Some example BusinessObjects and explanations of parts of their scripts

> **Note that all Developers can open the "DB Utility" shown below and from this screen access ALL the structural details of every BusinessObject in their App. This allows Developers to "learn by doing" as they can review the contents of the BusinessObject and then open its records/grids/reports in the App and see how it works.**

Orixa Creation Script for a Simplified "Contracts" BusinessObject

> **Note that this script was generated automatically, by opening the "BusinessObject" record for the "Contracts" BusinessObject and selecting "Reverse Engineer BusinessObject" from the Actions menu.**

The SQL script to Create the "Contracts" BusinessObject

CREATE TABLE "Contracts"

```
(
"ID" INTEGER DEFAULT UID() NOT NULL,
"OID" INTEGER DEFAULT OID('Contracts'),
"DateStart" DATE,
"DateEnd" DATE,
"CustomersID" INTEGER,
"Name" VARCHAR(100) COLLATE "ANSI",
"StatusID" INTEGER DEFAULT StatusID('In negotiation'),
"ContractsTypeID" INTEGER,
"Internal" BOOLEAN DEFAULT FALSE NOT NULL,
"Description" CLOB COLLATE "ANSI",
"AuthorID" INTEGER,
"LeadStaffID" INTEGER,
"DateCreated" TIMESTAMP DEFAULT Current_Timestamp,
"Current" BOOLEAN DEFAULT true NOT NULL,
"Complete" BOOLEAN DEFAULT false NOT NULL,
"FullName" VARCHAR(120) COLLATE "ANSI" COMPUTED ALWAYS AS IF(COMPLETE = true then 'zzz/' ELSE '') +
IF(Internal = true then 'I' ELSE '') +
'C-' + CAST(OID AS VARCHAR) + ': ' + Name,
"ShortName" VARCHAR(20) COLLATE "ANSI" GENERATED ALWAYS AS IF(Internal = true then 'I' ELSE '')
+ 'C-' + CAST(OID AS VARCHAR(12)) + '-' + OrgCode(CustomersID),
"VATExempt" BOOLEAN GENERATED ALWAYS AS CustomerVATExempt(CustomersID),
CONSTRAINT "PK_Contracts" PRIMARY KEY ("ID"),
CONSTRAINT "LeadStaffID" FOREIGN KEY ("LeadStaffID") REFERENCES "Staff" ("ID")
  ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT "ContractsTypeID" FOREIGN KEY ("ContractsTypeID") REFERENCES "Types" ("ID")
  ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT "StatusID" FOREIGN KEY ("StatusID") REFERENCES "Status" ("ID")
  ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT "CustomersID" FOREIGN KEY ("CustomersID") REFERENCES "Customers" ("ID")
  ON UPDATE NO ACTION ON DELETE NO ACTION
DESCRIPTION 'Default'
)
DESCRIPTION '[Properties]
AddDuplicateRecord=1'
```

> **Note the field-names used and how some fields are generated by functions, or link to other tables via contraints.**

Functions Used by the BusinessObject

CREATE FUNCTION "StatusID" (IN "Value" VARCHAR COLLATE "ANSI")

```
RETURNS INTEGER
BEGIN
  DECLARE Crsr Cursor FOR Stmt;
  DECLARE Result INTEGER;

PREPARE Stmt FROM
' SELECT ID
  FROM Status
  WHERE UPPER(Name) = UPPER(?) ';
OPEN Crsr USING Value;
FETCH FIRST FROM Crsr('ID') INTO Result;
CLOSE Crsr;
RETURN Result;
END
VERSION 1.00!
```

> **The "StatusID" function is built into Framework, you can use it in your table definition to create a default value for the StatusID field.**

CREATE FUNCTION "OID" (IN "aLinkTable" VARCHAR COLLATE "ANSI")

```
RETURNS INTEGER
BEGIN
  DECLARE Crsr CURSOR FOR Stmt;
  DECLARE Result INTEGER;
  DECLARE ID VARCHAR;

  PREPARE Stmt FROM
    ' SELECT
        CAST(ID as VARCHAR) as ID,
        OID
      FROM "SystemDB"."OIDGenerator"
      WHERE LinkTable = ? ';
  OPEN Crsr USING aLinkTable;
FETCH FIRST FROM Crsr ('ID') INTO ID;
FETCH FIRST FROM Crsr ('OID') INTO Result;

EXECUTE IMMEDIATE
  ' UPDATE  "SystemDB"."OIDGenerator"
    SET OID = OID + 1
    WHERE ID = ' + ID ;
RETURN Result;
END
VERSION 1.00!
```

> **The "OID" Function is built into the Framework, you can use it in your table definition to create a unique counting-field for each table, so that one column has a value where each new record in the table is one higher than the previous record. Note that "OID" fields are dependent on a database NOT using distributed data-structure.**

CREATE FUNCTION "OrgCode" (IN "aID" INTEGER)

```
RETURNS VARCHAR(12) COLLATE "ANSI"
BEGIN
  DECLARE Crsr CURSOR FOR Stmt;
  DECLARE Result VARCHAR(12);
IF aID IS NULL THEN
  SET Result = '';
  ELSE
  PREPARE Stmt FROM
  ' SELECT
      OrgCode
    FROM Organisations
```

```
    WHERE ID = ? ';
  OPEN Crsr USING aID;
  FETCH FIRST FROM Crsr('OrgCode') INTO RESULT;
  IF RESULT IS NULL THEN
    SET RESULT = '';
    END IF;
  END IF;
RETURN Result;
END
VERSION 1.00!
```

CREATE FUNCTION "CustomerVATExempt" (IN "aID" INTEGER)

```
RETURNS BOOLEAN
BEGIN
  DECLARE Crsr CURSOR FOR Stmt;
  DECLARE Result BOOLEAN;
IF aID IS NULL THEN
  SET Result = false;
  ELSE
  PREPARE Stmt FROM
  ' SELECT
      VATExempt
    FROM Customers
    WHERE ID = ? ';
  OPEN Crsr USING aID;
  FETCH FIRST FROM Crsr('VATExempt') INTO RESULT;
  IF RESULT IS NULL THEN
    SET RESULT = false;
    END IF;
  END IF;
RETURN Result;
END
VERSION 1.00!
```

CREATE TRIGGER "CascadeLinkTableDeletes" BEFORE DELETE ON "Contracts"

```
 BEGIN
EXECUTE IMMEDIATE
' DELETE FROM Comments WHERE LinkID = ' + CAST(OLDROW.ID AS VARCHAR(20)) +
' AND LinkTable = ''Contracts'' ';
END
```

The BusinessObjects record for this BusinessObject

INSERT INTO "BusinessObjects"

```
(Name, Color, BusObjType,
AddNewButton, LinkToImages, LinkToComments, LinkToAddresses,
LinkToPhones, SecurityLevel, Icon,
DisplayScript, ListScript, ViewScript, DiaryScript,
SummaryScript, Settings)
VALUES
('Contracts', 15519677, 'N', True, False, True, False, False, 0, 7,
  ' SELECT
  IF(Complete = true THEN ''zzz\'' else '''') +
  IF(Internal = true THEN ''I'' ELSE '''')
  + ''C-'' + CAST(OID AS VARCHAR) + '': ''
  + O.Name + '' '' +
  + C.Name AS Name,
  ID
FROM "Contracts" C
LEFT JOIN Organisations O ON O.ID = C.CustomersID
WHERE ID = %d ', ' SELECT
```

```
  IF(Complete = true THEN ''zzz\'' else '''') +
  IF(Internal = true THEN ''I'' ELSE '''') +
  ''C-'' + CAST(OID AS VARCHAR) + '': ''
  + O.Name + '' '' +
  + C.Name as FullName, ID
FROM "Contracts" C
LEFT JOIN Organisations O ON O.ID = C.CustomersID
ORDER BY FullName ', 'SELECT
  ID,
  Internal,
  ShortName,
  Name,
  DateStart,
  DateEnd,
  S.FullName as "Status",
  T.Name as ContractsType,
  O.Name as Customer,
  SUM(CI1.WorkedHours) as HoursWorked,
  SUM(CI1.BudgetQuantity) as BudgetQty,
  SUM(CI1.BudgetValue) as BudgetValue,
  SUM(CI1.BilledQuantity) as BilledQty,
  SUM(CI1.BilledValue) as BilledValue,
  IF(SUM(CI1.BudgetValue) = 0 THEN IF((SUM(CI1.WorkedHours) / SUM(CI1.BudgetQuantity) > 1) THEN
''Hours Exceeded'' ELSE ''Within Hours'')
    ELSE IF(SUM(CI1.WorkedValue) / SUM(CI1.BudgetValue) > 1 THEN ''Over Budget'' ELSE ''Within
Budget'')) as Budget,
  IF(SUM(CI1.BudgetValue) = 0 THEN SUM(CI1.WorkedHours) / SUM(CI1.BudgetQuantity) * 100
    ELSE(SUM(CI1.WorkedValue) / SUM(CI1.BudgetValue)) * 100) as PercentUsed,
  MIN(TotalBilled) as TotalBilled, --multiple lines so just pick one, not the sum
  MIN(TotalPaid) as TotalPaid, --multiple lines so just pick one, not the sum
  P1.FullName as Staff,
  DateCreated,
  Current,
  Complete
FROM "Contracts" C
  LEFT JOIN Status S ON S.ID = C.StatusID
  LEFT JOIN Types T ON T.ID = C.ContractsTypeID
  LEFT JOIN People P1 ON P1.ID = C.LeadStaffID
  LEFT JOIN Organisations O ON O.ID = C.CustomersID
  LEFT JOIN
  ( SELECT
      CI.ContractsID,
      WI.ContractItemsID,
      CI.BudgetQuantity,
      CI.BudgetValue,
      CI.BilledQuantity,
      CI.BilledValue,
      CI.PaidValue,
      SUM(WI.HoursWorked) as WorkedHours,
      SUM(WI.Value) as WorkedValue
    FROM ContractItems CI
    LEFT JOIN WorkItems WI ON CI.ID = WI.ContractItemsID
    LEFT JOIN People P ON P.ID = WI.StaffID
    GROUP BY ContractItemsID) as CI1 ON CI1.ContractsID = C.ID
LEFT JOIN
      ( SELECT
        ContractsID,
        SUM(ValueBilled) as TotalBilled,
        SUM(ValuePaid) as TotalPaid
        FROM ContractPayments
        GROUP BY ContractsID ) as CP ON CP.ContractsID = C.ID
%s
GROUP BY C.ID',
```

```
  '', 'SELECT
  ContractsID as ID,
  SUM(WorkedHours) as WorkedHours,
  SUM(WorkedValue) as WorkedValue,
  SUM(BudgetQuantity) as BudgetQty,
  SUM(BudgetValue) as BudgetValue,
  SUM(BilledQuantity) as BilledQty,
  SUM(BilledValue) as BilledValue,
  IF(SUM(BudgetValue) = 0 THEN IF((SUM(WorkedHours) / SUM(BudgetQuantity) > 1) THEN ''Hours
Exceeded'' ELSE ''Within Hours'')
    ELSE IF(SUM(WorkedValue) / SUM(BudgetValue) > 1 THEN ''Over Budget'' ELSE ''Within Budget''))
as Budget,
  IF(SUM(BudgetValue) = 0 THEN ROUND(SUM(COALESCE(WorkedHours, 0)) / SUM(COALESCE(BudgetQuantity,
0)) * 100 TO 2)
    ELSE ROUND(SUM(COALESCE(WorkedValue, 0)) / SUM(COALESCE(BudgetValue, 0)) * 100 TO 2)) as
PercentUsed,
  COALESCE(MIN(TotalBilled), 0) as TotalBilled, --multiple lines so just pick one, not the sum

  COALESCE(MIN(TotalPaid), 0) as TotalPaid --multiple lines so just pick one, not the sum
FROM
(SELECT
  CI.ContractsID,
  CI.ID as ContractItemsID,
  CI.BudgetQuantity,
  CI.BudgetValue,
  CI.BilledQuantity,
  CI.BilledValue,
  CI.PaidValue,
  SUM(COALESCE(WI.HoursWorked, 0)) as WorkedHours,
  SUM(COALESCE(WI."Value", 0)) as WorkedValue
FROM ContractItems CI
LEFT OUTER JOIN WorkItems WI ON CI.ID = WI.ContractItemsID
WHERE CI.ContractsID = %d
GROUP BY ContractItemsID) as CI1
LEFT JOIN
      ( SELECT
        ContractsID,
        ROUND(SUM(COALESCE(ValueBilled, 0)) TO 2) as TotalBilled,
        ROUND(SUM(COALESCE(ValuePaid, 0)) TO 2) as TotalPaid
        FROM ContractPayments
        GROUP BY ContractsID ) as CP ON CP.ContractsID = CI1.ContractsID

GROUP BY ID', '' )
!
```

> **Note the use of the "%d" wildcard in various parts of the above scripts. This indicates where the Orixa App will insert an ID Field from the BusinessObject to generate a list.**

Some examples of "Searches" "Types" and "Status" records for this BusinessObject

INSERT INTO Searches

```
(LinkTable, Name, SQLStr)
VALUES
('Contracts', 'Not Complete', 'Where Complete = false')!
-- INSERT SEARCHES DATA RECORDS --
INSERT INTO Searches
(LinkTable, Name, SQLStr)
VALUES
('Contracts', 'All Records', '')!
```

INSERT INTO Types

```
(Name, LinkField, LinkTable)
```

```
VALUES
('One-off', 'ContractsTypeID', 'Contracts') !
INSERT INTO Types
(Name, LinkField, LinkTable)
VALUES
('Annual', 'ContractsTypeID', 'Contracts') !
INSERT INTO Types
(Name, LinkField, LinkTable)
VALUES
('Quarterly', 'ContractsTypeID', 'Contracts') !
```

INSERT INTO Status

```
(Name, LogicalOrder, LinkTable)
VALUES
('In negotiation', 1, 'Contracts') !
INSERT INTO Status
(Name, LogicalOrder, LinkTable)
VALUES
('Active', 2, 'Contracts') !
INSERT INTO Status
(Name, LogicalOrder, LinkTable)
VALUES
('Complete', 3, 'Contracts') !
```

Some examples of "Resources" for this BusinessObject

INSERT INTO Resources

```
( ID, "Name", LocationID, ComponentID,
  SQLStr, "Description", ObjectProperties,
SecurityLevel, LinkTable, TargetTable )
VALUES
(11375, 'ContractItemsDashboardGrid', TypeID('Program Resource'), TypeID('Grid'),
'SELECT
  W.ID,
  W.DateDone,
  P2.FullName as "Staff",
  W.WorkPlanned + '' '' + W.WorkDone,
  T.Name as ItemType,
  W.HoursWorked,
  IF(C.Chargeable = true THEN CAST(C.UnitValue * W.HoursWorked AS DECIMAL(19, 4))
    ELSE CAST(0.00 AS DECIMAL(19,4))) as "BillableValue",
  C.Chargeable,
  S.FullName as "Status",
  C1.DatePaid,
  P.Name as Product,
  P1.FullName as LeadStaff
FROM ContractItems C
    LEFT JOIN Products P ON (C.ProductsID = P.ID)
    LEFT JOIN Types T ON (T.ID = C.ContractItemsTypeID)
    LEFT JOIN People P1 ON (C.StaffResponsibleID = P1.ID)
    LEFT JOIN Status S ON (C.StatusID = S.ID)
    LEFT JOIN ContractPayments C1 ON (C.ContractPaymentsID = C1.ID)
    LEFT JOIN WorkItems W ON (C.ID = W.ContractItemsID)
    LEFT JOIN People P2 ON (W.StaffID = P2.ID)
WHERE C.ID = %d
',
'',
  '',
0, 'Contracts', 'ContractItems' )!
```

INSERT INTO Resources

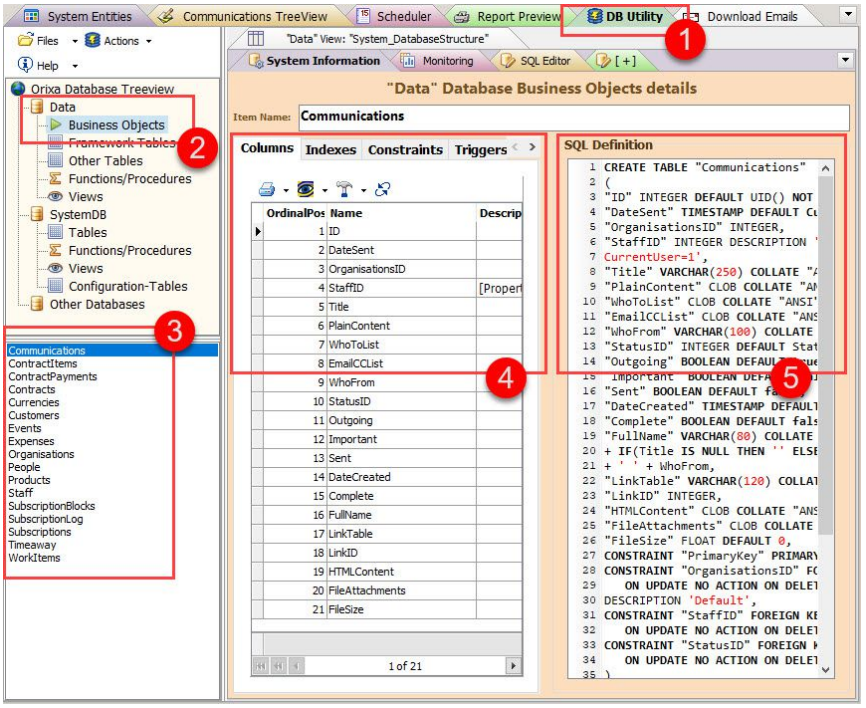
```
( ID, "Name", LocationID, ComponentID,
```

```
  SQLStr, "Description", ObjectProperties,
SecurityLevel, LinkTable, TargetTable )
VALUES
(38091, 'Contracts Completion Dashboard', TypeID('Record'), TypeID('Chart'),
'SELECT
  ContractItemsID as ID,
  SUM(CI.BudgetQuantity) as SumBudget,
  SUM(CI.BilledQuantity) as SumBilled,
  SUM(WI1.Hours) as SumWorked,
  SUM(WI2.Hours) as SumPlanned
FROM ContractItems CI
LEFT JOIN Types T ON T.ID = CI.ContractItemsTypeID
LEFT JOIN
  ( SELECT
    ContractItemsID,
    SUM(HoursWorked) as Hours
  FROM WorkItems
  WHERE DateDone < Current_Date
  GROUP BY ContractItemsID ) AS WI1 ON CI.ID = WI1.ContractItemsID
LEFT JOIN
  ( SELECT
    ContractItemsID,
    SUM(HoursWorked) as Hours
  FROM WorkItems
  WHERE DateDone < Current_Date
  GROUP BY ContractItemsID ) AS WI2 ON CI.ID = WI2.ContractItemsID
WHERE ContractsID = [BO Contracts]
GROUP BY ContractItems.ContractsID',
'',
'[General]
DateSaved=24/07/2020
SeriesCount=3
Title=Sum of Totals for Budget, Billing and Worked Hours

FooterText=
View3D=0
LegendVisible=1
AxisVisible=1

[Series0]
ArrowAxisFieldName=
XAxisFieldName=ID
YAxisFieldName=SumBudget
CalcOption=0
SeriesType=3
Title=SumBudget
Color=32768
ShowDataPoints=0

[Series1]
ArrowAxisFieldName=
XAxisFieldName=ID
YAxisFieldName=SumBilled
CalcOption=0
SeriesType=3
Title=SumBilled
Color=128
ShowDataPoints=0

[Series2]
ArrowAxisFieldName=
XAxisFieldName=ID
YAxisFieldName=SumWorked
```

```
CalcOption=0
SeriesType=3
Title=SumWorked
Color=8421376
ShowDataPoints=0

',
 0, 'Contracts', 'ContractItems' )!
```

> **Note that the Resources data-table data includes both SQL to return data from the database and Decoration data (created by the user in the App) to describe how the Resource dashboard will display.**

Accessing BusinessObject Details from the DB Utility



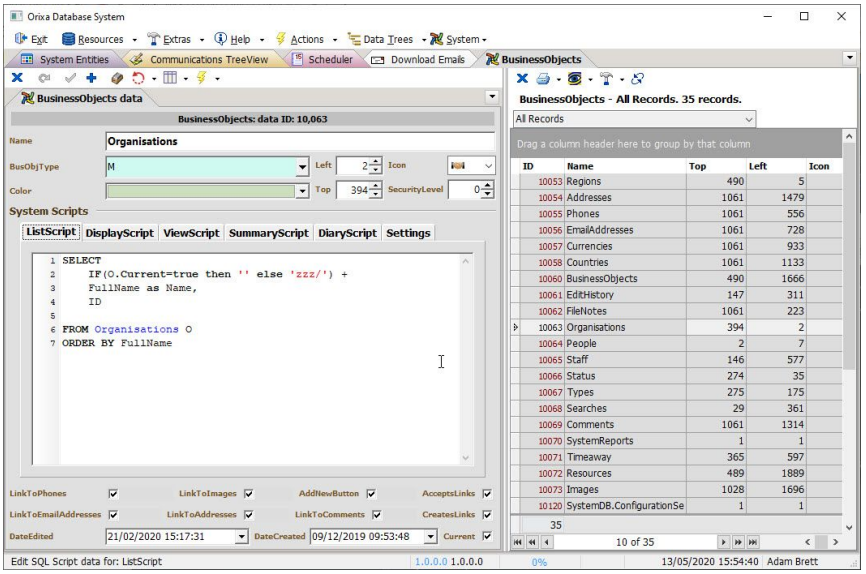
DB Utility Reviewing BusinessObjects

1. DB Utility Open in App, this can be accessed from the "System", "Show DB Management Utility" menu.
2. Select "BusinessObjects" from the Database Treeview.
3. All the BusinessObjects in your App will be listed.
4. The schema elements of the data-table are shown in detail.
5. The SQL Definition is shown in full. This includes the "Description" field, which may contain Orixa Decorations and generate in-App behaviour.

Accessing BusinessObject Details from the BusinessObjects Edit Form

Once a BusinessObject has been created as a Data-table, with the correct data-structure needed by your App, it is then made visible in-App by adding a record to the BusinessObjects data-table.

To understand how the different items in this table works it can be easiest to open an example App and review the contents of different BusinessObjects Records.



Business Objects data-grid

You can see from the above image that the BusinessObjects framework-table includes fields for the Icon, Name, BusinessObjectType & Color to be used in the App, plus SQL scripts to control how / whether details of the BusinessObject appears in different parts of the App.

Structure of the BusinessObjects Framework-table

```
CREATE TABLE "BusinessObjects"
( "ID" INTEGER DEFAULT UID() NOT NULL,
  "Name" VARCHAR(60) COLLATE "ANSI" NOT NULL,
  "Top" INTEGER DEFAULT 1 NOT NULL,
  "Left" INTEGER DEFAULT 1 NOT NULL,
  "Color" INTEGER
      DESCRIPTION '[Properties]
      DisplayControl=ColorCombo',
  "DisplayScript" CLOB COLLATE "ANSI",
  "ListScript" CLOB COLLATE "ANSI",
  "ViewScript" CLOB COLLATE "ANSI",
  "DiaryScript" CLOB COLLATE "ANSI",
  "SummaryScript" CLOB COLLATE "ANSI",
  "Settings" CLOB COLLATE "ANSI",
  "BusObjType" VARCHAR(2) COLLATE "ANSI" DEFAULT 'N' NOT NULL
       DESCRIPTION 'Distinct',
  "AddNewButton" BOOLEAN DEFAULT true NOT NULL,
  "LinkToImages" BOOLEAN DEFAULT false NOT NULL,
  "LinkToComments" BOOLEAN DEFAULT false NOT NULL,
  "LinkToAddresses" BOOLEAN DEFAULT false NOT NULL,
  "LinkToEmailAddresses" BOOLEAN DEFAULT false NOT NULL,
  "LinkToPhones" BOOLEAN DEFAULT false NOT NULL,
  "SecurityLevel" INTEGER DEFAULT 0 NOT NULL,
  "Icon" INTEGER,
  "DateCreated" TIMESTAMP DEFAULT Current_Timestamp,
  "DateEdited" TIMESTAMP DEFAULT Current_Timestamp,
  "Current" BOOLEAN DEFAULT true NOT NULL,
  "AcceptsLinks" BOOLEAN DEFAULT false NOT NULL,
  "CreatesLinks" BOOLEAN DEFAULT false NOT NULL,
CONSTRAINT "PK_ConfigurationSettings" PRIMARY KEY ("ID"),
CONSTRAINT "NameUnique" UNIQUE ("Name")
)
```