System Tables: Types and Statuss

The Orixa framework includes 2 framework tables, "Types" and "Status" which can used to create simple lists of values a user can pick from which can be linked to any BusinessObject.
The developer adds a number of records to each framework table, and users can then pick one of these values.
For example a table called "Farmers" might have a data-field "FarmersTypeID" linked to Types using the Orixa logic. Then Types records might be added with values such as "Cocoa Farmer" or "Tea Farmer" or any other required value.
These fields can be used to split data into categories.
The Status framework table has the additional capability that each record includes a "LogicalOrder" (an integer number) which allows the field to be used in "progress steps" in a business process.

NOTE Status and Types records are shown visually in standard data-edit form windows like all other Orixa BusinessObjects. The "base" data-tables have a fairly simple structure, including standard Orixa formatted entries in the framework tables. A Developer can extend these tables as much as they like, adding extra columns and features.

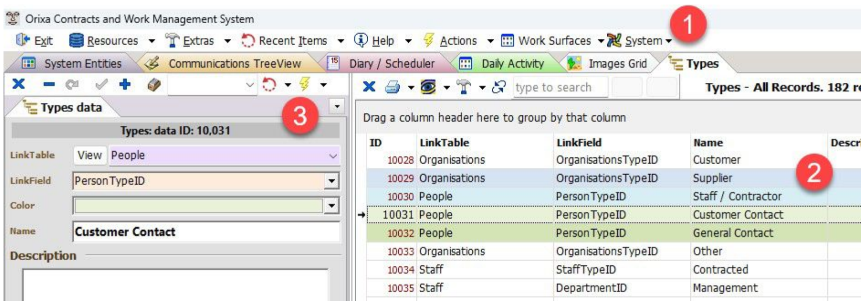
For general user-guide information review these Links:

The "Types" Edit Form

The "Status" Edit Form

Viewing and editing Types and Status records and lists

The global vaues for the Types and Status records are stored in a table, and can be updated by the developer at any time. This gives flexibility. If a Product Type starts as "Retail" but then needs to become "Retail/OnLine" only 1 update is needed, to the "Types" record, and**all** product records will update. However it is not without risk. A Types record can be changed by accident and this will immediately update the details of any record linked to this Type.



Types: data ID: 10,031

LinkTable	View	People
LinkField		PersonTypeID
Color		
Name		Customer Contact
Description		

ID	LinkTable	LinkField	Name	Descr
10028	Organisations	OrganisationsTypeID	Customer	
10029	Organisations	OrganisationsTypeID	Supplier	
10030	People	PersonTypeID	Staff / Contractor	
10031	People	PersonTypeID	Customer Contact	
10032	People	PersonTypeID	General Contact	
10033	Organisations	OrganisationsTypeID	Other	
10034	Staff	StaffTypeID	Contracted	
10035	Staff	DepartmentID	Management	

Types View Grid

To view / edit "Types"

1. Click "System", then "View / Edit System Types
2. Find a data record in the View Grid.
3. Double click to open and edit its value, or to add **new** types records.

Solving problems when bad or unnecessary Types or Status records are added to the system.

Incorrect "Types" and "Status" records can be added to the database, or a business may change and want to update the names it uses for Types or Status records.

If an incorrect record has been added and no **child** record has been linked to it, the bad record can simply be deleted and it will then disappear from the lists. However if any child record is linked to the incorrect Types record, it is impoosible to delete it due to**data-integrity controls**. If it needs to be deleted, all child records must first be updated.

If very few child records exist this can be done manually:

> 1. Search in the child-table for all records with type equal to the problematic value.
> 2. Update each one by opening the edit-form.

However, if a large number of records have been added this process is too slow.

In that case, it is better for the administrator to run a script to update the database to change all the records that use the incorrect type to a new value.

Example SQL for updating the Type record to a new value

A "Products" table includes a "ProductDevelopmentStageID" Types field. The Types table contains Records with ID 44356 and Name "New Product", this has been used but need to be updated by the new value: ID 56882 and Name "Product under development". To achieve this the following SQL would be used:

```
UPDATE Products
SET ProductDevelopmentStageID = 56882
WHERE ProductDevelopmentStageID = 44356
```

Note the basic form of this SQL, which is always the same:

```
UPDATE [Name-of-BusinessObject-Table]
SET [Name-of-TypesID-Data-field] = [old-value]
WHERE [Name-of-TypesID-Data-field] = [new-value]
```

If this kind of update script is needed regularly a SQL procedure can be drafted to make it easier to run regularly.

The data-structure of the Types Framework-table

```
CREATE TABLE "Types"
( "ID" INTEGER DEFAULT UID() NOT NULL,
  "LinkTable" VARCHAR(40) COLLATE "ANSI",
  "LinkField" VARCHAR(50) COLLATE "ANSI",
  "Name" VARCHAR(120) COLLATE "ANSI" NOT NULL,
  "Description" CLOB COLLATE "ANSI",
  "DateCreated" TIMESTAMP DEFAULT Current_TimeStamp,
  "Current" BOOLEAN DEFAULT true,
  "Color" INTEGER DESCRIPTION '[Properties]
  DisplayControl=ColorCombo',
  CONSTRAINT "PrimaryKey" PRIMARY KEY ("ID"),
  CONSTRAINT "LinkTable" FOREIGN KEY ("LinkTable") REFERENCES "BusinessObjects" ("Name")
  ON UPDATE NO ACTION ON DELETE NO ACTION )
)
```

> **Note that in-use the Orixa Framework collates the "LinkTable" and "LinkField" columns and uses them to create the "LinkRecord" entry that allows the user to link between the Comments / FileNotes record and the original record.**

The data-structure of the Status Framework-table

```
CREATE TABLE "Status"

( "ID" INTEGER DEFAULT UID() NOT NULL,
  "LinkTable" VARCHAR(40) COLLATE "ANSI",
  "LogicalOrder" INTEGER,
  "Name" VARCHAR(60) COLLATE "ANSI" NOT NULL,
  "Description" CLOB COLLATE "ANSI",
  "DateCreated" TIMESTAMP DEFAULT Current_TimeStamp,
  "Current" BOOLEAN DEFAULT true,
  "FullName" VARCHAR(80) COLLATE "ANSI" COMPUTED ALWAYS AS IF(LogicalOrder > 9 then '' else '0')
  + CAST(LogicalOrder as VARCHAR(2))
  + '. ' + Name,
  "Color" INTEGER DESCRIPTION '[Properties]
  DisplayControl=ColorCombo',
  CONSTRAINT "PrimaryKey" PRIMARY KEY ("ID"),
  CONSTRAINT "siLogicalOrder" UNIQUE ("ID", "LogicalOrder"),
  CONSTRAINT "LinkTable" FOREIGN KEY ("LinkTable") REFERENCES "BusinessObjects" ("Name")
  ON UPDATE NO ACTION ON DELETE NO ACTION
)
```