Explore Feature Restriction Strategies for Trial Solutions

Learn how to determine and implement the most effective protection for your trial FileMaker solution.

By Tony Tanevski, Hi-Voltage Corporations president (Australia)

Have you ever wanted to protect your commercial FileMaker Pro solution from trial users by limiting its usefulness with some form of feature restriction?

Choosing the right restriction strategy for your solution is imperative. The methods you use can lead to sales or missed opportunities. An effective strategy gives users the freedom to thoroughly evaluate the solution while still limiting its usefulness, thus giving them a reason to purchase the product.

In this article, I discuss a number of popular restriction methods you can employ to lock down your trial solutions. I also point out the advantages and disadvantages of using each method.

Examples

I have provided example files for each restriction method discussed in the article.

Before you start

Each of the restriction methods I discuss in this article requires a number field called TrialMode. Make sure you temporarily place the TrialMode field on a layout and insert "1" to indicate the solution is in trial mode. A blank value indicates the solution is running in regular mode. You could write a simple script to turn trial mode on and off. Make sure you remember to turn on trial mode before deploying your solution.

30-day trial period

The most popular restriction method is the 30-day trial method. I'm sure you've seen this method in use. The intention is to get users hooked into using your solution for 30 days, and hopefully influence them to purchase a license before it expires.

The 30-day trial method works best for smaller solutions that don't require a great deal of evaluation. Larger solutions take longer to evaluate; therefore restricting your potential customer to 30 days may not be the best option. You can allow longer trial periods -- for example, 45 days -- if you feel 30 days is insufficient.

Another thing to consider is your solutions' usage frequency. If your solution will be used regularly, 30 days may be sufficient time for an evaluation. On the other hand, if your solution will be used less frequently, your potential customers may not have enough time to thoroughly get a feel for your solution.

Although users can backdate their system clock to prevent the solution from expiring, this rarely becomes an issue because it's a chore to keep remembering to alternate dates -- especially if they're doing this with other products as well.

I'm not a fan of the 30-day trial method simply because I usually don't get around to thoroughly evaluating the software before it expires. It becomes an annoyance when I'm ready to try out the software, only to find it has expired. This isn't to say you should never employ 30-day trial periods. If you choose the right arrangement for your solution, the method can work well. (Refer to the example file, shown in figure 1, to see the method in action.)

Figure 1: 30-day trail method -- The user can evaluate the solution for 30 days before it expires.

The procedure

To set up a 30-day trial, first you have to create the following fields in a system table:

TrialMode	Number
StartDate	Date
Expired	Number

Make sure the TrialMode field contains "1" when testing the method. Next create the following script:

```
If [ TrialMode ]
      If [ isempty(StartDate) ]
            Set Field [ "StartDate"; "Get(CurrentDate)" ]
      Else
            If [ Get(CurrentDate) >= StartDate + 30 or
                  Get(CurrentDate) < StartDate or Expired ]
                  Show Custom Message [ "Your 30 day trial period
                  has expired."; "OK" ]
                  Set Field [ "Expired"; "1" ]
                  Exit Application
            End If
      End If
End If
```

You have to call this new script early in your start-up script. (You should create a start-up script if you don't already have one).

How it works

If the solution is in trial mode at start-up, the following script first checks if StartDate contains any data. StartDate determines when the user first ran the solution. If the field is blank, the user hasn't run the solution, so the script stores the current date.

Get(CurrentDate) >= StartDate + 30 checks if the current date is 30 days past the initial start date.

Get(CurrentDate) < StartDate checks if the current date is less than the initial start date.

This is a precautionary measure to prevent users from setting their system clock 10 years ahead, which would give them 10 years of unrestricted use if they first run the solution with the future date.

To prevent users from backdating their system and gaining illegal access, you set the Expired field to "1" to permanently expire the solution (figure 2).

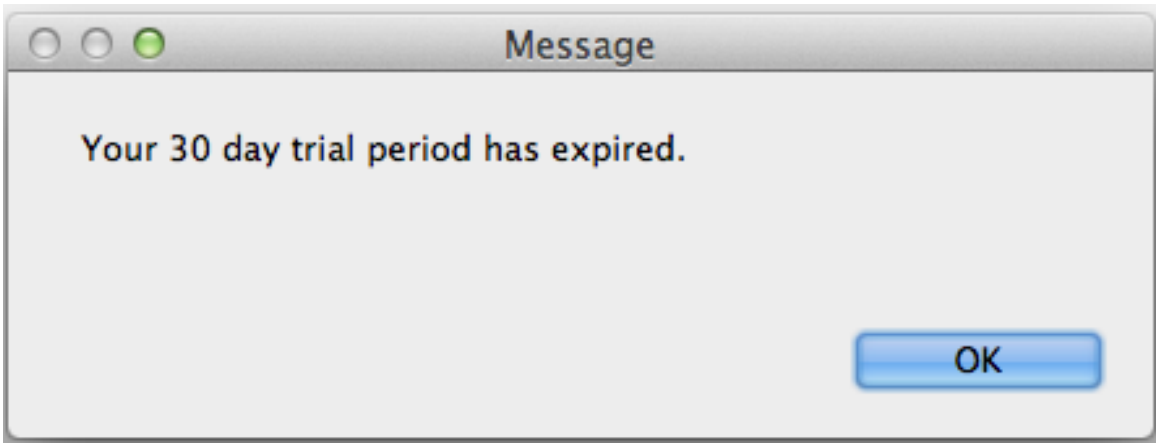
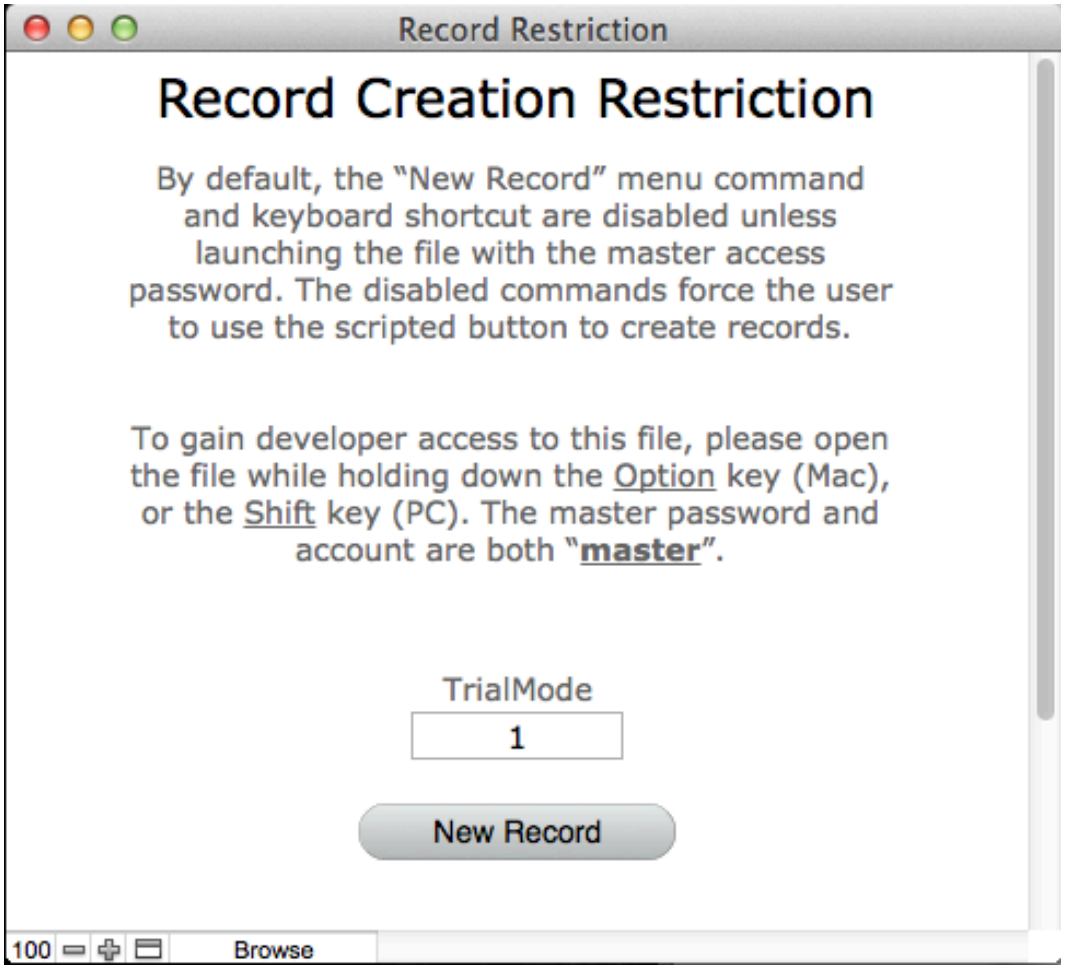


Figure 2: The solution expires -- A dialog informs the user as soon as the solution expires.

Record creation restriction

This restriction method lets you limit the number of records a user can create in your solution. The intention is to give users enough space to try the system while imposing a limit on its usefulness. For example, if your solution is a contact management system, you could limit the number of contacts the user can create.



When deciding the record limit to impose, you should consider whom you're targeting. If your solution targets larger organizations, you may decide to allow 10 or 20 records. If your solution targets smaller organizations, 5 to 10 records may be an adequate restriction. The key is to give users sufficient room to thoroughly evaluate the solution, but not too much, so they have a reason to purchase your product.

There are a few issues to consider when restricting the record creation process. Users can create records via keyboard shortcuts or from the menu, so you have to disable this ability and only let users create records via scripts. By disabling record creation commands, you also disable other commands such as Find and Preview. Therefore, you may have to provide these commands via buttons on your layouts.

The procedure

First create the TrialMode field in a system table (or file), making sure the field contains "1" when testing the method. Next, you have to edit the account privileges and set the available menu commands to "Editing only" (figure 3).

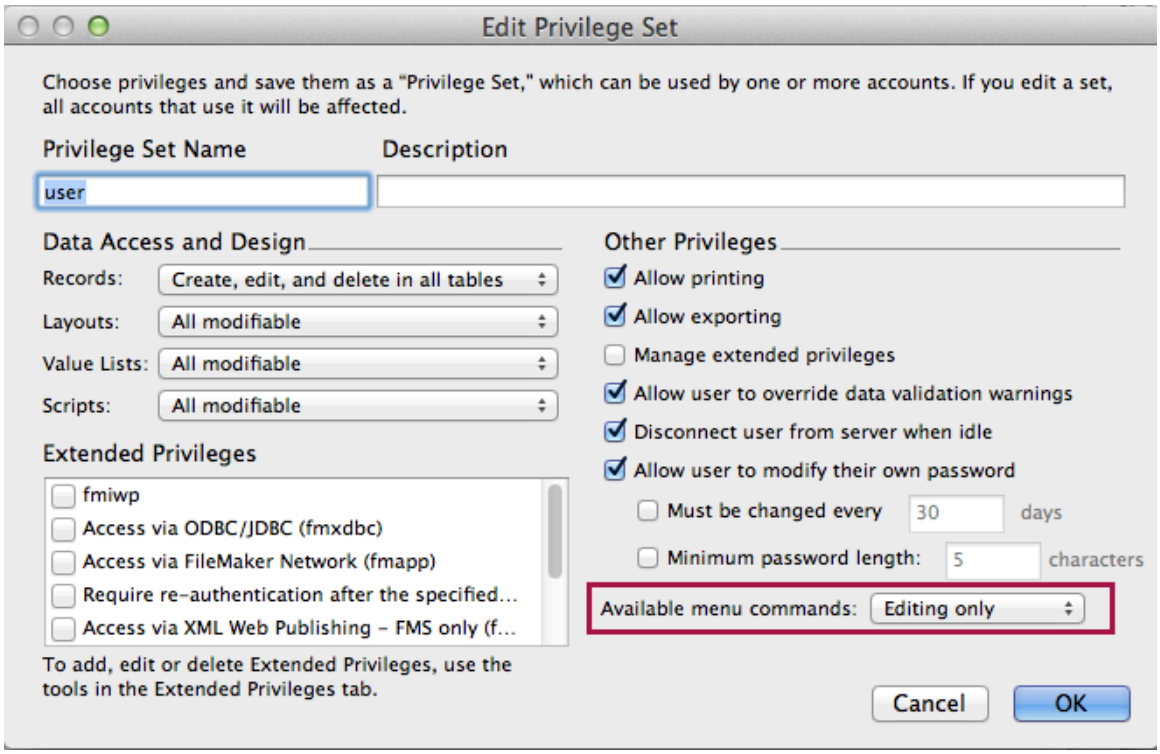


Figure 3: Setting the Privilege Set's available menu commands -- You won't be able to apply this to your master privilege set, so you have to create a different set if you don't already have one.

Next create this script and call it from your layouts via a button:

```
If [ TrialMode ]
      If [ Get(RecordCount) >= 5 ]
            Show Custom Message ["The maximum number of records
            has been reached."; "OK" ]
      Else
            New Record / Request
      End If
Else
      New Record / Request
End if
```

How it works

If your solution is running in trial mode, the script checks whether the record count has reached the limit of 5 records. If so, a dialog displays informing the user about the restriction (figure 4). If the record limit hasn't been reached, the script creates a new record.

Figure 4: Creating more than 5 records -- A dialog informs the user that the limit has been reached.

Nag screen

Another popular restriction method is to display a nag screen for several seconds before the user can gain access to the solution. The idea behind this is to nag the user enough times with the hope they'll get fed up and decide to register the solution. Nobody likes nag screens -- especially the ones that make you wait for 10 seconds or more. Long delays can deter users, so it's best to keep the nag screen short. Around 5 seconds is usually sufficient.

Nag screens alone may not be enough to encourage users to purchase your solution. It's best to combine nag screens with other restriction methods.

The procedure

First create the TrialMode field in a system table (or file), making sure the field contains "1" when testing the method. Next you have to create a script called "Nag Button - Close" with this line of code:

```
Exit Application
```

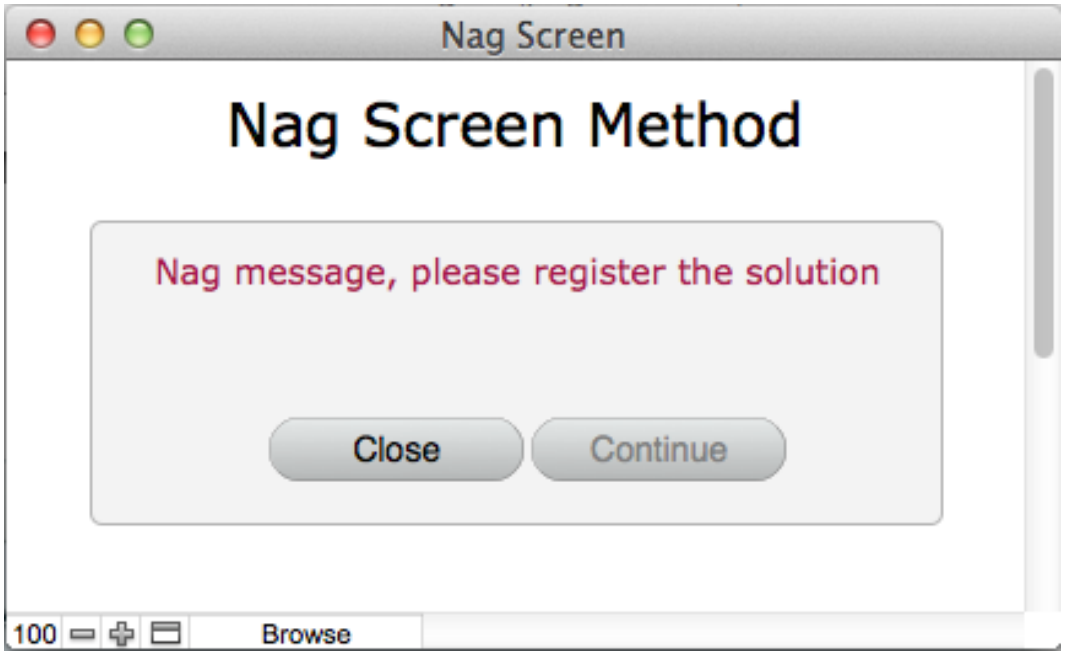
Now create another script called "Nag Button - Continue" with this code:

```
Go To Layout [ "Main" ]
```

You have to set the Go To Layout script step to point to your main layout.

Next create two new layouts. The first layout "Nag-Initial" displays while the user waits for the time delay to elapse, and the second layout "Nag-Secondary" displays after the time delay.

Place two buttons side-by-side on the Nag-Initial layout (figure 5). Label the first button Close and the other Continue. Set the Close button to call the "Nag Button – Close" script. Leave the "Continue" button without an assigned script, and set the label text color to grey. This button won't perform any functions; it will appear to the user as disabled while he waits for the delay to elapse.



- Figure 5: Nag-Initial layout -- The Continue button is disabled while the user waits for the time delay to elapse.

On the Nag-Secondary layout (figure 6), place another two buttons in the exact same position as the ones on the previous layout and label them the same. This time set the Continue button to call the "Nag Button – Continue" script and set the button text color to black to indicate it's now enabled. Again, set the Close button to call the "Nag Button – Close" script.

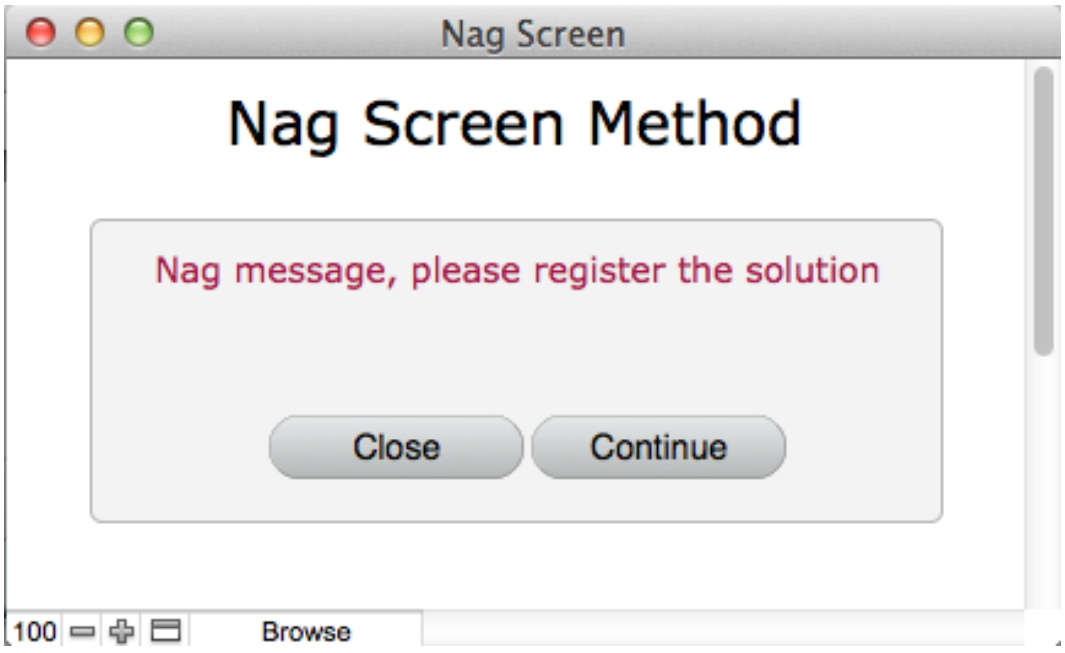


Figure 6: Nag-Secondary layout -- After the time delay, the script enables the Continue button, letting the user continue evaluating the solution.

Next, create a new script:

```
Allow User Abort (off)
If [ TrailVersion ]
      Go To Layout [ "Nag-Initial" ] Pause [ 5 seconds ]
      Go To Layout [ "Nag-Secondary" ]
Else
      Go To Layout [ "Main" ]
End if
```

Call this new script early in your start-up script. Of course, if you don't already have a start-up script, you have to create one.

To create a start-up script, refer to the steps discussed earlier within the 30-day-trial method.

How it works

If the solution is running in trial mode at start-up, the Nag-Initial layout displays for 5 seconds. While the user waits, he can close the solution by clicking on the Close button. However, he can't click on the Continue button because it's disabled.

After the 5-second time delay, the Nag-Secondary layout displays with an enabled Continue button, giving the user the option to continue or close the solution.

Restricting functionality

A great way to restrict your trial solutions is to disable certain functions such as printing, viewing certain layouts, etc. This form of restriction is sometimes referred to as crippleware because the functionality of the trial solution is crippled. This is one of the easiest restrictions you can create, and you can restrict any features you want.

When deciding which features to restrict, be sure to consider the consequences. You don't want to restrict key functionality that will prevent users from evaluating important features. For example, if your solution is a Project Management system, it would be disastrous if you didn't let users create their own projects. A better restriction would be to prevent users from printing invoices or running reports.

The key is to restrict features that are important in the grand scheme of things, but aren't important during the evaluation process (figure 7).

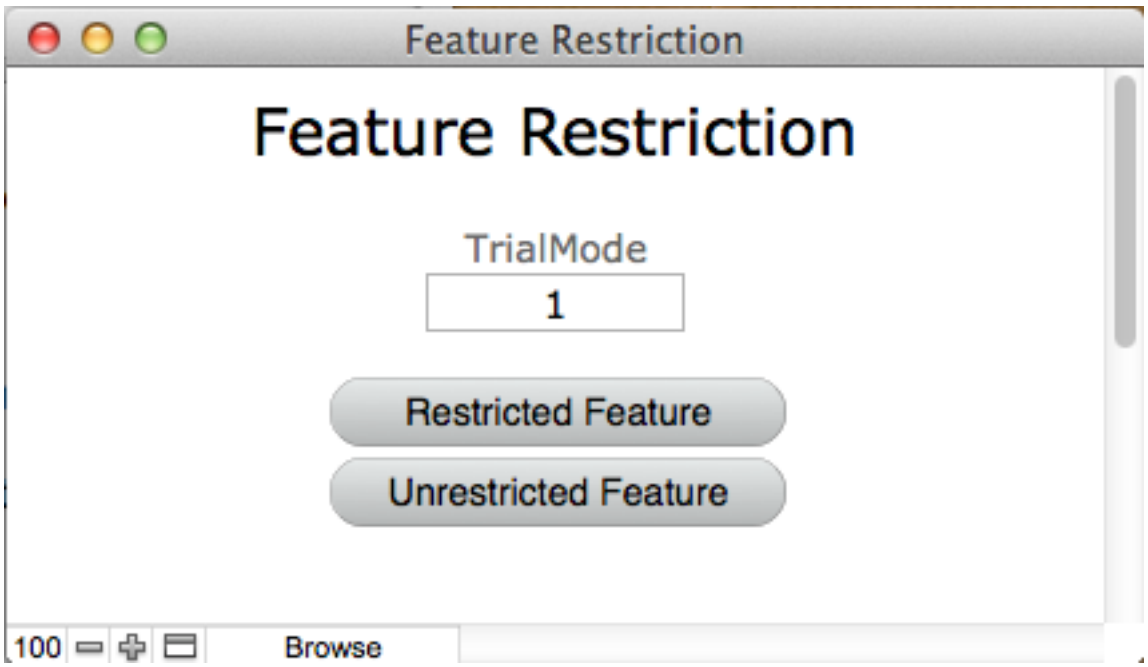


Figure 7: Feature restriction -- The user can't run the restricted script in trial mode but can run the unrestricted one.

The procedure

First create the TrialMode field in a system table (or file), making sure the field contains "1" when testing the method. Next create a new script with this code:

```
If [ TrailVersion ]
     Show Custom Message [ "This function is not available
     in the trial version."; "OK" ]
     Halt Script
End if
```

To restrict certain features, call this new script early in the script you want to restrict.

How it works

If the solution is running in trial mode and a user clicks on a button for a restricted feature, a dialog informs the user that the function can't be performed and the script is halted (figure 8). If the solution is running in regular mode, the script runs normally giving the user full access to the function.

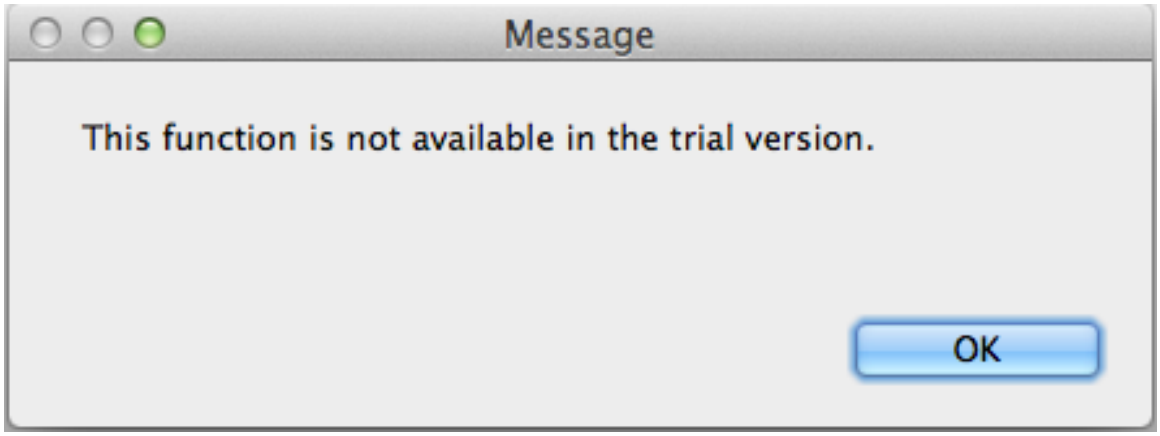


Figure 8: Running a restricted feature -- The user sees a dialog when he tries to run a script that has been restricted.

What's next?

Your final strategy could be a single method or a mix of the methods discussed in this article. You can even invent your own restriction methods to suit your needs.

After you've implemented your restriction strategy, the next step is to test your solution running in trial mode. You may want to make final adjustments before you deploy your trial solution. Then you can sit back and relax knowing your FileMaker Pro commercial solution is well-protected.