Customizing or creating your own WorkSurfaces

What are Worksurfaces for?

Please refer to the User Guide item [Worksurfaces, user interface for displaying data]

Creating Worksurfaces

Orixa will build linkages to Worksurfaces in your App if there are records in the**Resources** framework-table with Component = "Worksurface" or "Calendar Worksurface." then it will create items in user-menus that can be clicked to show a worksurface.

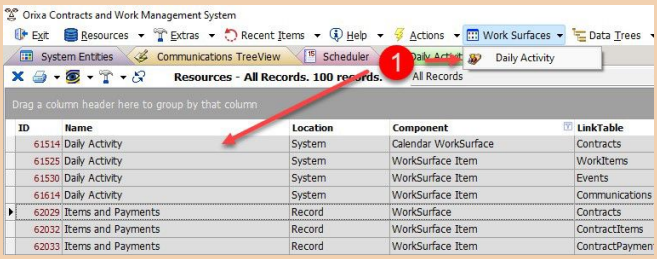
Please refer to the User Guide item [Using the "Resources" system-table to create reports, charts, data-cubes and dashboards]

for general notes on this process.

Worksurfaces are just another type of Orixa Resource, similar to charts, grids, Kanban boards and so on. Most of the values set when creating them are identical to those set for other types of resource.

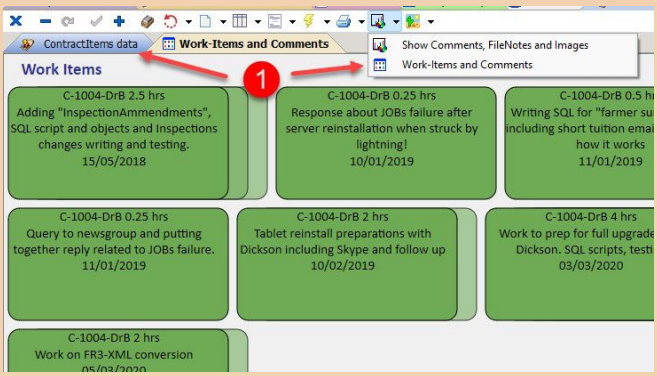
A key point of difference with Worksurfaces is that**multiple resources** are used to generate a single Worksurface, just as Orixa Dashboards can contain multiple elements each generated from separate SQL statements. A worksurface is formed from a master holder-record (with Component = "Worksurface" or "Calendar Worksurface") and child records (with Component = 'WorkSurface Item") each of which contain a single SQL statement that will run, and populate some elements of the worksurface.

Orixa uses the slightly clunky mechanism of grouping together Worksurfaces by Name, exactly as with Dashboards. All Resources with the same name that are Worksurfaces will be grouped together and run when the user clicks the appropriate menu item.



Resources Record generates System Work Surfaces Menu

If the "Location" of the Resource is set to "System" these Resources will be used to add a Work surfaces entry to the main App menu, and a menu item to this menu, as shown at 1., in the image.



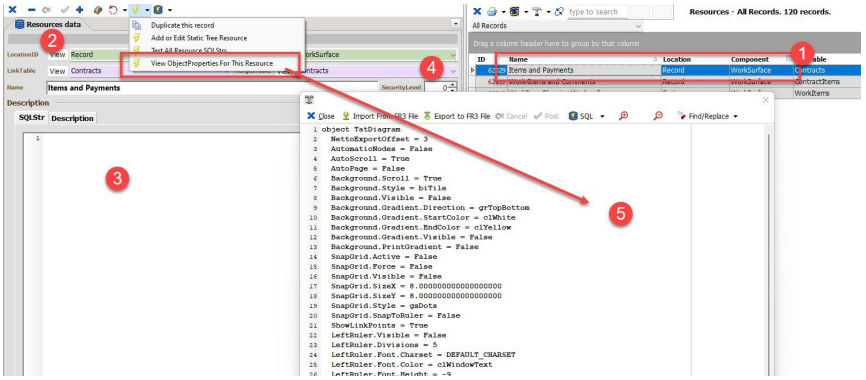
Resource with Location "Record" will show in the Record

If the "Location" of the worksurface is marked as "Record" then the Worksurface menu-item will appear as an extra item in the Business Object's Edit Form Dashboard menu as shown at 1., in the image.

If the "Location" of the workface is marked as "Entity" then an item will be added to the Dashboard menu on the System Entities the screen for the selected BusinessObject.

Formulating the SQL that creates a Worksurface

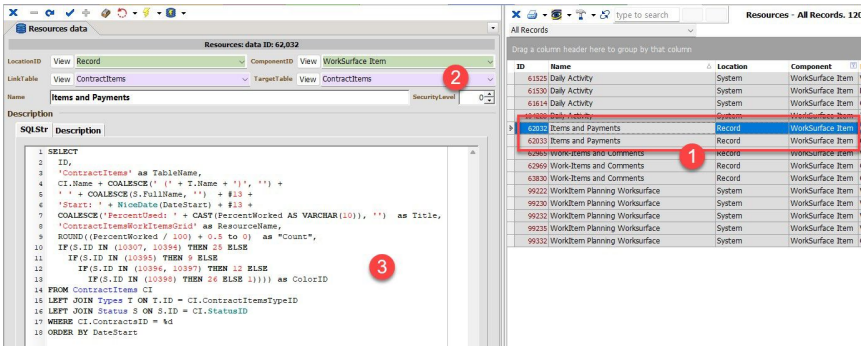
The "Master Record"



"Master" worksurface record

1. Row in the Resources data-table, with "Component" = "Worksurface". Or "Component" = "Calendar Worksurface" if you want to include **date navigation** in the Worksurface.
2. This master record has "Location" = "Record". This means that the Worksurface will refer to a **single record** when run. It will look for an Orixa ID and use this to populate the items it displays. "Location" can also be set to "Entity" or "System", if it is set to "Entity" a menu-item will be created on that Entity's "Reports / Dashboards" icon for the user to access it. "Entity" Worksurfaces do not make any assumptions about a particular Orixa ID when they open. As detailed above: If "Location" is set to "System", a "Worksurfaces\" menu will be added to the main-menu of your App, so the Worksurfcae can be accessed from there.
3. Note that the SQLStr is blank. No data is returned by the master record, it is simply used to hold the settings of the Worksurface.
4. It is linked to the Contracts BusinessObject.
5. The main use of the master-record of a worksurface is to hold the design details of the worksurface once it is developed. You do not have to manually write this code, it is generated based on the design you do in your App.

Individual Worksurface Resources records which return data



Worksurface Item Record(s)

1. Two records have the same name as the master. The SQL from these 2 records will be used to populate the worksurface.
2. The LinkTable of the resources are set to different "TargetTables", this means that when items are created on the worksurface, they will automatically link to the data-records of this BusinessObject, and use the ID field from the SQL data.
3. The Worksurface SQL. Examples of this are included below, but note the need for the following required **key fields**: "ID", "TableName" and "Title". All other fields are optional. The ID + TableName are used to return the Edit-Form of the BusinessObject when the user clicks a Worksurface Item. As well as required fields, the Worksurface also has **optional** fields: "Color", "Count", ResourceName" and "Hint", these are used to allow the Worksurface-item to show more dynamic data.

Data-fields the WorkSurface Item can consume

ID: As in most Orixa framework function, this identifies the displayed record

TableName: This identifies the name of the BusinessObject's data-table.

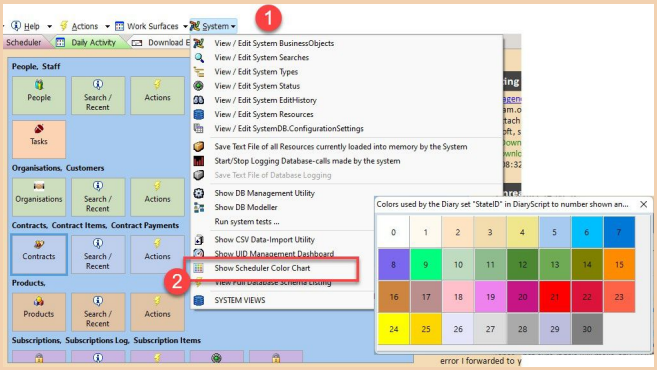
Title: The data returned in this column will actually be displayed for the user to see

ResourceName: (optional) If a Worksurface Item resource includes this static field, then an icon will appear on the Items in the worksurface, and when clicked your App will look for a chained resource record with this name. For more information about this, look at this help document:
[Using a "ResourceName" field to chain together Resources]

Count: (optional) Worksurfaces include the ability for items to be displayed as a "stack", giving the user visual indication of the number of items that are present. If a Count field is present Orixa will use this to build a "stack" of items. Take should be taken using this feature as very large stacks of items may look strange in the Worksurface.

Color or ColorID: (optional) Worksurface items are simple boxes filled with a colour. If a colorID field is present Orixa will use it to set the colour. The colour field can be drawn from the Color of the BusinessObject, or the Color of the Status or TypeID of individual records. Orixa also has some simple colours, set with numbers from 0 to 30. These can be accessed through the "Scheduler Color Chart", shown below.

Hint: (optional) If extra data is passed to the Worksurface item with this field name, the data will display when the user hovers over the item. This is useful to add optional details to an item, which do not need to be visible all the time.



Accessing the Scheduler Color Chart

The Scheduler Color Chart

Open this to show the form which details the number / colour correlation for basic colours in Orixa.

Where you can use a "Color" field in a SQL Script, these numbers **or** a "true" colour (LargeInteger) can be used.

1. From the System menu.
2. Click on "Show Scheduler Color Chart"

Example SQL for a WorkSurface Item which is linked to a Worksurface

```
SELECT
ID,
'ContractItems' as TableName,
CI.Name + COALESCE(' (' + T.Name + ')', '') +
' ' + COALESCE(S.FullName, '') + #13 +
'Start: ' + NiceDate(DateStart) + #13 +
COALESCE('PercentUsed: ' + CAST(PercentWorked AS VARCHAR(10)), '')
as Title,
'ContractItemsWorkItemsGrid' as ResourceName,
ROUND((PercentWorked / 100) + 0.5 to 0) as "Count",
IF(S.ID IN (10307, 10394) THEN 25 ELSE
IF(S.ID IN (10395) THEN 9 ELSE
IF(S.ID IN (10396, 10397) THEN 12 ELSE
IF(S.ID IN (10398) THEN 26 ELSE 1))))
as ColorID
FROM ContractItems CI
LEFT JOIN Types T ON T.ID = CI.ContractItemsTypeID
LEFT JOIN Status S ON S.ID = CI.StatusID
WHERE CI.ContractsID = %d
ORDER BY DateStart
```

Example SQL for a WorkSurface Item which is linked to a Calendar Worksurface

```
SELECT
ID,
'Communications' AS TableName,
COALESCE(C.Title + #13, '')
+ COALESCE('From: ' + CAST(C.WhoFrom as VARCHAR(50)), '')
AS Title,
S.Color
```

```
FROM Communications C
LEFT JOIN Status S ON S.ID = C.StatusID
WHERE CAST(C.DateCreated as DATE) = DATE '%0:s'
AND Outgoing = false
```

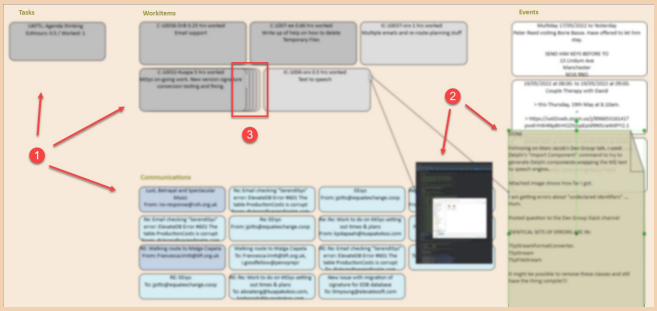
> **WorkSurface verses Calendar Worksurface:**
> **Orixa has to fill the worksurface with data when it is called on to do so.**
> **For a Worksurface, the execution causes Orixa to pass an ID into the Worksurface-Item scripts. If the Worksurface Location = Record then this will be the ID of the current record. If the Worksurface Location = Entity, Orixa will open a picking-list for the user to choose a record.**
> **For a Calendar Worksurface, the execution simply opens and builds the UI. The calendar-component (shown in images below) then allows the user to click on different dates to fill the Worksurface with records.**
> **The key difference for the Programmer, is that the SQL of a Worksurface will include a wild-card for the ID while the Calendar Worksurface will include a WHERE statement with wild-card(s) for a date.**

Customizing your Worksurfaces

When a Worksurface first displays very little will be visible. This is because, although records may have been returned, no design exists as yet.

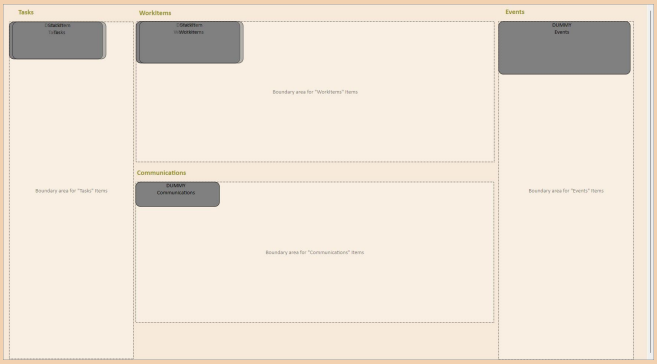
To create a layout or design for your Worksurface

Right click on the Worksurface and select "Start Editing Diagram" to design the layout of items, add text and other elements to a diagram. The saved design is stored in the ObjectProperties field of the Resources framework-table record that creates the Worksurface.



Worksurface in action

1. Items displaying with different colours on the worksurface.
2. Some Items include links to "Tasks" and "Images" these can be viewed by opening the "+" marked on the item.
3. Some Items include a "Count" indicating that there are a larger number of linked records, or that the linked record has a higher value.



Designing the layout for the Worksurface

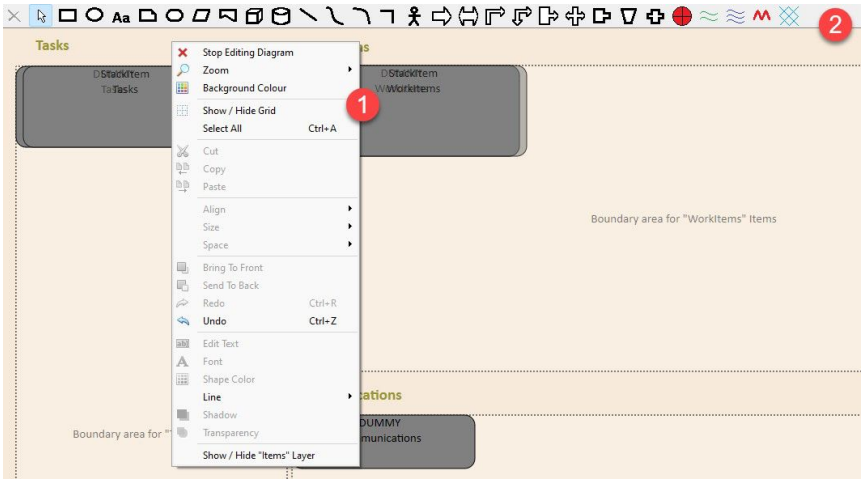
The very first time a new worksurface is shown in design mode, all the "Start-point" items and "fill-spaces" will simply be laid out across the empty worksurface, together with "Stack-items" for worksurface items that include a "count" field.

Right click on the worksurface and select "Start Designing Work Surface." The active worksurface will be replaced by a representation showing the "start-point" and "fill-space" for each of the Worksurface Items.

The Designer can:

1. Change the size, shape and position of each worksurface item. When data is returned all will be the same size, but will layout in the fill-space they have been allocated.
2. Add text headings to visually display the name of the data being shown.
3. Add Lines, textures and other features to the worksurface.
4. If the WorkSurface Item includes a "Count" field, the designer is free to position the "Stack-item" at an offset to the main item. This will determine how Items with a count greater than 1 apprear.

The Design-Worksurface Menu and Toolbar



Design-Worksurface Menu (1) and Toolbar (2)

1. **Worksurface menu:** Items include commands to Change the background colour. Cut, Copy and Paste Items.
2. **Worksurface Toolbar** Buttons on the surface of the toolbar allow the designer to add "furniture" such as text, boxes, lines curves and other visual components. Most of the items on this menu such as "Align", "Size", "Bring to Front" are commonly used in other similar software, and do not require detailed explanation here.

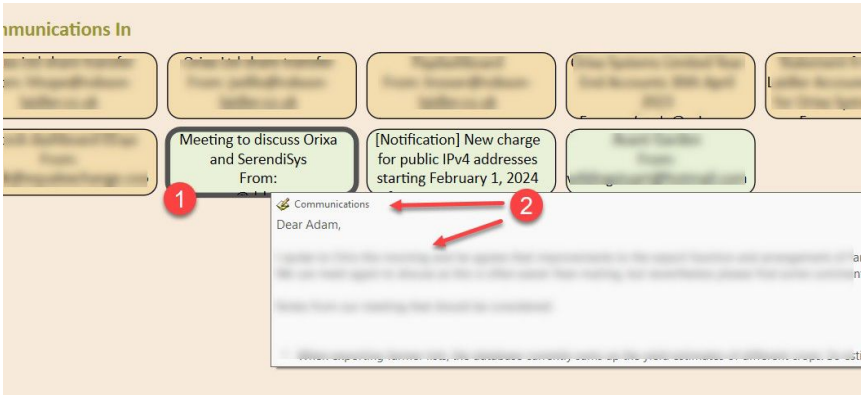
Worksurfaces are commonly used as visual surfaces for data such as Daily Activity, or to show a range of child-data items linked to a master record.

Orixa Worksurfaces can also be used to display layouts of factories and warehouses, so a set of items are included on the toolbar to allow graphic entries to represent parts of a factory. On the Toolbar you can see items intended to represent locations for water, electrical outlets, compresssed air, and a number of other simple shapes which can be added to represent machines or fittings.

Extending your Worksurfaces

Your Developer can add to or edit the Resources that create your Worksurfaces at any time. This will result in an update to how your Worksurfaces appear as soon as you restart your system. Note that if your design includes references to Worksurface items which are no longer present, this may generate non-fatal error messages. Just repeat the "Customizing your Worksurfaces" steps to rework the details of which items are displayed in which locations.

"Hint" feature of the Worksurface

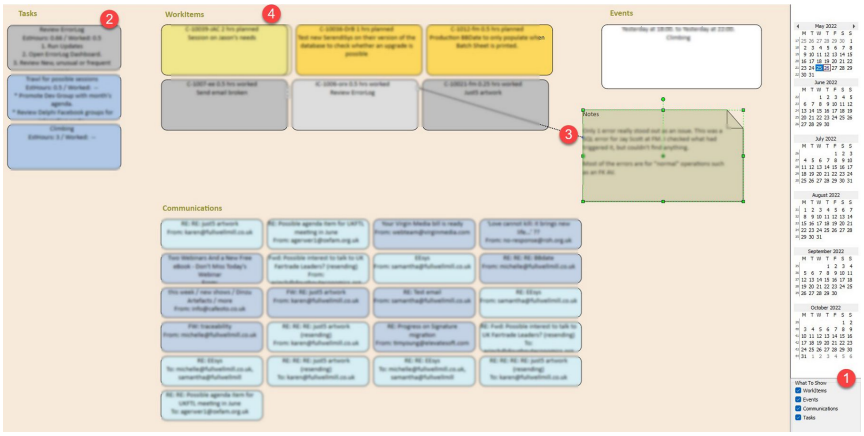


Hint showing in Worksurface

1. User holds the mouse over a Worksurface Item.
2. "Popup Hint" appears after 0.5 of a second. This includes the **tablename** for the BusinessObject, together with that BusinessObject's icon. Ay text data that is included in the "Hint" data appears below this. Note that multi-line hints are strongly supported, allowing quite large amounts of text to be shown.

The dataset returned by the Resource includes a field with the name "Hint", in such cases a multi-line pop up hint appears when the user hovers their mouse over any Worksurface Item. Note that if the Hint is empty or the dataset does not contain a "Hint" field, then the behaviour does not occur.

Worksurface Examples with SQL examples

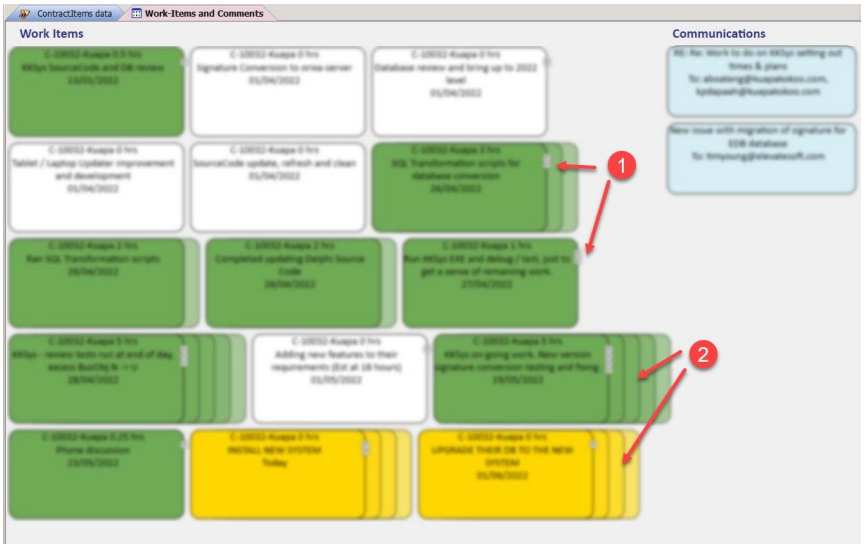


Calendar Worksurface showing daily-tasks, events, communications and work-items (work done)

1. By setting the Resources Component = "Calendar Worksurface" when created a Calendar component is added on the right-hand side of the Worksurface. Clicking on any date in this control will cause the Worksurface Items to run.
2. Items in this worksurface habe mostly been colour-coded according to their status (ie whether they are in need of work or not), with brighter colours indicating that work needs to be done. This effect is achieved by setting the colours of the Status records linked to the BusinessObject.
3. All Worksurfaces automatically create open-able links to the Comments, Images and File-Notes framework tables, so users can open these. These will also be colour-coded according to their status.
4. The "Count" field is used to "stack" work-items which have taken longer amounts of time.

Record Worksurface showing in-process and completed work

This worksurface is accessed from a "ContractItem" (a section of work) and enables users to quickly assess visually what work has been done, and still needs to be done. Note that the brightly coloured "gold" items are outstanding work which needs to be completed. It would be possible to add "warning" colour for overdue work.



Record Worksurface showing in-process and completed work in different colours

1. Note the presence of "+" symbols on the side of some items. THis indicates that "FileNotes" have been added to some records
2. Note that some items are "stacked" the optional **Count** field has been included in the dataset sent to the worksurface. If the Count is greater than 1, extra "stacked" items are added to the view, linked to that record. This helps give the user a visual indication of items which are "bigger" or "longer" than others.

SQL scripts of the Worksurface Item Resources for the above Worksurface

The first script returns the "Work-items" records shown on the left of the image, the second script returns the "Communications" on the right hand side. Note the use of the "%d" wild-card operator in each script.

The following SQL manually sets the "Color" based on a "StatusID" field. Note that any valid SQL can be used to set the Color, mapping to numbers between 0 and 60 for Orixa basic colours.

```
SELECT
WI.ID,
'WorkItems' as TableName,
IF(WI.StaffID = [CurrentUser] THEN '' ELSE P.FullName + ' ') +
C.ShortName + ' '
+ CAST(HoursWorked as VARCHAR(10)) + ' hrs ' + #13 + CAST(WI.Name as VARCHAR(200)) + #13 +
NiceDate(DateDone) as "Title",
IF(HoursWorked > 0 THEN ROUND(HoursWorked, 0) ELSE ROUND(HoursPlanned, 0)) as "Count",
CASE
WHEN StatusID = 62917 THEN 25 --Gold for "planned"
WHEN StatusID = 62918 THEN IF(CI.Chargeable = true THEN 13 ELSE 10) --stronger green for chargeable
work
WHEN StatusID = 62919 THEN 0
END
AS ColorID
FROM WorkItems WI
LEFT JOIN People P ON P.ID = WI.StaffID
LEFT JOIN ContractItems CI ON CI.ID = WI.ContractItemsID
LEFT JOIN Types T ON T.ID = CI.ContractItemsTypeID
LEFT JOIN Contracts C ON C.ID = CI.ContractsID
WHERE WI.ContractItemsID = %d
ORDER BY DateDone
```

The following SQL also sets the Color of the Worksurface Item, but it simply uses the Color of the **StatusID** of the record. Using this method allows color-continuity to be used across other parts of the Orixa App. Wherever you want a record with a particular status to appear in color, just use this feature.

```
SELECT
ID,
'Communications' AS TableName,
COALESCE(C.Title + #13, '')
+ COALESCE('To: ' + CAST(C.WhoToList as VARCHAR(50)), '')
AS Title,
S.Color
FROM Communications C
LEFT JOIN Status S ON S.ID = C.StatusID
WHERE LinkID IN
(SELECT
ID
FROM WorkItems
WHERE ContractItemsID = %d)
```

Worksurface showing work grouped by Date and Status

This is a "System" worksurface, accessed from the App's Main Menu. This means **no ID** is passed in when the SQL is run. Care should always be taken with a Worksurface of this type as it may return very large numbers of items.

Note how colour has been used in this case to indicate urgency of work. The red and pink items are over-due. The yellow items are at planning stage, and the different greens are used to indicate different status's of, "work-underway"

Worksurface showing work grouped by Date and Status

SQL Scripts to create this Worksurface

In this case each SQL statement is very similar. The only variation is the WHERE clauses, which fetch back different groups of items into each part of the Worksurface.The colouring of the items is not done using the Status of the records. Because multiple "Status" values exist (for Contract and Workitem) we want to set colours directly depending on a variety of conditions, including whether the Contract is "Internal" or not. For this reason the SQL uses Orixa's system-colour-numbers (0 to 30) to set these colours (as described above under "scheduler colour chart").

```
SELECT
ID,
'WorkItems' as TableName,
C.ShortName + #13 +
CI.Name + #13 +
Name + #13 +
'Hours: ' + CAST(HoursPlanned as VARCHAR)
as Title,
ROUND(HoursPlanned, 0) as "Count",
18 AS ColorID

FROM "WorkItems" W
LEFT JOIN ContractItems CI ON CI.ID = W.ContractItemsID
LEFT JOIN Contracts C ON C.ID = CI.ContractsID
WHERE W.StatusID = 62917
AND C.StatusID = 10292
AND DateDone <= Current_Date
AND CI.Chargeable = false
ORDER BY DateDone
```

```
SELECT
ID,
'WorkItems' as TableName,
C.ShortName + #13 +
CI.Name + #13 +
Name + #13 +
'Hours: ' + CAST(HoursPlanned as VARCHAR)
as Title,
ROUND(HoursPlanned, 0) as "Count",
21 AS ColorID

FROM "WorkItems" W
LEFT JOIN ContractItems CI ON CI.ID = W.ContractItemsID
LEFT JOIN Contracts C ON C.ID = CI.ContractsID
WHERE W.StatusID = 62917
AND C.StatusID = 10292
AND DateDone <= Current_Date
AND CI.Chargeable = true
ORDER BY DateDone
```

```
SELECT
```

```
ID,
'WorkItems' as TableName,
C.ShortName + #13 +
CI.Name + #13 +
Name + #13 +
'Hours: ' + CAST(HoursPlanned as VARCHAR)
as Title,
ROUND(HoursPlanned, 0) as "Count",
13 AS ColorID

FROM "WorkItems" W
LEFT JOIN ContractItems CI ON CI.ID = W.ContractItemsID
LEFT JOIN Contracts C ON C.ID = CI.ContractsID
WHERE W.StatusID = 62917
AND C.StatusID = 10292
AND DateDone > Current_Date
AND CI.Chargeable = true
ORDER BY DateDone
```

```
SELECT
ID,
'WorkItems' as TableName,
C.ShortName + #13 +
CI.Name + #13 +
Name + #13 +
'Hours: ' + CAST(HoursPlanned as VARCHAR)
as Title,
ROUND(HoursPlanned, 0) as "Count",
10 AS ColorID

FROM "WorkItems" W
LEFT JOIN ContractItems CI ON CI.ID = W.ContractItemsID
LEFT JOIN Contracts C ON C.ID = CI.ContractsID
WHERE W.StatusID = 62917
AND C.StatusID = 10292
AND DateDone > Current_Date
AND CI.Chargeable = false
ORDER BY DateDone
```

```
SELECT
ID,
'Contracts' as TableName,
C.ShortName + #13 +
O.Name + #13 +
C.Name + #13 +
'Budget Days: ' + COALESCE(CAST(ROUND(SUM(BudgetQuantity)/8, 0) as VARCHAR), '-nil-')
as Title,
ROUND(SUM(BudgetQuantity)/40, 0) as "Count",
IF(Internal = true THEN
IF(StatusID = 10291 THEN 2 ELSE 11)
ELSE
IF(StatusID = 10291 THEN 4 ELSE 12))
AS ColorID

FROM "Contracts" C
LEFT JOIN Organisations O ON O.ID = C.CustomersID
LEFT JOIN ContractItems CI ON CI.ContractsID = C.ID

WHERE C.StatusID IN (10291, 10292)
GROUP BY C.ID
ORDER BY DateStart Desc
```