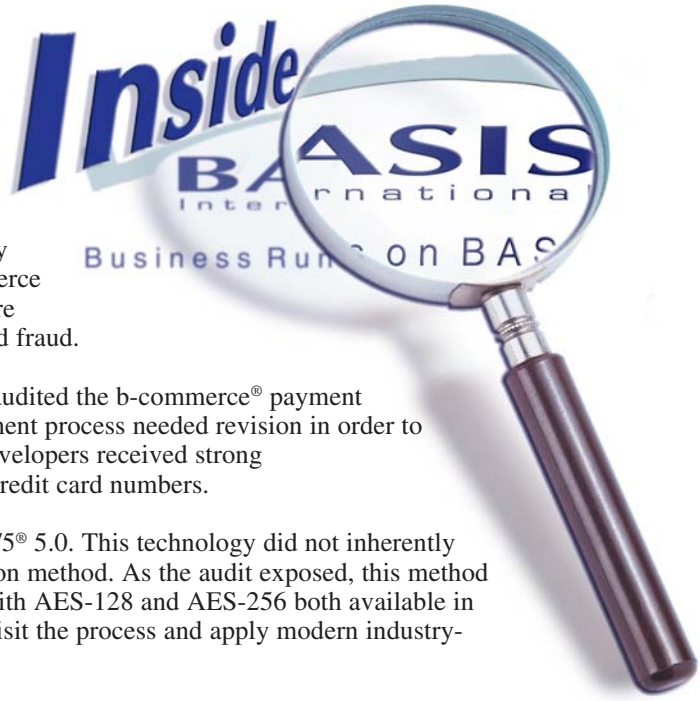
BASIS Puts Triggers to Work

By Tom Hines



A common occurrence in so many industries is the need to comply with updated regulations and improved standards. The e-commerce world is no exception. Today's industry standards require a more effective defense against the rise in identify theft and credit card fraud.

The vendor that manages the BASIS electronic credit card transactions audited the b-commerce® payment process earlier this year. This timely audit revealed that the BASIS payment process needed revision in order to comply with today's heightened security. Recently, many other BBx® developers received strong recommendations to tighten up security on the storage and handling of credit card numbers.

BASIS built their original solution on the BBx technology prior to PRO/5® 5.0. This technology did not inherently provide encryption capabilities, so BASIS developed their own encryption method. As the audit exposed, this method no longer met the newer, more stringent industry standards. However, with AES-128 and AES-256 both available in version 6.x of the PRO/5 family and in BBj® 6.0, BASIS could easily revisit the process and apply modern industry-standard encryption.

Considering all the tools available, we chose to run the ENCRYPT() and DECRYPT() functions automatically in the background using the triggers introduced in BBj 6.0. Before we could implement this program, we had to prepare the database.

Expand the Field Length

Depending on the encryption algorithm used, encrypted data may require more space than the decrypted version of the data. For example, encrypting an 18-byte string using AES-128 bit encryption will result in a 32-byte string, which can affect field definitions in files and string templates. Since we originally defined our credit card data file as a 16-character field, executing the following code showed that we needed 32 characters:

```
>encrypted_card$ = encrypt(fill(16,"0"),MODE="CRYPTPASS=SamplePassword")
>?len(encrypted_card$)
32
```

Since there was insufficient space in our file to store 32-character numbers, we created a new data file with a larger credit card number field into which we would copy the old data after we defined our triggers.

Tom Hines
MIS Programmer

Write New ENCRYPT() DECRYPT() Programs

We wrote two short programs to be used as triggers; one to encrypt on the "write" and one to decrypt on the "read."

continued...

Program to Decrypt

```
rem 'Program name: cc_decrypt.src
rem 'Trigger program to decrypt cc numbers after a read

trigger!=BBjAPI().getFileSystem().getTriggerData()
dim ccrec$:"cust_key:c(10),cc_num:c(32),cc_type:c(1),cc_name:c(30),cc_expire_mo:c(2),cc_expire_yr:c(4),misc:c(21)"
ccrec$ = trigger!.getReadBuffer()
encr$=cvs(ccrec.cc_num$,3)
ccrec.cc_num$=decrypt(encr$,mode="cryptpass=our_password")
trigger!.setReadBuffer(ccrec$)
```

Program to Encrypt

```
rem 'Program name: cc_encrypt.src
rem 'Trigger program to encrypt cc numbers before a write

trigger!=BBjAPI().getFileSystem().getTriggerData()
dim ccrec$:"cust_key:c(10),cc_num:c(32),cc_type:c(1),cc_name:c(30),cc_expire_mo:c(2),cc_expire_yr:c(4),misc:c(21)"
ccrec$ = trigger!.getWriteBuffer()
encr$=cvs(ccrec.cc_num$,3)
ccrec.cc_num$=encrypt(encr$,mode="cryptpass=our_password")
trigger!.setWriteBuffer(ccrec$)
```

Set up Triggers

To run these new programs automatically, we followed these simple instructions:

1. Start Enterprise Manager and connect to the server.

2. Right click on 'Triggers' to show the option list.

3. Choose 'Mount Directory of Files' from the list to display a screen for navigating to the data directory, then mount the data directory under the Triggers folder.

4. Right click the mounted directory to display the options.

5. Select 'Add Trigger' to display the data files in the mounted directory.

6. Choose the credit card data file to add the file name below the mounted directory.

7. Click on the file name and view the trigger information.

8. Select the desired trigger type and enter the definition.

   a. Set and define first trigger.

   We chose 'Before Write' and 'After Read' types. (Before Write defines the action to occur immediately before the application writes a record to the file, and After Read defines the action to occur after reading the record from the disk and before the application receives it.) Selecting 'Before Write,' we entered the path and program name for encrypting **`/[path to program]/cc_encrypt.src`** *into the File Name field.*

   i. Press [Save Changes] to display the program in the code area.

   ii. Check the 'Enabled' checkbox and [Save Changes] to complete this trigger as shown in **Figure 1**.

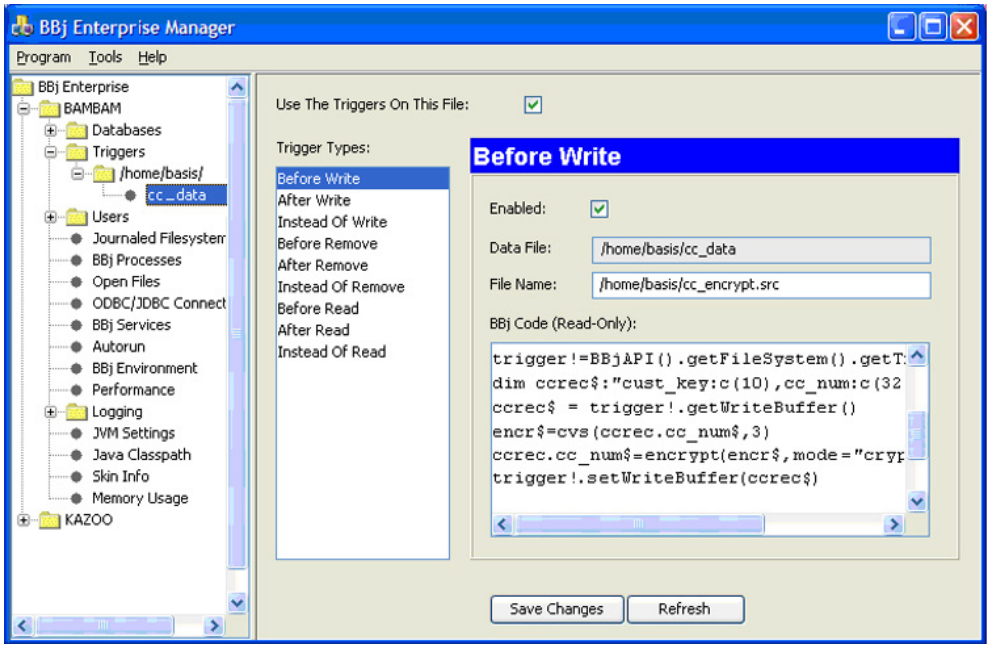
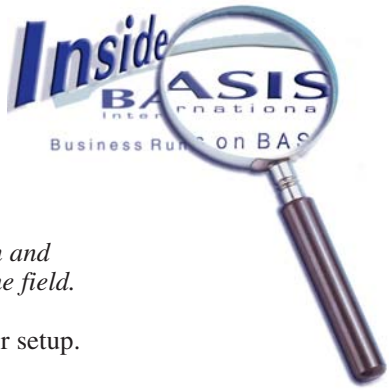


Figure 1. The trigger set up dialog box

continued...



b. Set and define second trigger.

Repeat the process listed in a. *but selecting the 'After Read' trigger type, entering the path and decryption program name* **`/[path to program]/cc_decrypt.src`** *into the File Name field.*

9. Check the 'Use The Triggers On This File' box to save the changes and complete the trigger setup.

SAVEP Program

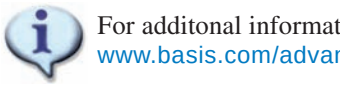
Using SAVEP, we saved the trigger programs to a protected status.

With the triggers in place, we copied all of the data from the old file into the new file and the triggers automatically encrypted the credit card numbers before writing them into the new file. Now all additional interaction with the file containing the new encrypted credit cards will happen transparently for any program or ODBC/JDBC driver reading or writing the file, as the triggers are responsible for automatically encrypting and decrypting the credit card data "on-the-fly."

Summary

Apart from the trivial task of expanding the length of the credit card field in the data file, our applications required no changes whatsoever to implement this new encryption standard. There was no disruption to existing code, ensuring continuity of the world-wide BASIS operations. Perhaps this real-world example might encourage you to review your own encryption needs and utilize these great new features offered in BBj 6.0.

Looking beyond encryption, there is a multitude of ways triggers can enhance your applications and add value to the end users. Consider the far-reaching impact of changes made at the database level rather than in the application. Triggers can deliver unseen benefits limited by only your imagination. Try it - you may become trigger happy!



For additonal information, read *Using Stored Procedures to Add Business Logic to the Database* at www.basis.com/advantage/mag-v10n1/sprocs.zip