

Using the 4D Method Flowchart Editor

By Walt Nelson, Director of Partner Development

4th Dimension Technical Note 97-23

Technical Notes for 97-08-August 1997

Introduction

If you are like most 4D developers, you probably seldom, if ever, use 4th Dimension's Flowchart Editor. Although it does not fit every situation, there are some circumstances in which a method written in flowchart style is the best tool for the job. The purpose of this technical note is to acquaint you with the Flowchart Editor, and to suggest some situations in which you may want to consider using it.

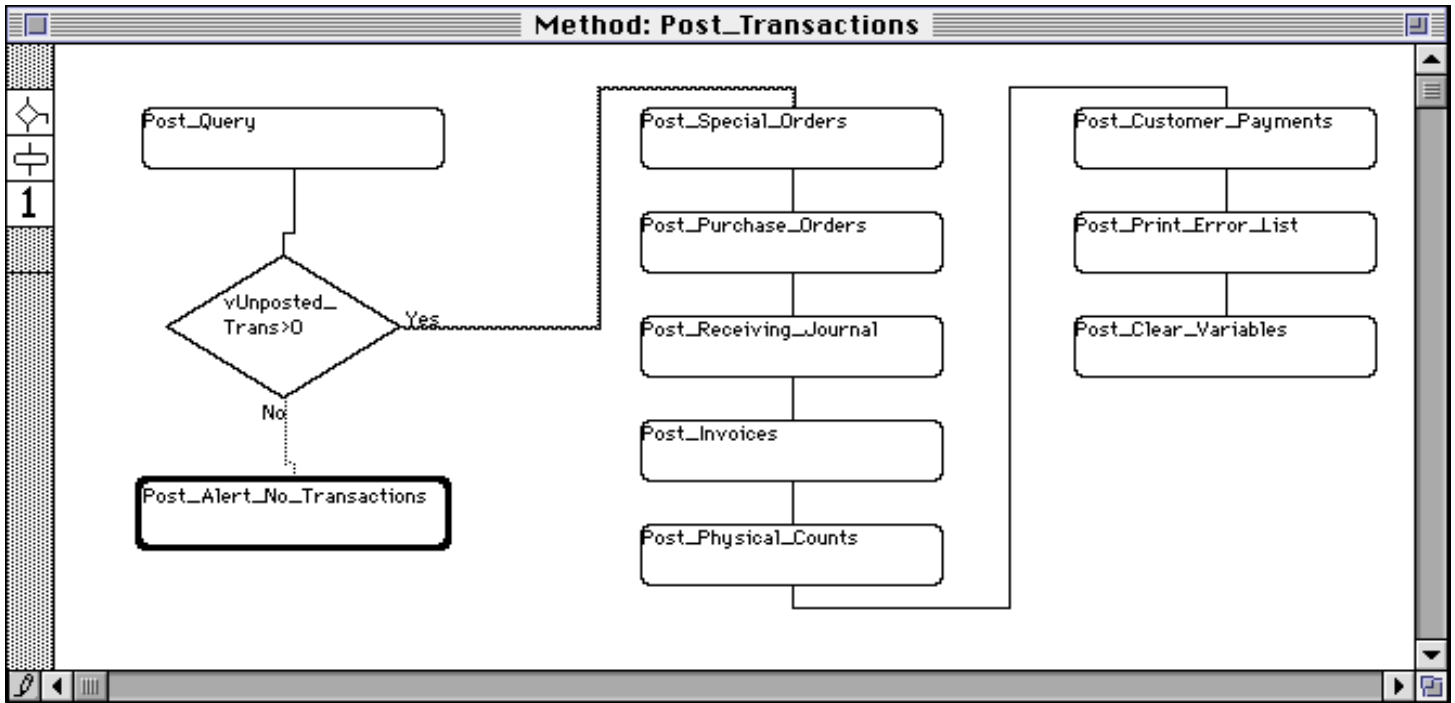
Why You Would Use the Flowchart Editor

In the world of programming, flowcharts are widely accepted and widely used. There are three reasons why flowcharts are so popular as developer tools:

¥ A flowchart is most useful when you need to plan, document, or understand a method that contains a complex series of steps, often involving calls to several other methods. In these cases, flowcharting helps you to think through the complex series of programming tasks, and to plan the sequence in which operations will take place.

¥ Flowcharting helps you to identify flaws in your programming logic and/or your planned sequence of events.

¥ Flowcharting helps you to communicate your ideas to other developers. If you use 4D's Flowchart Editor to write a method, you can capture a screen shot or make a printout to document that method. This one picture can truly be worth a thousand words.



When You Would Use the Flowchart Editor

The preceding example is the flowchart of an actual batch posting routine in an inventory/invoicing/payment receipt system. This routine looks for various types of unposted transactions, then uses those transactions to update customer balances, inventory levels, and statistical summaries. The entire routine involves calls to several dozen other routines, totaling more than one thousand lines of code. This is an extremely complex process; nevertheless, using the Flowchart Editor, we are able to describe this routine in a single method. From this example, you can see the primary advantage of the Flowchart Editor: it enables you to communicate, in one picture, the steps involved in a very complex operation. If you come back to this routine later – six months or a year after you wrote it – you will immediately grasp what the routine is doing.

The Flowchart Editor and Structured Programming

The term “structured programming” was a popular buzzword among programmers several years ago. It means different things to different people, so it might be a good idea to clarify exactly what we mean. In this technical note, we use the term structured programming to mean the technique of dividing your project methods into logical “paragraphs.” You use method calls to access those paragraphs when they are needed, rather than putting everything into one large method.

Structured programming has two primary advantages:

More Efficient Use of Memory in 4D

Whenever 4D calls a method, 4D retains that method on the process stack until the method has finished executing. The larger a method (in kilobytes) and the longer it needs to remain in memory, the less efficient it is in memory usage. As you might imagine, this can be a serious issue with a batch posting routine that might run for several hours. On the other hand, if you write one master method that calls several other methods as it steps through the execution process, each

ÔchildÓ methods does its job and releases the memory back to 4D.

Easier to Modify

Remember that your code, no matter how well-written, is not likely to be permanent. All things change, and that includes your 4D code. You should expect your project methods to change, and you should do whatever you can to make your programs easy to modify. Because it is easy to read and understand, a structured program is far easier to modify than the same program written in a non-structured manner. The 4D Flowchart Editor is an excellent aid to writing structured programs.

How to Use the Flowchart Editor

In this section, we take you through the major steps involved in writing the method Post_Transactions with the Flowchart Editor.

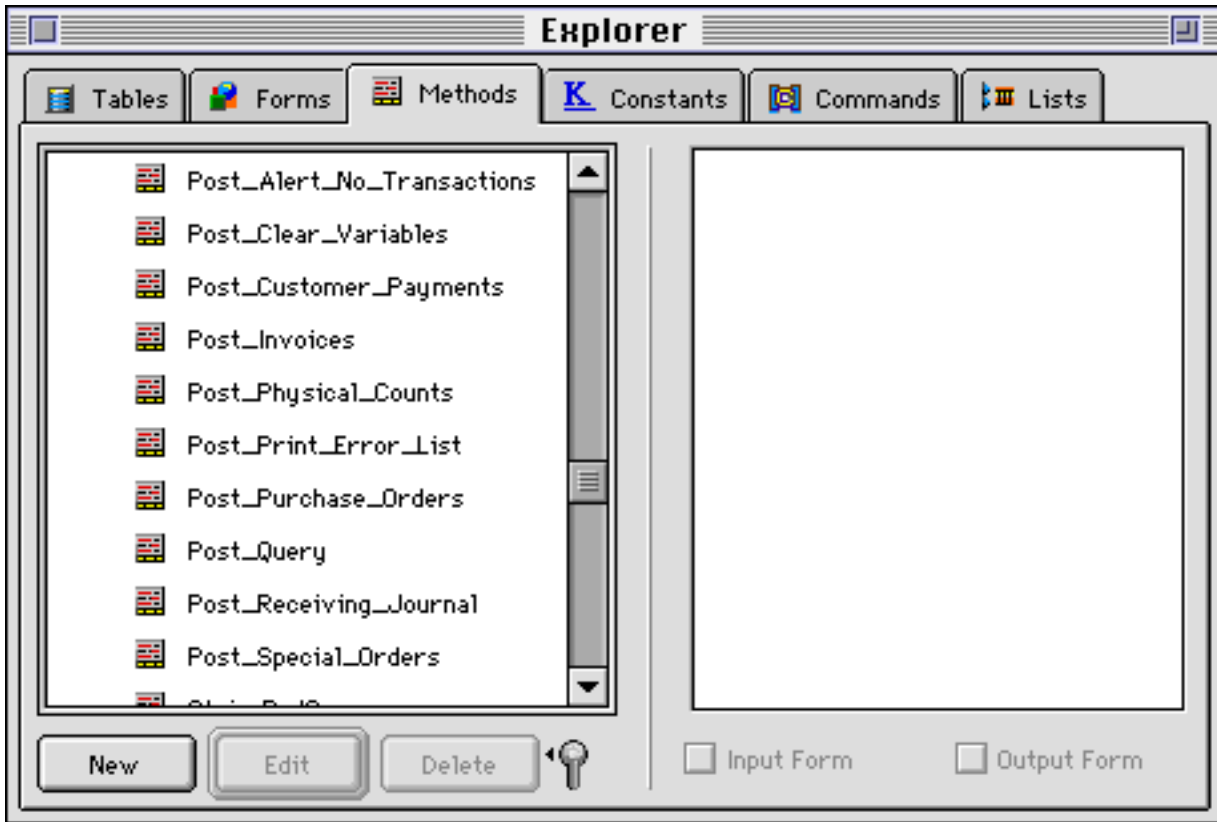
1. Begin with a tentative game plan.

When you want to write a method that is appropriate for the Flowchart Editor (one that calls several other routines), you should start with a general idea of what methods you will call, and in what order. This is only a starting point. If you change your mind later, the Flowchart Editor makes it easy for you to add, delete, or change the order of method calls.

2. Create the ÔchildÓ methods that you will call.

All you need to do at this stage is to create the methods so their names will appear in the list within the formula editor. If you are not ready to do so, you do not need to write the detailed code within the child methods. You can save that part for later.

Here is the list of the child methods that we created.

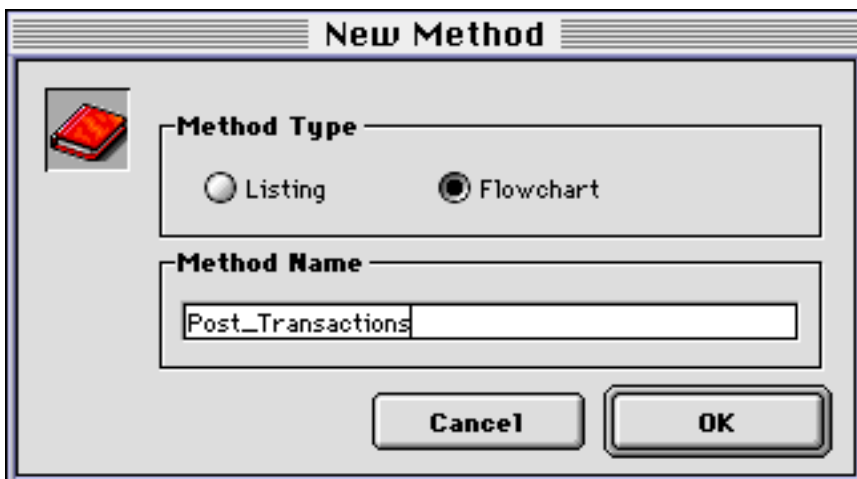


Note that all the method names begin with the prefix Post_. Also note that you can create each of these methods with the Listing Editor or with the Flowchart Editor, whichever is appropriate. If a child method will merely call other methods, then you can create it in the Flowchart Editor. If a child method will contain several detail lines of code, then you should create it with the Listing Editor; the Flowchart Editor is not appropriate for any method that needs several detail lines of code.

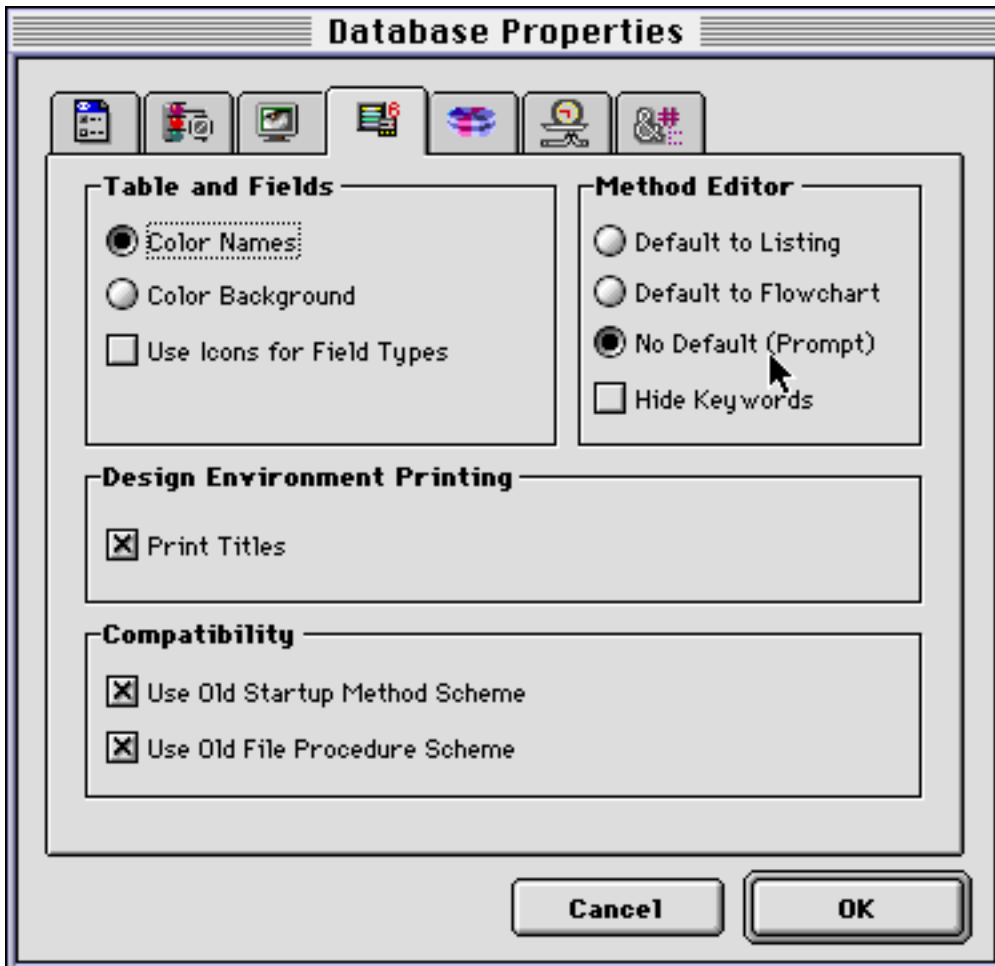
3. Click New to create a new method.

The New Method dialog appears.

4. Choose the Flowchart method type and name the method Post_Transactions.



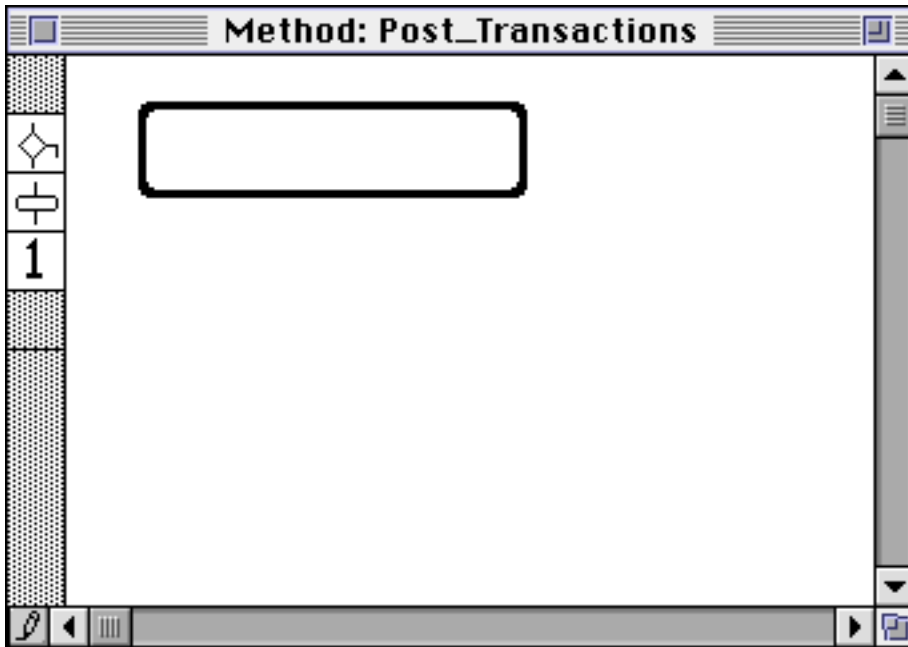
Note: If 4D did not give you the option to create a Listing or a Flowchart, it means that your Method Editor default is set to Listing Editor. You need to go to the Database Properties window, Design Environment page, and change the setting to No Default (Prompt).



5. Click the New Step icon.

6. Move the cursor to the upper left part of the window and click within the window to create a new Step icon.

A Step Icon appears.

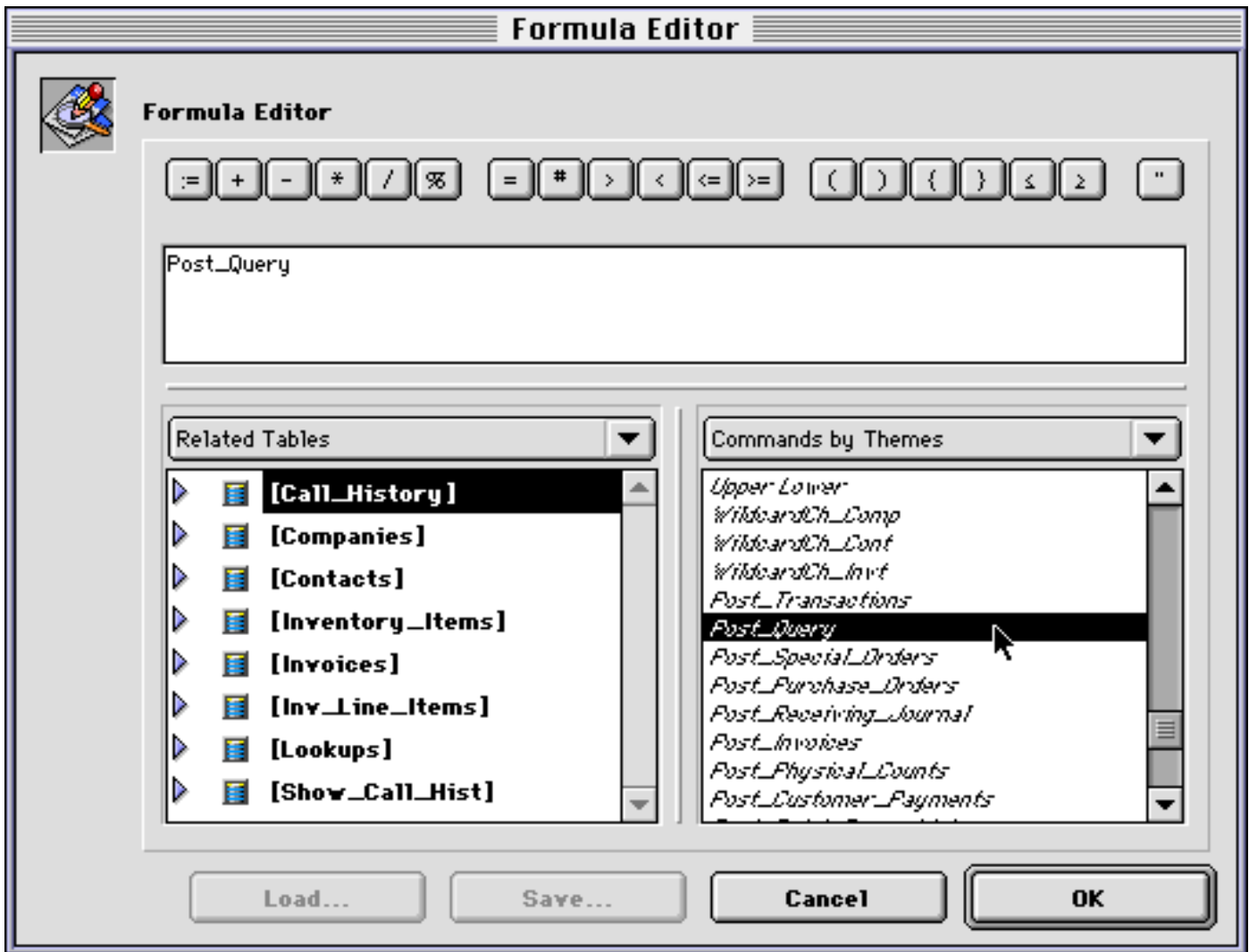


7. Double-click the Step Icon.

The Formula Editor appears.

8. Scroll down the list of methods and choose the first "child" method that you will call in this sequence.

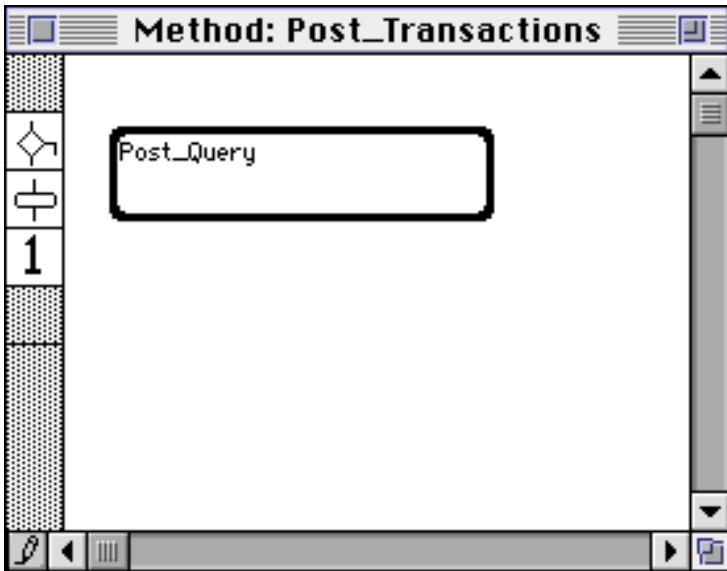
In our example, the first child method that we will call is named Post_Query.



9. Click OK.

Note: The Formula Editor only supports single-line statements such as Boolean expressions and calls to other methods.

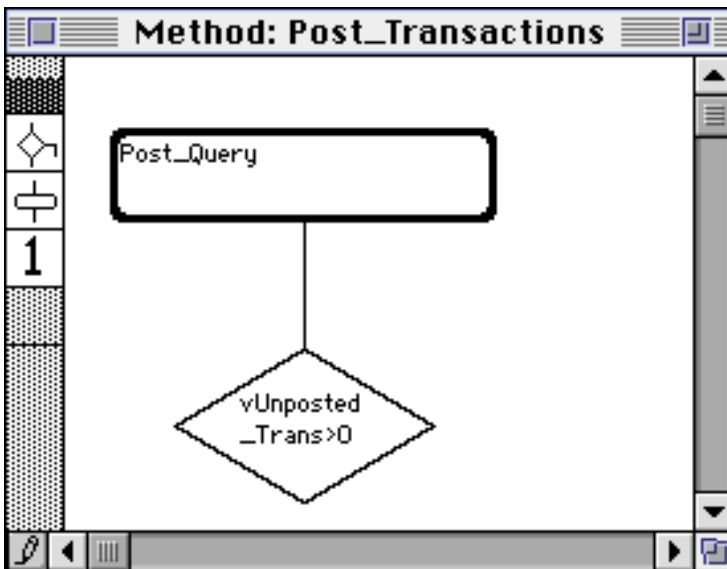
The first step now contains a call to the method Post_Query.



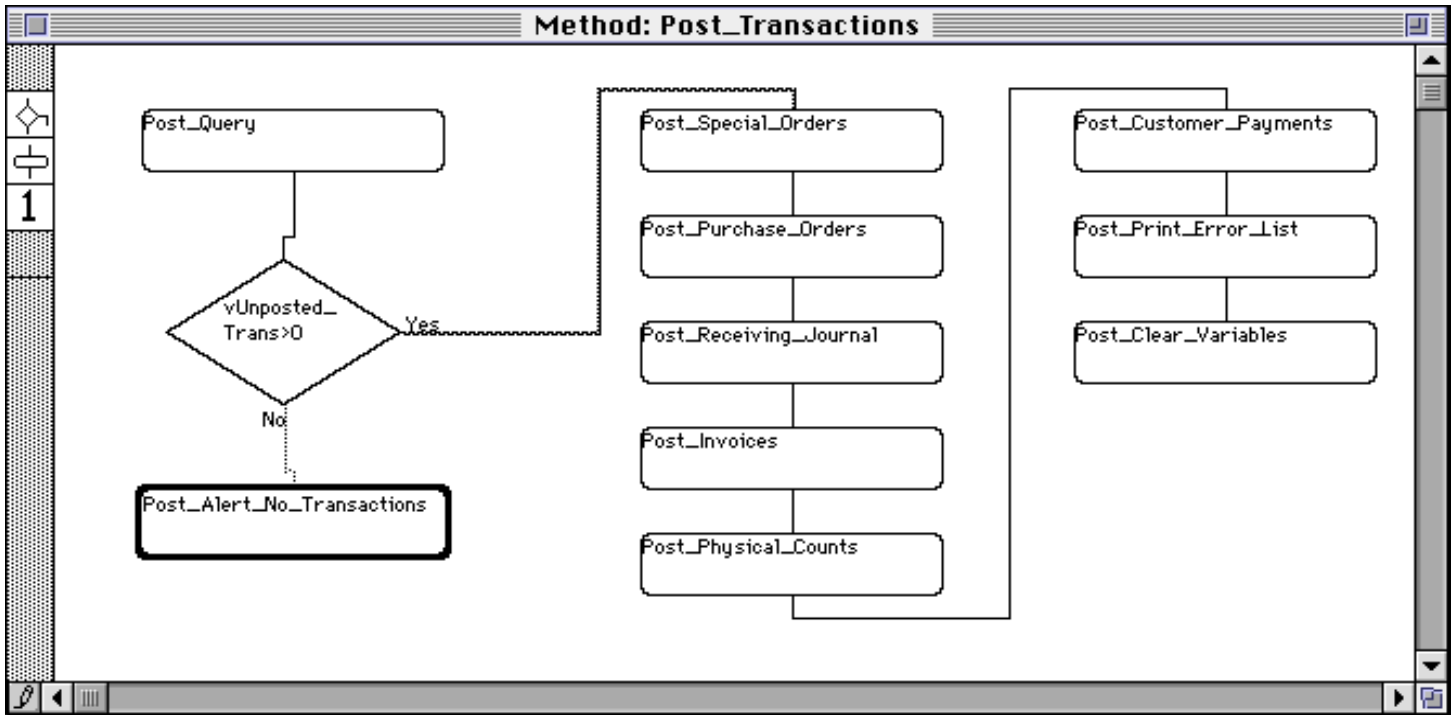
10. Click a tool to create the next icon.

In our case, the next step is a Test, so we create a Test icon and enter an expression that will evaluate as a Boolean (True or False).

11. Draw a line from the first icon to the second icon.



12. Continue to build the method until all the icons are drawn and connected.



When you complete the flowchart, you will have a simple and powerful self-documenting 4D method. Any 4D developer will be able to open this method and, after a brief look, understand what the method is doing.

Summary

If you are like most 4D developers, you probably seldom, if ever, use 4th Dimension's Flowchart Editor. In this technical note, we have introduced the Flowchart Editor, and have suggested some situations in which you may want to consider using it. When you need to write a long, complex method that calls several other methods, the Flowchart Editor could be the perfect tool for the job.