System "Actions" - Stored-procedures surfaced in the App

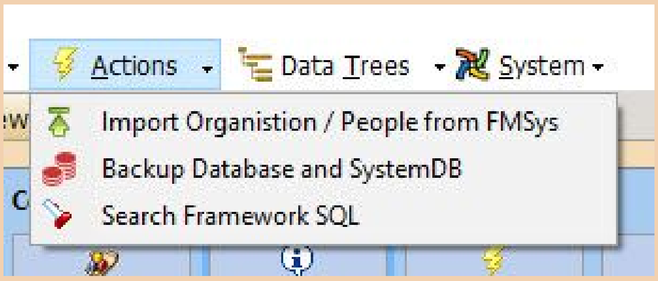
Your system includes a set of program features which have been added to the main toolbar. How to access them is detailed below.
Orixa Developers can add extra entries to this "Actions" menu-list by creating Stored Procedures in the database and setting their Description Properties so that the App can install them.
Stored-procedures can be added to the App as a whole, or linked to BusinessObjects, or targetted to the level of individual records of BusinessObjects. At each target-level the properties of the target can be accessed by the Procedure if needed.

This is a powerful feature of Orixa as it allows Developers to update and change the Actions users can execute in different parts of the App without the need for rebuilding the source code of the App.

These Stored-procedures can be used to insert new data, import data, find a record and display it, display grids of data after other actions have been undertaken and many for other uses.

As each system is different, the actions they contain will all be unique. To find details of the specific functions performed by your own system's System Actions, please review the Help below.

System Actions

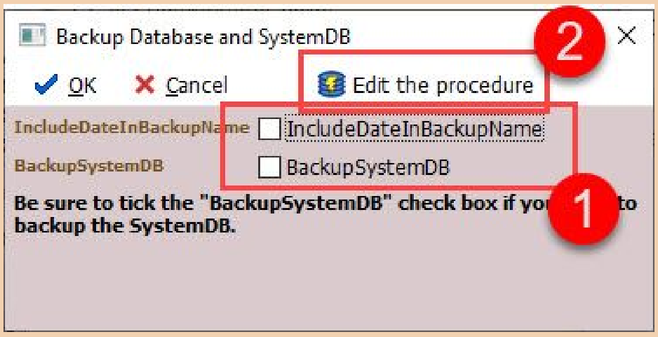


Actions
Data Trees
System
Import Organistion / People from FMSys
Backup Database and SystemDB
Search Framework SQL

System Actions Menu
Note the content of this menu is created dynamically, and will vary between different Apps

Accessing the "System Actions"

1. From the main toolbar click on the "Actions" icon.
2. The list of specialized actions that has been added to your system will display below.



Be sure to tick the "BackupSystemDB" check box if you want to backup the SystemDB.

System Actions Procedure Form

The "Backup Database and SystemDB" Action

Once an item is selected from the menu, a screen will show allowing the user to undertake the selected action.

1. User editable elements are listed.
2. System Administrators have access to edit the procedure. (This button is only visible when System Administrators are running the procedure, otherwise it is not shown).

System Actions can be created by System Administrators. Please consult items in the "Developer Guide" to see how this is done.

How are System-Actions Created?

A System Action is simply a Stored Procedure on your Orixa Database which has been decorated with a**Description** that allows it to be accessed by the App.

For the "Backup Database and SystemDB" System Action shown above, here is the**SQL Definition** of the Stored Procedure:

```
PROCEDURE "BackUpDatabase" (IN "DatabaseName" VARCHAR COLLATE "ANSI", IN "IncludeDateInBackupName"
BOOLEAN, IN "BackupSystemDB" BOOLEAN)
```

```
BEGIN
DECLARE aBackupName VARCHAR;
IF IncludeDateInBackupName THEN
  SET aBackupName = DatabaseName + '-BU-' + CAST(Current_Date as VARCHAR(10));
  ELSE
  SET aBackupName = DatabaseName + '-BU';
  END IF;
EXECUTE IMMEDIATE
 ' BACKUP Database "'+ DatabaseName +'" AS   "' + aBackupName + '"
   TO STORE "Backup" INCLUDE Catalog ';
IF BackupSystemDB THEN
   EXECUTE IMMEDIATE
  ' BACKUP Database "SystemDB" AS
  "' + aBackupName + '-SystemDB"
  TO STORE "Backup" INCLUDE Catalog ';
  END IF;
END
DESCRIPTION
'[Properties]
Type=System
Title=Import Organistion / People from FMSys
LinkTable=
UserMessage=Find an ID for an Organisation from FMSys, input this and the procedure will pull over
data from FMSys non-"Current" data is ignored
CheckFirst=true
SecurityLevel=40
ImageIndex=265';
```

The above **SQL Definition** creates the Stored Procedure and sets the **Description** to a value that means your Orixa App will load it at start-up.

Notes: How to use the "Properties" saved with a Stored-procedure to control where the Procedure appears in your App

Type=System
This means that the stored procedure will be loaded into the App's main menu. The developer can use "System", "Entity" or "Record" after the word "Type=" If Entity is selected the Stored Procedure will be added to the Entities Screen, and linked to the BusinessObject named in the "LinkTable" Property.

Title=
Add a title to be displayed in the App, and to be shown on the execute-procedure window.

LinkTable=
Add the name of any BusinessObject data-table in the system. The Stored Procedure will then be linked to this data-table. This means it will be displayed in the App wherever this BusinessObject is displayed. It also means that values from the BusinessObject will feed into parameters of the Stored Procedure.

UserMessage=
Add an (optional) message to be displayed in the execute-procedure window.

Checkfirst=[true|false]
If "true" the user is asked to confirm: "Run Procedure [name]. Are you sure?" prior to execution.

SecurityLevel=[integer value 0 - 100]
If a user has a SecurityLevel **below** the value named, this Stored Procedure will **not** be shown as an Action when they log on to the App.

ImageIndex=[integer value 0 - 300]
Selects which App Icon is displayed when displaying the Stored Procedure Title.

> **When your Orixa App Starts all Stored Procedures that have correctly formatted Descriptions will be added to your App User-Interface.**

What happens when a user selects a Stored Procedure from within an App?

1. The App reviews the **Parameters** of the Stored Procedure. If the names of these parameters match the names of a field in the Linked data-table the value from the currently open record will be substituted as the value to be used in the procedure. If no matching field-name is found, a user-entry field is added to the execute-procedure window. When the user clicks "OK" the parameter values added by the user will be passed into the procedure.

2. The window is opened and awaits user input. If the "Cancel" button is clicked, nothing happens. If the "OK" button is clicked, the procedure is excuted.

Special Parameters which can be accessed by Your Orixa App if they are present in your Stored-procedure

When a procedure is run, your Orixa App will review the **Parameters** it contains. Orixa will look for a few special parameter-names and if it finds them its behaviour will change.

Input-Parameter: "ID"

If a procedure is run at the level of a Record, a Parameter called "ID" will be automatically filled in with a value equal to the ID of the selected record. This allows procedures that undertake work such as inserting child-data records.

Output-Parameter: "Result"

Note this is slightly confusing, as "Result" is the general term used with Stored-functions, in this case the "Result" is somewhat different, it can be **any type of value**. If the Stored-procedure includes a parameter called "Result" its value will be shown to the user in a standard message box after the Stored-procedure has run.

Output-Parameter: "UserMessage"

After the Stored-procedure has run, if the parameter "UserMessage is not blank, it will be shown to the user in a standard message box.

Output-Parameter: "NewID"

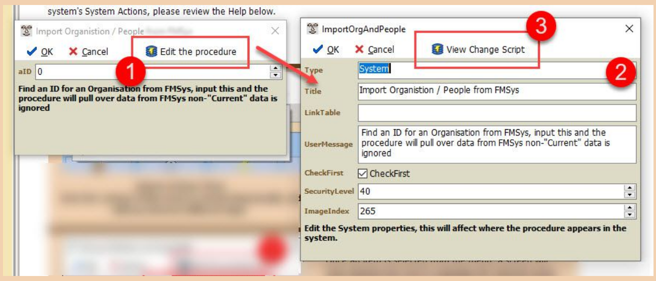
After the Stored-procedure has run, if the parameter "NewID" is not blank, the App will attempt to open the record with ID = NewID in the data-table named by the "LinkedTable" Property of the procedure.

Using In-App Tools to edit a System Accessible Stored Procedure

Stored-procedures can be created and altered in the DB Utility, either within your App or with your App closed. Just write a CREATE or ALTER statement for the procedure in the SQL Editor window. You can copy the Definition of another procedure if needed.

In addition to this, Admin users with high Security Level, will see an "Edit this procedure" button in the "Run Procedure" window. **This button is not visible to normal users.**

Using the Edit this Procedure Button



system's System Actions, please review the Help below.
Import Organistion / People
OK
Cancel
Edit the procedure
1
ImportOrgAndPeople
OK
Cancel
View Change Script
3
2

aID 0

Find an ID for an Organisation from FMSys, input this and the procedure will pull over data from FMSys non-"Current" data is ignored

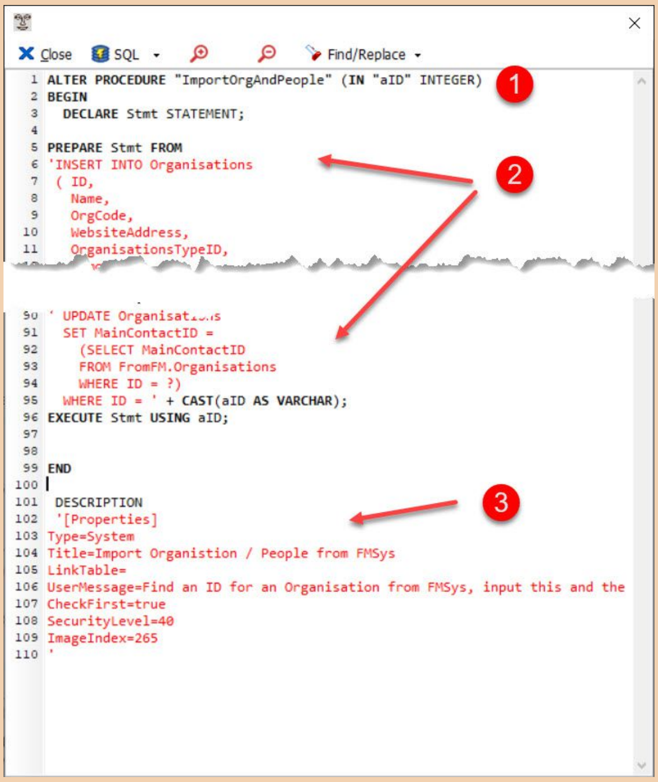
Type: System
Title: Import Organistion / People from FMSys
LinkTable:
UserMessage: Find an ID for an Organisation from FMSys, input this and the procedure will pull over data from FMSys non-"Current" data is ignored
CheckFirst: ☑ CheckFirst
SecurityLevel: 40
ImageIndex: 265

Edit the System properties, this will affect where the procedure appears in the system.

Editing System Procedures

1. From the window where the Stored Procedure is executed, click the "Edit the procedure" button. Note this button is only visible to Develope-level users.
2. A window opens with edits in which you can enter **all** Orixa decorations that can be added to a procedure. These map to the values saved in the Description of the Procedure (as explained above).
3. Once you have edited the decorations, you can click "View Change Script" to show the procedure.

1. From "View Change Script" a window opens with the whole **ALTER PROCEDURE** SQL statement.
2. A Developer can extend and ammend the main SQL of the procedure at this point, if they wish to change what the procedure does.
3. The **Description** which has been added matches the values entered in the previous screen. You can manually edit these further here if you wish.



```
ALTER PROCEDURE "ImportOrgAndPeople" (IN "aID" INTEGER)
BEGIN
  DECLARE Stmt STATEMENT;

PREPARE Stmt FROM
'INSERT INTO Organisations
 ( ID,
   Name,
   OrgCode,
   WebsiteAddress,
   OrganisationsTypeID,
```

```
' UPDATE Organisat...s
  SET MainContactID =
    (SELECT MainContactID
    FROM FromFM.Organisations
    WHERE ID = ?)
  WHERE ID = ' + CAST(aID AS VARCHAR);
EXECUTE Stmt USING aID;


END

 DESCRIPTION
 '[Properties]
Type=System
Title=Import Organistion / People from FMSys
LinkTable=
UserMessage=Find an ID for an Organisation from FMSys, input this and the
CheckFirst=true
SecurityLevel=40
ImageIndex=265
'
```

Editing System Procedure Alter Script

Once you are happy with the SQL, you can click the "SQL" button, selecct the choice on the menu to execute the SQL. This will update the procedure

If you make changes without executing immediately, when you return to the prior window you will be prompted to run the SQL so changes are not lost.

Using a Stored-procedure to return a grid of data

A Stored-procedure may have added data to a database. If this is the case it may be useful to view a grid of the inserted data.

SQL includes a special set of keywords to allow this.

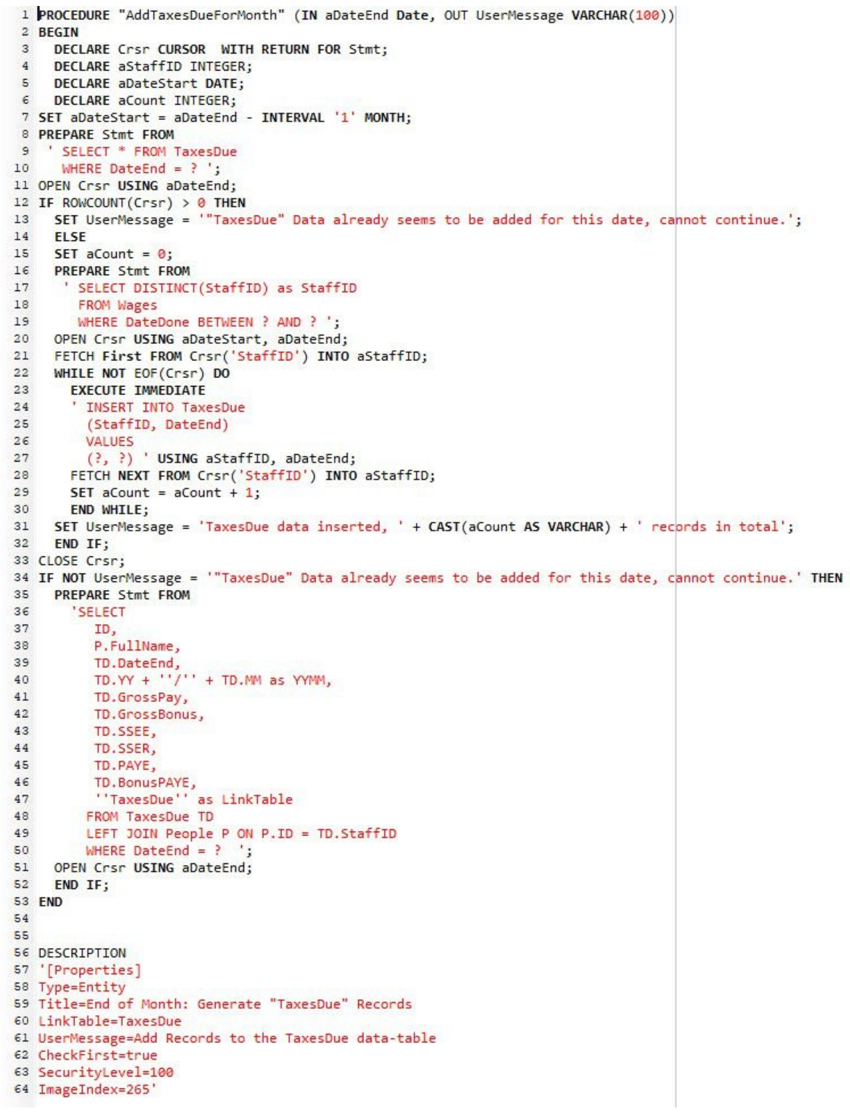
```
DECLARE Crsr CURSOR WITH RETURN FOR Stmt;
```

The Usual "DECLARE Crsr" keywords have the "WITH RETURN" keywords added. This results in a Stored-procedure which creates a grid of the records selected by the "Stmt" Statement and displays it to the user.

Note that for this to work, the Stored-procedure in question must PREPARE the Stmt using valid SQL and end with the line:

```
OPEN Crsr;
```

Example of a Stored-procedure which returns a grid, and uses the "UserMessage output parameter



```
PROCEDURE "AddTaxesDueForMonth" (IN aDateEnd Date, OUT UserMessage VARCHAR(100))
BEGIN
  DECLARE Crsr CURSOR  WITH RETURN FOR Stmt;
  DECLARE aStaffID INTEGER;
  DECLARE aDateStart DATE;
  DECLARE aCount INTEGER;
SET aDateStart = aDateEnd - INTERVAL '1' MONTH;
PREPARE Stmt FROM
 ' SELECT * FROM TaxesDue
   WHERE DateEnd = ? ';
OPEN Crsr USING aDateEnd;
IF ROWCOUNT(Crsr) > 0 THEN
  SET UserMessage = '"TaxesDue" Data already seems to be added for this date, cannot continue.';
  ELSE
  SET aCount = 0;
  PREPARE Stmt FROM
   ' SELECT DISTINCT(StaffID) as StaffID
     FROM Wages
     WHERE DateDone BETWEEN ? AND ? ';
  OPEN Crsr USING aDateStart, aDateEnd;
  FETCH First FROM Crsr('StaffID') INTO aStaffID;
  WHILE NOT EOF(Crsr) DO
    EXECUTE IMMEDIATE
    ' INSERT INTO TaxesDue
      (StaffID, DateEnd)
      VALUES
      (?, ?) ' USING aStaffID, aDateEnd;
    FETCH NEXT FROM Crsr('StaffID') INTO aStaffID;
    SET aCount = aCount + 1;
    END WHILE;
  SET UserMessage = 'TaxesDue data inserted, ' + CAST(aCount AS VARCHAR) + ' records in total';
  END IF;
CLOSE Crsr;
IF NOT UserMessage = '"TaxesDue" Data already seems to be added for this date, cannot continue.' THEN
  PREPARE Stmt FROM
    'SELECT
       ID,
       P.FullName,
       TD.DateEnd,
       TD.YY + ''/'' + TD.MM as YYMM,
       TD.GrossPay,
       TD.GrossBonus,
       TD.SSEE,
       TD.SSER,
       TD.PAYE,
       TD.BonusPAYE,
       ''TaxesDue'' as LinkTable
      FROM TaxesDue TD
      LEFT JOIN People P ON P.ID = TD.StaffID
      WHERE DateEnd = ?  ';
  OPEN Crsr USING aDateEnd;
  END IF;
END


DESCRIPTION
'[Properties]
Type=Entity
Title=End of Month: Generate "TaxesDue" Records
LinkTable=TaxesDue
UserMessage=Add Records to the TaxesDue data-table
CheckFirst=true
SecurityLevel=100
ImageIndex=265'
```

Procedure using "UserMessage" and Cursor with RETURN

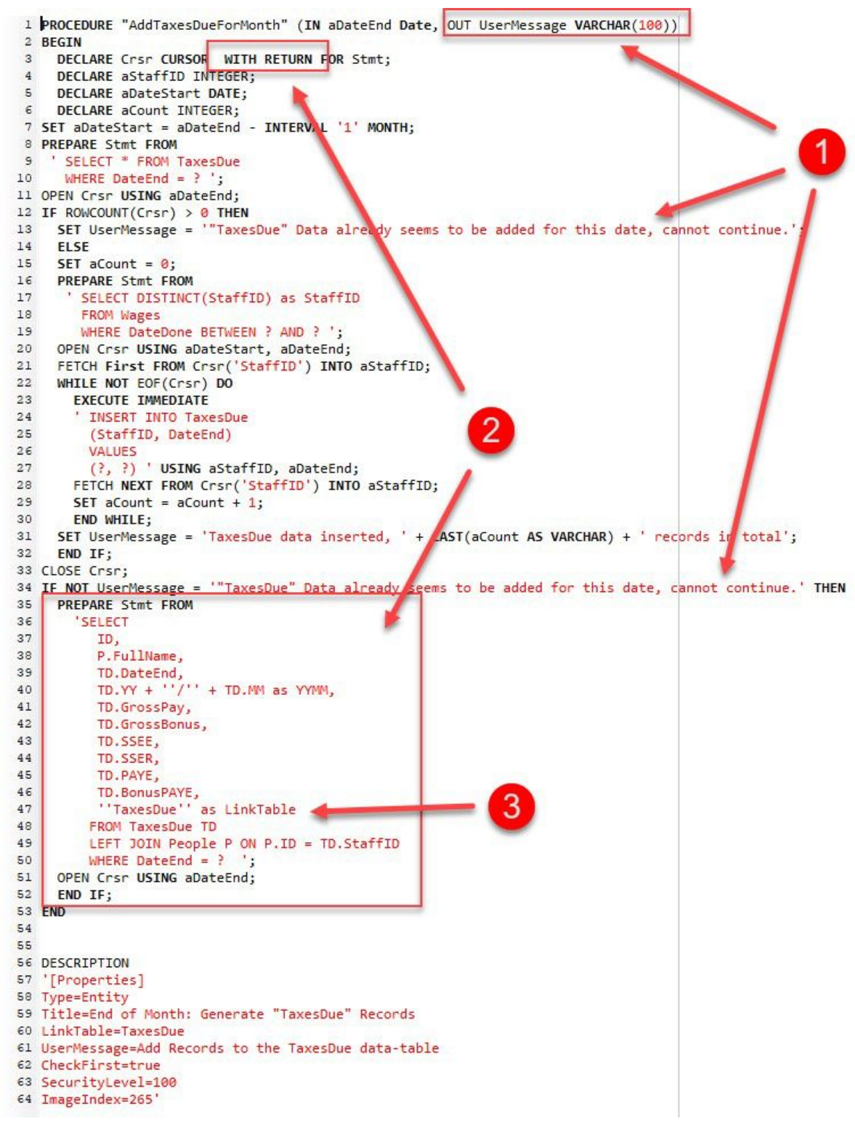
The above Stored-procedure (full SQL code is repeated below) uses an "OUT Parameter" of "UserMessage" and sets this message in the body of the Procedure.

It also DECLARES a CURSOR WITH RETURN, meaning that when the Statement at the end of the procedure runs and is opened, the user will see the resulting data.

What the above procedure does:

1. Check whether data has already been entered for the "DateEnd" submitted. If it has, the procedure sets the UserMessage to "... cannot continue" and stops.
2. If there is no data present for the given date, the procedure then inserts records into a "TaxesDue" data-table. In this data-table computations are done to return the value of tax owed by staff.
3. Once all the records are inserted for the month in question, a statement is run to allow the user to view all the inserted records.

Note: By adding the ID field and "LinkTable" field in the returned data, Orixa will be able to allow the user to access the individual TaxesDue records in the Orixa App, giving them a chance to update and edit these records.



```
PROCEDURE "AddTaxesDueForMonth" (IN aDateEnd Date, OUT UserMessage VARCHAR(100))
BEGIN
  DECLARE Crsr CURSOR  WITH RETURN FOR Stmt;
  DECLARE aStaffID INTEGER;
  DECLARE aDateStart DATE;
  DECLARE aCount INTEGER;
SET aDateStart = aDateEnd - INTERVAL '1' MONTH;
PREPARE Stmt FROM
 ' SELECT * FROM TaxesDue
   WHERE DateEnd = ? ';
OPEN Crsr USING aDateEnd;
IF ROWCOUNT(Crsr) > 0 THEN
  SET UserMessage = '"TaxesDue" Data already seems to be added for this date, cannot continue.';
  ELSE
  SET aCount = 0;
  PREPARE Stmt FROM
   ' SELECT DISTINCT(StaffID) as StaffID
     FROM Wages
     WHERE DateDone BETWEEN ? AND ? ';
  OPEN Crsr USING aDateStart, aDateEnd;
  FETCH First FROM Crsr('StaffID') INTO aStaffID;
  WHILE NOT EOF(Crsr) DO
    EXECUTE IMMEDIATE
    ' INSERT INTO TaxesDue
      (StaffID, DateEnd)
      VALUES
      (?, ?) ' USING aStaffID, aDateEnd;
    FETCH NEXT FROM Crsr('StaffID') INTO aStaffID;
    SET aCount = aCount + 1;
    END WHILE;
  SET UserMessage = 'TaxesDue data inserted, ' + CAST(aCount AS VARCHAR) + ' records in total';
  END IF;
CLOSE Crsr;
IF NOT UserMessage = '"TaxesDue" Data already seems to be added for this date, cannot continue.' THEN
  PREPARE Stmt FROM
   'SELECT
      ID,
      P.FullName,
      TD.DateEnd,
      TD.YY + ''/'' + TD.MM as YYMM,
      TD.GrossPay,
      TD.GrossBonus,
      TD.SSEE,
      TD.SSER,
      TD.PAYE,
      TD.BonusPAYE,
      ''TaxesDue'' as LinkTable
     FROM TaxesDue TD
     LEFT JOIN People P ON P.ID = TD.StaffID
     WHERE DateEnd = ?  ';
  OPEN Crsr USING aDateEnd;
  END IF;
END


DESCRIPTION
'[Properties]
Type=Entity
Title=End of Month: Generate "TaxesDue" Records
LinkTable=TaxesDue
UserMessage=Add Records to the TaxesDue data-table
CheckFirst=true
SecurityLevel=100
ImageIndex=265'
```

Annotated version of Procedure

1. "UserMessage" declared as an Output Parameter, and updated and used in the SQL script.
2. Use of the "WITH RETURN" keywords, and the statement that will be run at the end of the procedure's operation.
3. The user of the "LinkTable" field, allowing Orixa to contextualize the returned data and link it to editable records in the App.

Example SQL Script for the procedure

```
PROCEDURE "AddTaxesDueForMonth" (IN aDateEnd Date, OUT UserMessage VARCHAR(100))
BEGIN
DECLARE Crsr CURSOR WITH RETURN FOR Stmt;
DECLARE aStaffID INTEGER;
DECLARE aDateStart DATE;
DECLARE aCount INTEGER;
SET aDateStart = aDateEnd - INTERVAL '1' MONTH;
PREPARE Stmt FROM
' SELECT * FROM TaxesDue
WHERE DateEnd = ? ';
OPEN Crsr USING aDateEnd;
IF ROWCOUNT(Crsr) > 0 THEN
SET UserMessage = '"TaxesDue" Data already seems to be added for this date, cannot continue.';
ELSE
SET aCount = 0;
PREPARE Stmt FROM
' SELECT DISTINCT(StaffID) as StaffID
FROM Wages
```

```
WHERE DateDone BETWEEN ? AND ? ';
OPEN Crsr USING aDateStart, aDateEnd;
FETCH First FROM Crsr('StaffID') INTO aStaffID;
WHILE NOT EOF(Crsr) DO
EXECUTE IMMEDIATE
' INSERT INTO TaxesDue
(StaffID, DateEnd)
VALUES
(?, ?) ' USING aStaffID, aDateEnd;
FETCH NEXT FROM Crsr('StaffID') INTO aStaffID;
SET aCount = aCount + 1;
END WHILE;
SET UserMessage = 'TaxesDue data inserted, ' + CAST(aCount AS VARCHAR) + ' records in total';
END IF;
CLOSE Crsr;
IF NOT UserMessage = '"TaxesDue" Data already seems to be added for this date, cannot continue.'
THEN
PREPARE Stmt FROM
'SELECT
ID,
P.FullName,
TD.DateEnd,
TD.YY + ''/'' + TD.MM as YYMM,
TD.GrossPay,
TD.GrossBonus,
TD.SSEE,
TD.SSER,
TD.PAYE,
TD.BonusPAYE,
''TaxesDue'' as LinkTable
FROM TaxesDue TD
LEFT JOIN People P ON P.ID = TD.StaffID
WHERE DateEnd = ? ';
OPEN Crsr USING aDateEnd;
END IF;
END

DESCRIPTION
'[Properties]
Type=Entity
Title=End of Month: Generate "TaxesDue" Records
LinkTable=TaxesDue
UserMessage=Add Records to the TaxesDue data-table
CheckFirst=true
SecurityLevel=100
ImageIndex=265'
```