Safe and Retrievable Intellectual Property

By Adam Hawthorne

Security is a topic that receives plenty of attention these days. (Typically, we think of security in terms of protecting our data and communications; however, there is another very critical need for security. Today's SAVEP verb provides developers the ability to save their programs so that no one, including the original author, can subsequently list the programs to obtain the original source code. While this effectively protects their intellectual property, many developers avoid using it because they want unrestricted access to their code to perform software maintenance and enhancement updates conveniently. Also, to ensure future access to the unencrypted version, developers must take extra care to avoid accidentally overwriting the original unencrypted version with the SAVEP'd version.

In BBj® 6.0, BASIS enhances the SAVEP verb and introduces the CLEARP verb. Developers can retain their original code, maintain and enhance it without restrictions, and still prevent others from obtaining access to their intellectual property. This article introduces the syntax, behavior, and examples of these exciting new features.

Syntax

The following list specifies the ways the BBj developer can use the SAVEP verb:

```
SAVEP "filename"
SAVEP "filename", 0
SAVEP "filename", $$
SAVEP "filename", $$, 0
SAVEP "filename", "passwd"
SAVEP "filename", "passwd", 0
```

The first two statements listed above follow the existing syntax, and the last four statements use the additions to the syntax. The obvious addition is the password parameter. This new parameter causes BBj to save some state in the program that allows developers to recover the listing from a program previously saved with the SAVEP verb.

Use CLEARP, the twin of the new syntax for SAVEP, according to the following syntax:

```
CLEARP "passwd"
```

CLEARP allows authors to restore the program to a listable state. A program saved with a password and the SAVEP verb, will list again after successfully executing the CLEARP verb.

Behavior

Think of the new SAVEP as a way to set the protected status of the program. Previously there was no analogue to SAVEP, but with the new password parameter and the CLEARP verb, developers have the means to clear the protected status of a program, while retaining a high degree of security.

For discussion purposes, ignoring the last integer parameter on the SAVEP verb allows the collapsing of the six forms of the SAVEP syntax into these three forms:
1) the syntax with an empty string for a password,
2) the syntax with a non-empty password, and
3) the syntax with no password.
The existence of the integer parameter retains its current meaning, instructing BBx® to create a new file.

The first form of SAVEP takes an empty string (**`$$`**) and sets the protected status of the program, but does not provide a password for CLEARP. This is the explicit form of the existing SAVEP syntax and works the same way as it did in previous versions of the language. Users who desire to mix unrecoverable programs saved using the existing SAVEP implementation with programs using the new password implementation may consider using this option in the future for the sake of consistency.

The second form of SAVEP takes a password parameter and sets the protected status of the program, giving BBj developers the option to clear the protected status of the program with the CLEARP verb. When developers use this syntax and specify a password **`foo`**, they can clear the protected status of the program so that it is listable with the following statement:

```
CLEARP "foo"
```

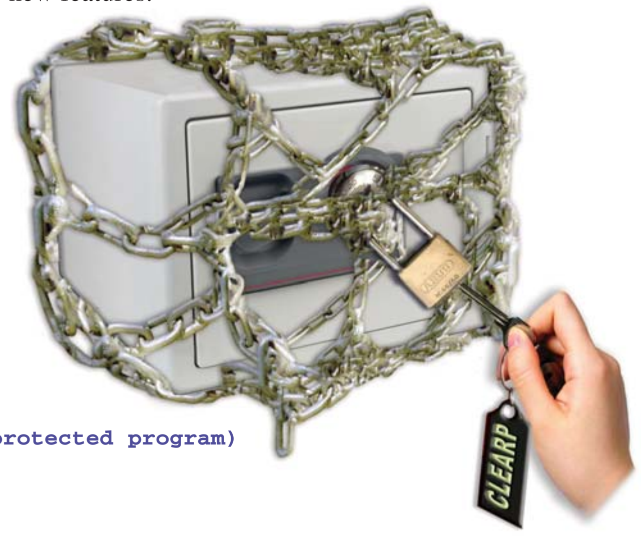
The third form of SAVEP does not have a password. This form is enhanced to protect the developer from making casual mistakes when working with password protected programs. SAVEP, with no password, now saves the program the same way as before it was loaded. This means that SAVEP in this form will continue to work as it always has for programs that were not previously saved with a password. However, if a developer executes a SAVEP without a password on a program that was previously saved with a password, the original password is maintained and the original source code will remain recoverable via the CLEARP verb. This maintains language compatibility with previous versions of PRO/5® and BBj.

BASIS also added some powerful features along with the syntax to maintain the security of existing SAVEP programs and programs that will use the new password feature. The first feature prevents users with access to the console from using SAVEP to recover the listing with their own password. This means a developer cannot save a program using the SAVEP verb with one password and then SAVEP it again using a different password. Another important feature is that programs previously saved with SAVEP continue to require the same amount of care as before, preventing malicious access to listings of programs that were previously unobtainable. While this also prevents programmers from adding a password to an

Adam Hawthorne
Software Engineer

continued...

existing protected program, it allows them to protect their original programs, if they have the original source, with a password to make use of these new features.



Examples

```
REM ' SAVEP with an empty password.
SAVEP "program.bbj", 0
SAVEP "program.bbj", $$, 0
SAVEP "program.bbj"

REM ' SAVEP with the password "password"
SAVEP "program.bbj", "password", 0
SAVEP "program.bbj", "password"
SAVEP

REM ' Causes an !ERROR 18, different password.
> SAVEP "program.bbj", "helloworld", 0
> SAVEP "program.bbj", "goodbye", 0
!ERR=18 (Different password specified on SAVEP of protected program)

REM ' Cannot change the unclearable password
> SAVEP "program.bbj", 0
> SAVEP "program.bbj", "goodbye", 0
!ERR=18 (Different password specified on SAVEP of protected program)
```

Summary

With these new enhancements, BASIS makes it possible for authors to protect their most important attribute – their intellectual property – and manage their updates. Security is more than the buzzword of the decade. It is a reality and one that has a worthwhile and beneficial place in your application.