

## LogActivity event

### Specification

```
var LogActivity: procedure(ComponentType: Integer; AObject: TObject;  
    Flag, Info: Integer; const Description, SQL, ResultMessage: string;  
    Variables: TVariables; Parameters: TStrings; StartTime: LongInt);
```

### Description

This event is triggered before and after a database access function is called, so that you can log or monitor these events.

The **ComponentType** parameter indicates what type of component has triggered the action: CkSession, ckQuery, ckDataSet, ckPackage, ckEvent, ckLOBLocator, ckObject, ckReference, ckScript, ckLoader, or ckQueue.

The **AObject** parameter is the object that has triggered the action (e.g. a TOracleSession instance, a TOracleQuery instance, and so on).

The **Flag** parameter can be afStart when an activity starts, afEnd when an activity ends, or afSingle when an Activity does not have a start/end moment.

The **Info** parameter is for internal use only.

The **Description** parameter is a single line description of the activity. For example 'Session.Logon as scott@detroit', or 'Query.Execute'.

The **SQL** parameter contains the SQL text, if applicable.

The **ResultMessage** contains an error message, or is empty if the activity succeeded.

The **Variables** parameter is a list of bind variables, before (Flag = akStart) or after (Flag = akEnd) the activity.

The **Parameters** parameter is for internal use only.

The **StartTime** parameter is the start time of the activity in milliseconds, and can be used at the end of the activity (Flag = akEnd) to calculate the elapsed time of the activity.

## Example

After writing your log procedure, assign it to the global LogActivity event:

```
Oracle.LogActivity := @LogDBActivity;
```

There is no need to include the OracleMonitor unit.

An example of a log procedure:

```
procedure LogDBActivity(ComponentType: Integer; AObject: TObject; Flag, Info: Integer;
  const Description, SQL, ResultMessage: string; Variables: TVariables;
  Parameters: TStrings; StartTime: LongInt);
var Lines: TStringList;
    t: LongInt;
    s: string;
    C: TComponent;
    v: Integer;
    VD: TVariableData;
begin
  procedure AddToLines(const Prefix, s: string);
  begin
    if Trim(s) <> '' then Lines.Add(Prefix + TrimRight(s));
  end;
begin
  try
    // Create an empty log text
    Lines := TStringList.Create;
    // Concatenate the object name and activity description
    s := AObject.ClassName;
    if AObject is TComponent then
    begin
      s := s + ' ';
      C := AObject as TComponent;
      if C.Owner <> nil then s := s + C.Owner.Name + '.';
      s := s + C.Name;
    end;
    case Flag of
      afStart: s := s + ' Start';
      afEnd: s := s + ' End';
      afSingle: s := s + ' Single';
    end;
    AddToLines('', s + ' ' + Description);
    // Add the SQL
    AddToLines('SQL = ', SQL);
    // Log the input variable names, types, and values
    if (Flag = afStart) and (Variables <> nil) and (Variables.Count > 0) then
    begin
      for v := 0 to Variables.Count - 1 do
      begin
        VD := Variables.Data(v);
        s := VD.Name + ' = ' + VD.DisplayString(AObject);
        Lines.Add(s);
      end;
    end;
    // Log the result message
    AddToLines('Result = ', ResultMessage);
    // Log the elapsed time
    if Flag = afEnd then
    begin
      t := Integer(GetTickCount) - StartTime;
      AddToLines('Duration = ', FloatToStr(t / 1000));
    end;
    if Lines.Text <> '' then DebugLog(Lines.Text);
    Lines.Free;
  except
    // Always handle exception
  end;
end;
```