Pascal Script basic functions and procedures

> **The following lists some of the most useful basic functions and procedures which are available within the scripting-engine.**
> **Developers will probably find it useful to play with these commands to fully understand how they work.**

Showmessage('Hello World'); Inform your user with a simple "pop up message" using this statement.

ShowmessageFMT('Hello user: %s', [StringVariable]); Inform your user and pass in a variable from your program into the message.

IntToStr(i: Integer): String use this function to convert a whole-number into string data, for example to add it into a SQL update statement.

FloatToStr(e: Extended): String use this function to convert a number with a fractional part into string data, for example to add it into a SQL update statement.

DateToStr(e: Extended): String use this function to convert a date into string data, note that dates are stored in Orixa as a complex data-type. They are not stored as text.

TimeToStr(e: Extended): String use this function to convert a time into string data, note that dates are stored in Orixa as a complex data-type. They are not stored as text.

DateTimeToStr(e: Extended): String use this function to convert a date with a time into string data.

VarToStr(v: Variant): Stringuse this function to convert variant data into string data.

StrToInt(s: String): Integer use this function to convert a string into a whole number. Note that if the string does not contain a whole number this will produce an error.

StrToFloat(s: String): Extended use this function to convert a string into a number with a fractional part. Note that if the string does not contain a number this will produce an error.

StrToDate(s: String): Extended use this function to convert a string to a date. Note that the string must contain a properly formatted SQL date. SQL dates are presented in the form YYYY-MM-DD, Year-Month-Day. For example 15th November 2022 would be written 2022-15-11.

StrToTime(s: String): Extended use this function to

StrToDateTime(s: String): Extended use this function to

Format(Fmt: String; Args: array): String use this function to insert variables into a section of text.

FormatFloat(Fmt: String; Value: Extended): String use this function to

FormatDateTime(Fmt: String; DateTime: TDateTime): String use this function to

FormatMaskText(EditMask: string; Value: string): stringuse this function to

EncodeDate(Year, Month, Day: Word): TDateTimeuse this procedure to

DecodeDate(Date: TDateTime; var Year, Month, Day: Word)use this procedure to

EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime use this procedure to

DecodeTime(Time: TDateTime; var Hour, Min, Sec, MSec: Word) use this procedure to

Date: TDateTime use this function to return the current system date.

Time: TDateTime use this function to return the current system time.

Now: TDateTime use this function to return the current system time.

DayOfWeek(aDate: DateTime): Integer use this function to extract the number of the week-day from a date.

IsLeapYear(Year: Word): Booleanuse this function to return true if the year is a leap year and false if it is not. The year would be passed in as a 4 digit number.

DaysInMonth(nYear, nMonth: Integer): Integeruse this function to return the number of days in a month.

Length(s: String): Integer use this function to return the length of a string.

Copy(s: String; from, count: Integer): String use this function to copy-out part of a string.

Pos(substr, s: String): Integer use this function to confirm whether a particular string of data is contained in another string.

Delete(var s: String; from, count: Integer): String use this procedure to delete a section of a string of data.

Insert(s: String; var s2: String; pos: Integer): String use this procedure to add text into a section of a string of data.

Uppercase(s: String): String use this function to convert a string to uppercase text.

Lowercase(s: String): String use this function toconvert a string to lowercase text.

Trim(s: String): String use this function to

NameCase(s: String): String use this function to

CompareText(s, s1: String): Integer use this function to

Chr(i: Integer): Char use this function to

Ord(ch: Char): Integer use this function to

SetLength(var S: String; L: Integer) use this function to

Round(e: Extended): Integer use this function to

Trunc(e: Extended): Integer use this function to

Int(e: Extended): Integer use this function to

Frac(X: Extended): Extended use this function to

Sqrt(e: Extended): Extended use this function to

Abs(e: Extended): Extended use this function to

Sin(e: Extended): Extended use this function to

Cos(e: Extended): Extended use this function to

ArcTan(X: Extended): Extended use this function to

Tan(X: Extended): Extended use this function to

Exp(X: Extended): Extended use this function to

Ln(X: Extended): Extended use this function to

Pi: Extended use this function to return the value of Pi.

Inc(var i: Integer; incr: Integer = 1) use this procedure to increase a variable by the amount of the second parameter.

Dec(var i: Integer; decr: Integer = 1) use this procedure to decrease a variable by the amount of the second parameter.

RaiseException(Param: String) use this procedure to stop a script and show an error message to the user.

Randomize use this function to return a random whole number.

Random: Extended use this function to return a random floating-point number.

ValidInt(cInt: String): Boolean use this function to confirm that a string variable is a whole number. This is useful to use prior to "StrToInt" to ensure no errors occur.

ValidFloat(cFlt: String): Boolean use this function to confirm that a string variable is a whole number.

ValidDate(cDate: String): Boolean use this function to check that a string is a valid date.

Fast Script include a substantial set of methods, procedures and properties the following are all workable

Q.Open this function calls the database query variable "Q", and returns data. The "Q" variable acts in the same way as a SQL cursor. Giving access to a dataset which the user can interact with. As well as "Open" there are many other functions and properties on the "Q" object: **Q.Close**, **Q.First**, **Q.Last**, **Q.Next**, **Q.Prior**, **Q.Cancel**, **Q.Delete**, **Q.Post**, **Q.Append**, **Q.Insert**, **Q.Edit** etc.

Data access and manipulation Example

```
var
i : integer;
begin
  Q.First;
  for i := 0 to DM.RecordCount - 1 do
    begin
      if Q.FieldByName('CheckedOnBankStatement').AsBoolean = true then
      RunSQL(FORMAT('UPDATE Expenses SET CheckedOnBankStatement = true, ' +
        ' Name = ''%s'', ' +
        ' "Value" = %f ' +
        ' WHERE ID = %d ', [Q.FieldByName('Name').AsString,
          Q.FieldByName('Value').AsFloat,
          Q.FieldByName('ID').AsInteger]));
      Q.Next;
    end;
end.
```

The above **example script** shows a simple case of use of the pascal script capabilities of Orixa.

A Resources record has returned a set of records, which can then be referenced by the script.

```
    for i := 0 to DM.RecordCount - 1 do
```

The "DM" (DataModule") variable holds the record count of the active resource dataset.

```
      if Q.FieldByName('CheckedOnBankStatement').AsBoolean = true then
```

The "FieldByName" property on the "Q" variable allows the programmer to test the value of a variable in the resource dataset, and run a set of code conditional on its performance.

```
        RunSQL(FORMAT('UPDATE Expenses SET CheckedOnBankStatement = true, ' +
```

The "RunSQL" function allows the programmer to call a SQL Statement to make changes to records in the database.

Other Properties and functions on the "Q" system variable

Q.FieldByName(const FieldName: string): TField return a field who's value you can test, **Q.GetFieldNames(List: TStrings)** return a list of all the fields in the query,

Q.FindFirst: Boolean, go to the first record **Q.FindLast: Boolean**, go to the last return **Q.FindNext: Boolean**, go to the next record (returns false if the user is already at the first record, **Q.FindPrior: Boolean**, go to the prior record. Other "field / data" related properties: **Q.FieldCount**, **Q.FieldDefs**, **Q.Fields**, **Q.IsEmpty: Boolean**, **Q.Bof**, **Q.Eof**, **Q.Filter**, **Q.Filtered**, **Q.FilterOptions**.

Properties and functions to find data

Use the programmatic "Bookmark" object to store a particular position in the data-set. **Q.FreeBookmark(Bookmark: TBookmark)**, **Q.GetBookmark: TBookmark**, **Q.GotoBookmark(Bookmark: TBookmark)**

```
var
  bm : TBookmark;
begin
  bm:= Q.GetBookmark;
  try
    //... some code to do work
  finally
  Q.GoToBookmark(bm);
end
```

Locating specific records

Use the **Q.Locate(const KeyFields: string; const KeyValues: variant)**function. The following function shows code to find a record with the ID = 123456.

```
begin
  Q.Locate('ID', 123456);
  //... some code to do work
end;
```