryption of columns which are indexes (keys) to the data in a native SQL context, and this includes RPGSQL applications. However, there are some severe limitations with legacy RPG and COBOL applications around encrypted index order. It is important to understand these limitations if you are approaching FieldProc with a large inventory of legacy applications.

Legacy RPG applications use record-oriented operations and not set-oriented operations that are typical of SQL. Many record-oriented operations in RPG will work as expected. For example, a CHAIN operation on an encrypted index to retrieve a record from a Db2 table will work. If the record exists, it will be retrieved, and FieldProc will decrypt the value. However, many range and data ordering operations will not work as expected with legacy RPG programs. Consider the following logic:

- SETLL (position to an particular location in the index for a fiel)
- WHILE (some condition)
- READ (Read the next record by the index)
- END WHILE

The SETLL (Set lower limit) operation will probably work if the particular index value exists. However, the program logic will then read the next records based on that position in the file. IBM i developers are surprised to learn that they will be reading the next record based on the ENCRYPTED value, and not on the decrypted value which is what they might expect. The result is often empty subfiles and printed reports. This is very common logic in applications where indexes are encrypted. Note that your RPG program will not get a file I/O error, it just won't produce the results you expect.

In simpler applications this side-effect of encrypted indexes is not significant or can easily be programmed around. However, in some applications where sensitive data is encrypted across a large number of tables (think social security number in banking applications) this can be a significant limitation. The solution to this issue is discussed in the following section.

Overcoming Encrypted Index Limitations in RPG

The limitations of encrypted indexes in legacy RPG applications often lead IBM i customers to abandon their encryption projects. The prospect of converting a large number of legacy RPG applications to newer SQL interfaces can be daunting. Their legacy RPG applications contain a lot of valuable business logic, and the effort to make the conversion can be quite large.

Wouldn't it be great if you could wave a magic wand and make legacy RPG applications use SQL without any changes? IBM opened a path to this type of solution with Open Access for RPG, or OAR. OAR allows for the substitution of traditional file I/O operations with user-written "Handlers". In essence, you can replace the native file I/O operations of RPG with your own code. No change to program file handling or business logic! The OAR capability spawned a number of user interface modernization products, and other solutions that take advantage of this. Imagine if your RPG screen handling I/O with Execute Format (EXFMT) could be replaced with a web-based GUI library. Instant modernization of the UI! There are now many solutions that leverage OAR for UI modernization.

Join Logical Files With DDS

One limitation of logical files created with DDS specifications involves join logical files. You will not be able to create DDS join logical files where the join involves an encrypted field with FieldProc. You will get an error about invalid data type usage. This is an IBM limitation, and there is no known workaround for this issue. Note that this limitation only applies to DDS join logical files and does not apply to SQL joins using encrypted indexes. Most IBM i customers will need to change their RPG or COBOL program logic to avoid the use of DDS join logical files which use encrypted indexes.

Precisely offers an OAR solution that maps all of the traditional RPG I/O primitives (CHAIN, READ, READE, SETLL, etc.) to equivalent SQL operations. With one line of code that defines an OAR handler for a file, the program I/O is changed to a native SQL operation, or set of operations, that mimic the original RPG primitive. By using the native SQL Query Engine (SQE) in place of legacy RPG I/O operations, the problem with encrypted indexes is eliminated.

Application Changes

Legacy RPG applications that use encrypted indexes often need re-design and re-programming to avoid the problems of encrypted indexes. You can avoid these issues if you are using an Open Access for RPG (OAR) solution that maps the legacy RPG record-based file operations to native SQL and the SQL Query Engine (See the note above about Precisely's OAR/SQL solution).

If you need to re-design your RPG applications to avoid encrypted indexes, consider putting all of your sensitive data in a table that is indexed by some non-sensitive value such as a sequence number or customer number, and use FieldProc to encrypt that table. There are many approaches to application redesign, and you should be able to find a method that works for you.

Change Management

IBM i customers who have deployed change management solutions can encounter challenges with FieldProc implementations. While most of these challenges are surmountable, it will take effort on the part of the systems management team to integrate FieldProc controls into their change management strategy. Unfortunately, the implementation of FieldProc does not provide much in the way of support for change management systems. The activation of FieldProc changes an attribute on the column in the table, but change management systems generally are not aware of this attribute. It can be challenging to promote table and column level changes and properly retain the FieldProc attribute of the data. Look for a FieldProc implementation that provides a command-level interface to stop, start, and change FieldProc definitions. These commands can help you integrate FieldProc encryption into your change management strategy.

Additional Security Controls

User Access Controls

The implementation of a FieldProc encryption solution also opens the door for additional security controls. One of the important security controls is user access control to decrypted data. You will want to prevent unauthorized users from viewing decrypted data and your FieldProc application should provide for this ability.

Good security practice includes establishing proper object-level security for your applications and databases. FieldProc-level access controls complement object-level security and provide additional levels of control and audit. Good security practice is to use a whitelist approach for these access controls and data masking. This gives you the ability to control access to sensitive information for highly authorized users such as QSECOFR or any user with All Object (*ALLOBJ) authority.

Data Masking

Another important security feature that should be implemented in your FieldProc solution is data masking. Compliance regulations such as the PCI Data Security Standard (PCI-DSS) require appropriate data masking, and this can only effectively be applied as a part of the FieldProc decryption process. In the same way that user access controls should be based on a whitelist approach, data masking should be based on a similar approach. Look for a data masking implementation that allows you to specify a default masking rule – you will want to fully mask data by default and only allow selected users to see partially masked or unmasked data.

IBM and Third Party Utilities and Backup

Backup and Archival Encryption

One advantage of using FieldProc is that your backup will also be encrypted when moved to off-site storage. With the advent of CryptoLocker and similar malware, the need for independent, non-attached backup and archival has increased. When you use any of the IBM SAV commands such as SAVLIB, SAVOBJ, and so forth, the encrypted status of your FieldProc protected data is preserved.

If you are using a third party solution for backup, be sure that the vendor is using the native IBM save command in their solution. The use of FTP, ODBC and other methods for backup will almost certainly result in unencrypted data in the backup.

Precisely offers solutions for both access control and multi-factor authentication. Implementing these disciplines can provide additional protection for Fieldproc configurations.

FieldProc and FTP, ODBC, and Other File Utilities

Many IBM and third party file utilities are used to access Db2 data. Since FieldProc is implemented at the Db2 database layer, when these utilities access information that is protected by FieldProc, the database will invoke FieldProc to decrypt the data on a read operation. This means you must use caution when granting access to Db2 data by utilities such as FTP, ODBC and others. Be sure you have user access controls in place and be sure to implement data masking in your FieldProc implementation. Unfortunately, the FieldProc application does not have visibility to which application is making the data access request and cannot apply applicationlevel controls.

Be sure to consider adding additional data protection controls to your IBM i server. This should include exit point security for FTP, ODBC and other utilities. You should also consider implementing multi-factor authentication for administrator access to your IBM i server.

Assure Encryption

Assure Encryption is trusted by organizations worldwide to protect their private data. The solution helps organizations meet the data security compliance requirements of PCI DSS, HIPAA, and state privacy notification laws. Assure Encryption protects sensitive information in database fields, backup tapes, Save Files, IFS files, reports, and everywhere your data resides on the IBM i. The solution is optimized for performance and the encryption APIs perform up to 100x faster than IBM's encryption APIs.

Assure Encryption is the only NIST-validated AES database encryption solution for IBM i. With an intuitive and familiar IBM i interface, administrators can easily configure and manage encrypted files and reduce security exposures by implementing controls about who can view decrypted data. Assure Encryption is in use by Fortune 500 companies worldwide.

“Staples wouldn’t even consider a solution that didn’t go through a NIST validation. Assure Encryption was a natural choice.”

Staples

precisely

About Precisely

Precisely is the global leader in data integrity, providing accuracy and consistency in data for 12,000 customers in more than 100 countries, including 90 percent of the Fortune 100. Precisely's data integration, data quality, location intelligence, and data enrichment products power better business decisions to create better outcomes. Learn more at www.precisely.com.

www.precisely.com