Writing Pascal Scripts

> **Orixa allows developers to add Pascal Scripts to the Object Properties of certain types of Resources. When this is done the user can click a button to run the script the developer has written. This allows very extensive customization of any Orixa App.**

> **Pascal Scripts are added into the Orixa script-editor.**
> **Writing scripts is an extremely complex process. The following topic is intended as an introduction to basic aspects of script-writing.**
> **As with many other processes in Orixa, Developers are encouraged to review scripts present in their own App, and available in on-line examples. It is always possible to test and extend your own scripts in your App using test data.**
> **Developers with experience writing any other scripted computer code should be able to rapidly understand the basics of Orixa Pascal Script.**

Why use Pascal?

Pascal is not a massively popular programming language, but it is very clear and fairly easy to learn in comparison with other languages. Orixa is wriiten in Delphi, which uses Pascal. Developers wish to graduate to writing their own App source-code they will have to do this in Pascal, so it is good to learn some through the scripting process.

On-line Resources to learn Pascal

Pascal is an easy-to-learn computer language, which uses a close-to-human style language syntax. It was originally developed in the 1970s, and was one of the first **strongly typed** languages. Being strongly typed allows it to convey complex data and be highy testable and easier to debug than some other languages.

Link to on-line Pascal teaching and information

Pascal On-line Tutorial This is an absolutely vast archive of Pascal resources, the subsections called "Decision Making" and "Loops are particularly useful for beginners.

Pascal Wikipedia This gives a short background and history of the language.

Why use Pascal Script in an Orixa App?

When a user clicks "Post" in an Orixa Rapid Entry Grid the program usually wants to do very simple things:

> 1. Take values from a row in the data-grid the user has edited, and check them. If entries are wrong or don't make sense show the user a message to re-enter or change their data-entry.
> 2. If data has been entered correctly, post new records, or insert data to one or more data-tables in the App database.
> 3. Repeat the above steps for all the rows in the data-grid the user is editing.

If a Developer can undertake these actions they can definitely write useful Rapid-data-entry scripts.

Most Apps include at least a small number of Pascal Scripts, linked to Rapid-entry-grid Resources. A Developer with basic skills can take these Resources and repurpose the pascal scripts to work with new Resources.

There are a number of other actions which can be useful:

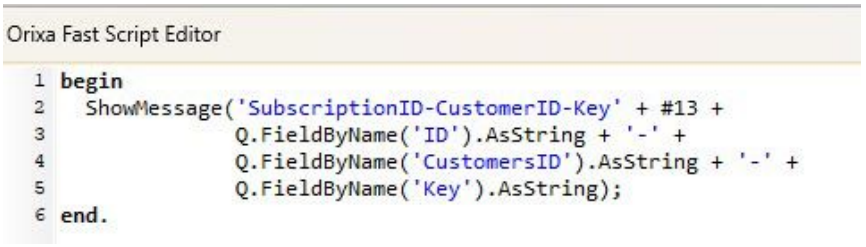
> 1. Ask a user for input, in the form of some text or numbers.
> 2. Convert string-data in the grid into Orixa-IDs that set Linkages between data.
> 3. Add other data to the database as you go along, such as audit data recording what the user has done.

These additional skills allow the user to elevate their programming and start to make their coding more responsive. The following examples show examples of scripts which undertake the above actions.

Example Scripts

Simplest Script: ShowMessage

The following script calls "ShowMessage('some string value')", to demonstrate basic features of all scripts.



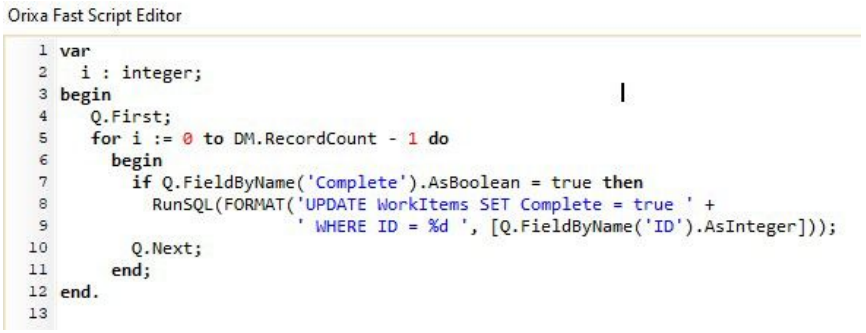
ShowMessage Pascal Script

```
begin
  Showmessage(Q.FieldByName('ID').AsString + '-' +
              Q.FieldByName('CustomersID').AsString + '-' +
              Q.FieldByName('Key').AsString);
end.
```

- All Scripts start with "begin" and end with "end." between these start / finish markers, a script can contain additional "begin / end" sections, but these all end with a semi-colon ";"
- The semi-colon ";" is the symbol used to mark the end of a statement.
- The **method** "Showmesssage" is one of many built in pascal methods the programmer can use. Showmessage does exactly what it says, it opens a message-dialog box with the text the programmer has detailed.
- Within the brackets of the ShowMessage call, a set of string variables and constants are concatenated together to create a single string. Note the use of "#13", this is a special character which will add a new-line / paragraph break. Strings can be concatenated using the "+" operator.

Iteration of records in a grid to update rows depending on values entered by user

The following script is contained in a Rapid Entry Grid resource. When the App executes this resource a data-grid is opened using the SQLStr of the resource, and the script is passed to the



Example Pascal Script Iteration

```
var
  i : integer;
begin
  Q.First;
  for i := 0 to DM.RecordCount - 1 do
    begin
      if Q.FieldByName('Complete').AsBoolean = true then
        RunSQL(FORMAT('UPDATE WorkItems SET Complete = true ' +
                      ' WHERE ID = %d ', [Q.FieldByName('ID').AsInteger]));
      Q.Next;
    end;
end.
```

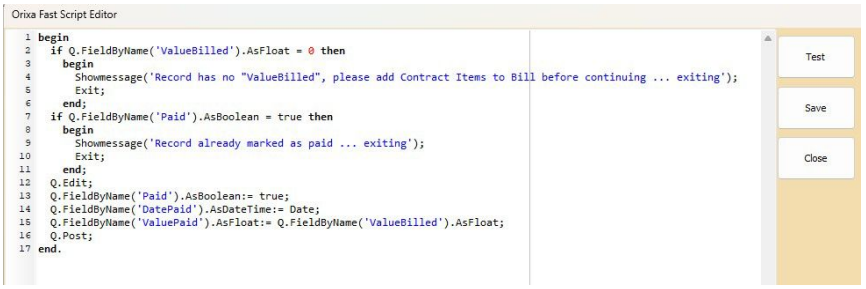
The above short script shows all the basic language elements required for any basic pascal script in an Orixa App.

1. If a script requires **variables** these must be declared before the "begin - end" after the **var** keyword. Data-types of variables are detailed in other help topics.
2. If a variable is used in a script it must be declared in the "var" section at the top of the script. In the above script the simple variable "i" (a counter) is declared at the top, and then used to count the number of

2. times a section of the script will run.
3. Within Pascal there are many, widely used elements of **language syntax**, such as "if", "for", "case" etc.
4. Within Orixa Scripts there are a good number of **keywords** and **variables** which allow the Developer to refer to values in the Orixa App.

Call to update values in a Business Object

The following script is called from a Business Object Record. When the user clicks on the Action in the Edit Form Actions menu, the script runs, referencing the data in the Edit Form.



Example Pascal Script Edit / Update

Note that the "Q" object in the script is the Query of the Edit Form, so the current values of any field can be referenced and updated using the syntax "Q.FieldByName('Fieldname').AsString:= 'some value';. FieldByName('FieldName') has properties AsString, AsFloat, AsInteger, AsDate and asBoolean for different basic data-types.

```
begin
  if Q.FieldByName('ValueBilled').AsFloat = 0 then
    begin
      Showmessage('Record has no "ValueBilled", please add Contract Items to Bill before continuing
... exiting');
      Exit;
    end;
  if Q.FieldByName('Paid').AsBoolean = true then
    begin
      Showmessage('Record already marked as paid ... exiting');
      Exit;
    end;
  Q.Edit;
  Q.FieldByName('Paid').AsBoolean:= true;
  Q.FieldByName('DatePaid').AsDateTime:= Date;
  Q.FieldByName('ValuePaid').AsFloat:= Q.FieldByName('ValueBilled').AsFloat;
  Q.Post;
end.
```

Note the call to "Q.Edit" prior to actually changing values in the dataset, and the call to "Q.Post" to actually post the data to the database.

Pascal Language Syntax

Links to useful Pascal Language Syntax:

"IF" Statements

"CASE" Statements

Using local variables:

The following code declares a number of variables of different data-types, which can then be used in your script:

```
var
  age, weekdays : integer;
  taxrate, net_income: real;
  choice, isready: boolean;
  initials, grade: char;
  name, surname : string;
```

Explanations of "data-types" such as "integer" and "real" can be found on-line. Pascal data-types have a strong overlap with data-types used in SQL database programming, which Orixa Developers also need to be familiar with.

Orixa Keywords and Variables

Your Pascal script must refer to data in your App in order to be useful. Orixa has added a set of**system-variables and classes** to the scripting process which allow programmers to do this.

When writing a script the coder puts together "Objects" and "Methods" or "Properties", an Object is something you can reference by writing a keyword. Adding a dot after the keyword allows you to then add a reference to a property of the object.

In the script:

```
Q.First;
```

The "Q" (data-Query) object "calls" the method "First" which triggers the program to move to the first row in the rapid-entry-grid.

Name of Variable	What it means	How it is used
Q	Use the keyword Q to reference the Query data in the rapid entry grid.	Q.FieldByName('MyField').AsInteger; Q.FieldByName('MyField').AsString; Q.First; Q.Next;
DM	Use the keyword "DM" to reference the "datamodel" of the data. There are a number of useful methods which can be called from the DM Variable.	DM.RecordCount; DM.ID;
RunSQL(' <SQL Statement> ')	Use the RunSQL method, followed by a pair of brackets enclosing a SQL Statement to run SQL on the database. If the statement is an **UPDATE** or **INSERT** statement, this will result in changes to the database.	RunSQL(' INSERT INTO People ' + '(FirstName, LastName)' + 'VALUES' + '(''John'', ''Smith'' ');
RunScript('<SQL Script>')	Use the RunScript method, followed by a pair of brackets enclosing a SQL Statement to run a SCRIPT on the database. The difference between RunSQL and RunScript is that Run Scipt allows **stored procedures** to be referenced in the SQL using the **CALL** keyword.	RunScript('SCRIPT BEGIN + CALL MyProcedure(); END ')

Confirm('<User Message>')	Use the Confirm method, followed by a pair of brackets enclosing a confirmation question you want to ask the user.
	
	If the user clicks "yes" the function returns "true" if they click "no" it returns "false".
	Use the Confirm function to make **conditional logic** like **if then else** to work in your scripts.
		if Confirm('Are you sure?') then
		begin
		end
		else
		begin
		end;

Taking your first steps in writing Pascal Scripts in Orixa

1. Don't be scared of trying. Remember the Scripting window has a "Test" button which will try to**compile** your script and will return an error message if you make a mistake. This compilation process**does not** actually run the script, it just tests that it **can** run.
2. Writing scripts is very, very precise. A single missed semi-colon, single quote or mis-spelling will make a script unworkable, but error messages should help you to find mistakes.
3. Some aspects of writing scripts just require experience to get used to them and avoid basic errors. For example when putting together elements of a **string** variable it is easy to make mistakes by forgetting the "+" at the end of each line, or forgetting that single-quotes must be doubled-up within a string to be readable. As you become more experienced you will stop making these types of errors.

4. Use **comments.** Any line of a script which starts with "//" or and section which is enclosed with "/* */" will **not** be included when a script is compiled or run. This means you can add notes within your script which explain what it does and how it works. Comments are **always** a good idea!

More script examples

Summarized Basics of Pascal Language are detailed here:

Pascal Language and statement structure

Summarized Basics of Pascal Language are detailed here:

Pascal Functions and Procedures

Summarized Basics of Pascal Language are detailed here:

Pascal Example with OLE