# Chapter 10—Report Programming

You can create many different types of report using a template or wizard, each with very different layouts and data handling capabilities. With reports you can print out all or a subset of your data, and collect up data from different sources and print it on a single report. Each type of report in your application is defined as a *report class*. This chapter describes report classes and report sections and their properties.

Reports can contain data fields, pictures, text, and graphics. You can also place graphs on your reports, or base a report on an Omnis list. On the Omnis fat client, you can print reports to a number of destinations, including the current printer, the screen, a file, a port, the clipboard, or for the JavaScript Client you can print to a PDF file and display it in the end user’s web or mobile browser.

## Report Fields and Sections

You use report fields and sections to build all types of report. You can use standard *data fields* that can contain data from your server or Omnis database. You can use *picture fields* to display picture data. You place s*ections,* or horizontal dividers, across your report class that structure and position the data in the printed output. You can create subtotals, totals, header and footer sections for most types of reports. By setting the appropriate properties in a report class you can print labels as well. Furthermore you can add methods to a report class and the fields and section markers on the report.

The following fields or objects are available for report classes under the Report Fields tab in the Component Store:

1. **Entry field**  
   display the data from your database; you specify the variable or column name in $dataname
2. **Picture field**  
   displays picture or image from your database; you specify the variable or column name containing the picture data in $dataname
3. **Report Positioning Bar**special type of report section that allows you to position data fields and other objects in your report;
4. **External Components**various objects and components under the External Components tab in the Component Store

## Report Wizard

The SQL report wizard lets you create a report that contains fields that map directly to a SQL class in your library, which lets you print data on your SQL server database. Before you can use report wizards you must create the schema, table, or file classes necessary for SQL.

**To create a new report using a wizard**

1. Select your library in the Studio Browser
2. Click on the Class Wizard option, then click on the Report option
3. Select the report SQL Report wizard and click on the Create button

For SQL reports you need to select the Session to be associated with the new report. The SQL report wizard creates a report based on a schema or query class; each separate field on the new report maps to a schema column, which in turn maps to your server database

**Note:** there is an “Omnis Report Wizard” which creates a report based on an Omnis file class which is only suitable for storage in an Omnis datafile.

When you finish in the Report Wizard the new report class is opened ready for you to modify or print. To modify your report you need to edit its properties. You can also add new report objects from the Component Store, and you can add methods to the report class or the objects on the report.

## Report Tools

The toolbar at the top of the report editor lets you set the page size, preview the report on screen, and show or hide connections between the different sections of the report as shown down the left-hand side of the report editor. In addition, you set the sort levels in your report from this toolbar. Position your mouse over each tool to see what it does.

Some of the options in the report editor toolbar are available using the report context menu by Right-clicking on the report background.

The Narrow Sections option displays the section markers as narrow lines which shows you how the report will look when you print it. The Show $ident displays the ident numbers for the fields and section markers in the report. The Field List option displays a list of fields and section markers in the report class. You can expand the tree in the Field list to show the fields within each report section.

The Class Methods option lets you add methods to the report class, and the Properties option shows the properties for the class. The Page Setup option or toolbar button opens the Page Setup dialog in which you can select the printer, and set the page size and orientation. This dialog will vary greatly across the different operating systems.

### Zoom In/Out

The report class editor toolbar has Zoom In and Zoom Out buttons which control the DPI value used to convert report coordinates to and from pixels, and the DPI value used to create fonts used in the editor. "Zoom in" increases the DPI value, “Zoom out” decreases it. Note this is for design mode only, and you can zoom through a limited set of DPIs:

1. 72 – objects displayed at standard resolution for macOS
2. 96 – objects displayed at standard resolution for Windows (with default system scaling of 100%)
3. 144 – objects displayed at 2x resolution for macOS
4. 192 – objects displayed at 2x resolution for Windows

In addition, if Windows is using a different scaling value, the editor inserts the system DPI into this list at the appropriate point.

These values correspond to the design coordinate system used in Omnis, so on HD displays 96 DPI maps to 192 physical pixels.

You can use Ctrl+ and Ctrl- (Cmnd+/Cmnd- on macOs) to zoom in and zoom out respectively. The current zoom level is saved with the window setup by the save window setup context menu item.

Note that the section bars and the text in the left panel do not increase in height when you zoom. Note also, that zoom does not affect the size of lines drawn in fields on the report - only the text, and in some cases images will scale.

The Modify Report field has a new runtime property, $dpi, that can be assigned to one of the values above.

#### External Components

If you create external components for reports then you will need to make some changes in order to draw text at the correct DPI. Typically, if the component just displays its name or dataname using the standard interface, you won’t need to do anything, as the text DPI will be handled by the Omnis core. Where components that can be placed on reports draw custom text, there are some changes to make in the component:

1. There is a new callback ECOgetFontDpi(HWND) that returns the current DPI to use to create fonts - this will return zero unless the component is on a report design window, in which case it will return one of the above values.
2. There is a new class GDIfontCreator, that you construct with the HDC for drawing, and the return value from ECOgetFontDpi. This has a method createFont that you then use to create the font rather than calling GDIcreateFont. When you have finished with the font, call GDIdeleteObject as usual. You cannot cache the HFONT generated by createFont in your component.
3. If you require font or text metrics, use the HDC versions of GDIfontHeight, GDIfontPart, GDItextWidth etc, with a font created using GDIfontCreator.
4. In addition, for more advance use there are classes GDIhdcFontCacheHelper which removes all fonts cached by the Omnis font cache for a particular HDC at the end of the block and GDIoverrideHDCDPI which means that all fonts created for a specific HDC are created at a specified DPI while GDIoverrideHDCDPI is in scope. You need to use GDIoverrideHDCDPI if you are drawing styled text, as styled text drawing may create new fonts. In addition, when drawing styled text, you need to set mFontHdc in the GDIdrawTextStruct, in order for fonts to be created at the correct DPI.
5. You can also call GDIcreateDcFont with a DPI parameter to manage fonts yourself.

## Report Sections

Sections are horizontal markers or dividers across the report class that structure and position the data when your report is printed. To create a complex report with headers, footers, subtotals, and totals, such as an invoice or catalog listing, you have to place the appropriate sections in your report class in the right order. When you enable the various sections in your report using the Property Manager, their position and order is handled for you automatically.

There are two sections that you must have in a report: the *Record* section indicates the start of the display of records or rows of data, and the *End of report* section indicates the end of the report. These sections appear automatically in every new report class.

The following section types are available.

|  |  |
| --- | --- |
|  |  |
| Report heading | defines the area at the start of the report, which prints only once; you can use this to create a report title page |
| Page header | defines an area at the top of each page below the top margin, printed at the top of each new page |
| Subtotal heading | prints before each subtotals section; it would normally contain column headings for your subtotal sections |
| Subtotal heading 1 to 9 | each subtotal heading prints before its corresponding subtotal level |
| Record | defines the section containing the fields that print your data; the record section expands to accommodate your data which may extend over several pages when printed |
| Positioning | divides a section into two or more subsections; you can control exactly where on the page a positioning section is printed |
| Subtotals level 1 to 9 | defines the fields that will print subtotals; you can have up to 9 levels of subtotaling |
| Totals | prints at the end of the report and defines the fields that you want to total |
| Page footer | defines the area at the bottom of each page; printed at the bottom of each new page of your report |
| End of report | defines the end of the report; must be present on every report |

To enable a particular report section, set the appropriate property to true in the Property Manager. When you enable a particular report section, it is shown on your report in the correct position. You can click on individual sections to change their properties in the Property Manager.

### Page Headers and Footers

To create a page header for your report, you must set the **pageheader** property of the report to true using the Property Manager. The Page header section will appear on your report above the Record section, or any subtotal headings if you have any.

Similarly, to create a page footer for your report, you must set the **pagefooter** property to true in the Property Manager. The Page footer section will appear on your report below the Record section, or any Subtotals and Totals sections if you have any.

Any fields or graphics you place in the header section, that is between the Page header section and the next section marker will print at the top of each page. Likewise, any fields or graphics you place in the footer section, that is, below the Page footer section marker, will print at the bottom of each page. Note that the connection between the different sections is shown in the left margin of the report editor: the current section is shown in red.

When printing to a non-paged device such as File or HTML, by default the footer section is not printed. The Report header and first Page header sections are printed at the beginning of the report. However it is possible to force the footer section to be printed by calling $printsection( kFooter ) for the report instance. The default positioning for a footer for a non-paged device is to follow on from where the last section stopped printing.

To change the height of any section, including the record, header and footer sections, you can click on the section marker (the gray bar) and drag it into position. All the sections below the one you move will adjust automatically.

To show you more how the report will look when you print it, you can view the sections as narrow lines. To view sections as narrow lines, click on the Narrow sections button in the report editor toolbar, or Right-click on the report background and select the Narrow Sections option.

### Printing Sections as Record Sections

The $printsection method has a new parameter to force the report section to print as a record section. The method $printsection(iSection[,bPosnIsRecord=kFalse]) prints a section: note bPosnIsRecord applies to positioning sections only. If bPosnIsRecord is kFalse (the default), this method prints a section based on the position of the previous section; otherwise, when this parameter is true the method prints the section as a record.

### Borders and shading

You can set the border and fill properties for each section in a report. The report class and section objects have the following new properties: $effect, $forecolor, $backcolor, $bordercolor, $backpattern, $linestyle. In addition, sections have $topmargin, $leftmargin, $bottommargin and $rightmargin properties to allow the sections border and fill to be inset from the sections boundary.

Positioning sections cannot have their own border and fill properties. The border and fill of the main section extends through all its position sections. The border and fill properties of each section are visible in design mode.

### Watermarks

You can add a watermark effect (using a background image) to your reports by assigning a picture to the $backpicture property of the report. The alignment can be one of the following within the report margin: TopLeft, TopCenter, TopRight, CenterLeft, Center, CenterRight, BottomLeft, BottomCenter, BottomRight, Stretch (stretch the image to fit the report margin), or Tiles (tile the image within the report margins).

The horizontal and vertical DPI (dots per inch) specify the resolution at which the image is to be printed.

When using notation, $backpicture refers to the binary representation of the image and the formatting properties. To change the image or formatting properties you can use $backpicture.$picture or $backpicture.$align. The full set of notation is as follows:

|  |  |
| --- | --- |
|  |  |
| $backpicture.$picture | the actual image (24bit color shared). |
| $backpicture.$picturealign | alignment of the image within the report margins. This can be one of the kPALxxx constants. |
| $backpicture.$horzdpi | the horizontal dpi (defaults to 150), disabled for kAlignMargins and kAlignPrintable |
| $backpicture.$vertdpi | the vertical dpi (defaults to 150) , disabled for kAlignMargins and kAlignPrintable. |
| $backpicture.$horzoffset | additional horizontal offset in cms or inches from the alignment. Disabled for alignments of kAlignMargins and kAlignPrintable |
| $backpicture.$vertoffset | additional vertical offset in cms or inches from the alignment. Disabled for alignments of kAlignMargins and kAlignPrintable |

#### Icon IDs for background pictures

You can use the $backiconid property to assign an image to the background of a report using an image or icon ID, rather than using $backpicture. The property must refer to an icon ID in an icon set, or an alpha page in #ICONS in your library or an icon data file. If you specify a value for $backiconid it takes precedence over $backpicture.

## Section Positioning

When you print a report, each section follows the previous section by default, and is positioned down the page according to the height of the previous section set in the report class. However for some types of section, you can control where a section prints and whether or not a new page is forced using the **pagemode** and **startmode** properties of the section. You can use a special type of section marker called a *position* section to print part of your report literally anywhere on the page. To do all these things you have to modify the properties of the appropriate section marker.

To view the properties of a section, open your report class in design mode and click on the appropriate section to view its properties in the Property Manager.

### Page Mode

You can control whether or not *Record, Subtotals, Totals*, and *Subtotal heading* sections start a new page when they are encountered by setting their **pagemode** property. You can select one of the following options for this property.

1. **Nopage**  
   does not force a new page; uses the pagination in the report class (the default)
2. **Newpage**  
   always starts a new page before this section
3. **Testspace**starts a new page before starting this section if there is not the specified amount of space available on the current page

If you select the Testspace option, the **pagespacing** property is enabled in the Property Manager in which you can enter the amount of space required for the section. If this amount of space is not available on the page, a new page is started. The figure you enter in pagespacing is shown on the section marker.

Omnis works with units that are 1/72 of an inch; therefore it may round exact numbers in centimeters or inches to the next real unit. For example, 1cm becomes 0.99cm.

### Start Mode

All sections except for *Page footer* and *End of report* let you specify the **startmode**, which tells Omnis where to start the section. You can choose one of the following options.

1. **Follow** previous section  
   starts the section on the line following the end of the previous section (the default)
2. **Fromtop** of previous section  
   starts the section n inches/cms from the *top* of the previous section
3. **Fromend** of previous section  
   starts the section n inches/cms from the *end* of the previous section
4. **Fromtopmarg**  
   starts the section n inches/cms from the *top margin* of the report
5. **Frombottommarg**  
   starts the section n inches/cms from the *bottom margin* of the report

When you choose one of the start modes the **startspacing** property is enabled in the Property Manager, which lets you enter a measurement for the startmode. The startmode and spacing is shown on the section marker.

Omnis ignores previous section settings if the previous section was a *Page header* section or a *Positioning* section within a *Page header* section. The spacing comes before the page start test that examines the amount of space left on a page. Omnis ignores top and bottom margin settings for reports that are not paged.

Note that when you set up a report to print labels, you can use the Fromtop or Frombottom options to set the spacing between your labels.

You can enter a negative value for the start spacing of a positioning section, for example Start –1.000cms from end of previous section. This allows you to align fields with the bottom of an extending field.

### Record Spacing

The default spacing between records or rows of data on your printed report is determined by the height of the *Record* section in your report class. However you can override this spacing by setting the **userecspacing** property for the Record section. This property forces the report to use the vertical spacing set in the **recordspacing** property of the report class.

### Position Sections

A *Position* section is a special type of report section that you can print literally anywhere in the report. For example, using a positioning section, you could print a footnote or a logo at the bottom of a letter, regardless of the content or amount of data in the main report letter.

A positioning section placed over the second line of a two-line extending field with the Follow previous section property prevents the second line from printing as a blank. You can also follow extending fields by a positioning section with Follow previous section to prevent them from writing over any fields below. A positioning section within a subtotal section lets you trigger a print position change by changing a sort field value.

### Section Print Height

Report sections have the property $printheight which lets you specify the printing height of a section more accurately than using positioning sections. The printing height of a section is displayed in the section bar. Setting the sections print height does not affect the positioning of sections in the report class or on the design window. If a section area is larger than its print height, the area below the print height is filled with gray. Any report objects that fall entirely in the gray sections ($top of object is greater than $printheight of section) are evaluated but not printed.

You can create label reports using a combination of the $printheight property and the positioning mode kFitOnPage. If a section is too high to print on the current page, it will be printed the next page. Some objects are normally moved to fit on a page at the time they are added to the report, e.g. a single line of text. This behavior is disabled for objects in sections that have kFitOnPage selected.

Standard Lines between sections can expect an overall vertical placement inaccuracy of +-1/72nd of an inch. Hair lines can expect an overall placement inaccuracy equal to that of the DPI of the printer, but not better than +-1/600th of an inch.

### Section Calculations

Sections have the $printif property which can trigger the section to print based on a calculation. If a section has a calculation, it is skipped if the calculation evaluates to zero, otherwise the section is printed as normal. If a section has been given a calculation, the text “Print if calculation” will appear in the section bar.

If a main section is skipped, all positioning sections belonging to the main section are also skipped, regardless of their $printif property. Calling $printsection from a method will ignore the $printif calculation and always print the section.

### Custom Sections

A custom section is effectively a positioning section with $printif calculation of ‘0’. This means the section will not print unless, $printsection() is called with a reference to the section. You can add a custom section from the Component Store.

Custom sections can be placed inside any of the main sections, but can be printed before or after any of the other report sections, but not during. Changing the $printif calculation to anything other than ‘0’, will change the section into a normal positioning section, and vice versa.

### Hiding Sections in design mode

You can collapse or expand the sections in a report to simplify the visual appearance of the report in design mode. The collapse/expand icon in the report editor lets you collapse a section, leaving the section divider visible. The current collapse/expand state of a section will be saved in the report class.

When a ‘main section’ is collapsed, any positioning sections in that section are also hidden. Positioning sections can also be collapsed or expanded individually. Collapsing of sections is disabled when narrow sections are viewed.

## Sorting and Subtotaling

To implement sorting and subtotaling for your report, you need to specify the fields on your report to be sorted, and create subtotals sections containing those fields. *Sort fields* define how Omnis subtotals the records or rows of data when printing a report. With no sort fields, Omnis displays records in the order they are listed on your server, or if the data is from an Omnis data file, in the order the data was inserted. When you add sort fields to your report, the report will print subtotals when the values change in the sort fields.

To specify sort fields for your report, click on the Sort Fields button in the report editor toolbar; the Sort fields window opens. In the Sort fields window you can specify up to nine sort fields by entering each field or variable name in the left-hand column. Note that you can use the Catalog to enter your field or variable names. These sort fields form a nested sequence of sorts on the records that trigger printing of up to nine nested subtotal sections.

The sort fields for a report class are stored in the $sorts group for the class. You can modify the contents of this group at runtime using the notation, to change the sort fields for the report, but if you want the changes to take effect this must be done in the $construct() method of the report before the report instance is created. If you have more than one instance of a report class, each instance will have the sort fields specified in the class, but you can modify the $sorts group for a particular instance if you wish to change its sort fields.

To subtotal a field, place a copy of the field in the Subtotal section and select the appropriate **totalmode** for the field. This is independent of the sort fields, which trigger the printing of the Subtotal sections.

When you enter a field or variable name in the list of sort fields the sorting options are enabled for that field. You can enable any of these options by clicking on the cell and selecting true.

Each sort field has the following options.

1. **Descending** sort   
   sorts the field in descending (Z to A and 9 to 0) instead of the default ascending order
2. **Upper case** conversion   
   converts field values to upper case before sorting, so the use of mixed case in your database does not affect subtotaling or sorting
3. **Subtotals** when field changes   
   tells Omnis to print subtotals using the corresponding subtotal section (1 to 9) when the value of the field changes; that is, if sort field 4 changes, subtotal level 4 will print
4. **New page** when field changes   
   starts a new page as well as printing a subtotal when the value of the field changes

When you enable the Subtotals or New page options for a sort field, you can specify the number of characters that must change before a subtotal is triggered or new page is printed.

### Subtotal Sections

You can specify a *Subtotal heading* section in your report. It prints before the first Record section and successive Record sections following each Subtotals section. The subtotal heading can print column names and anything else you want to apply to each subtotaled Record section.

The *Subtotals* section prints whenever the Record section breaks on the corresponding sort field, with the subtotal printing before the record with the changed value. Since there are up to nine sort fields, you can have up to nine *Subtotal heading* and *Subtotal levels* numbered 1 through 9 corresponding to the sort fields specified in the report. The higher numbered sort fields are nested within the lower ones and hence change more often. That is, sort field 5 changes within sort field 4 which changes within sort field 3, and so on. Correspondingly, the Subtotal heading and Subtotals sections with higher numbers print more often as the sort fields change.

When you have multiple subtotals which print consecutively, the corresponding heading sections print one after another, starting with the one for the last subtotal. Subtotals and totals can be aggregations of several kinds, including sums, averages, counts, minimums, or maximums, depending on the field’s **$totalmode** property. Omnis maintains the total for each subtotal printing, then resets the subtotal to zero for the next section.

The *Totals* section prints at the end of the report. As for subtotals, you place the fields to aggregate in this section, and Omnis accumulates the aggregate values across the entire report. You can set the **totalmode** property for a field in the totals section.

### Calculated Fields

In addition to totaled fields in subtotal or total sections, you can place calculated fields in report sections which take their value from the result of some calculation. In this case the $dataname property of the field is left blank, the $calculated property is enabled and the calculation is placed in the $text property. For example, this could be the concatenation of a Firstname and Surname variable to display a person’s full name. Note that such calculated fields cannot be totaled in subtotal or total sections.

## Printing Reports

Omnis provides a wide range of choices for printing your reports, including sending your report to the printer, the screen, to a page preview, to a text or HTML file. End users can set the report destination using the File>>Print Destination menu option. In your finished library, you can provide a menu, popup menu, or toolbar button to print your report to the required destination. There are a number of commands that let you set the print destination, including *Send to screen, Send to file,* and *Send to clipboard*.

While you are creating your report class you may need to print it to try it out. You can use one of the buttons on the report editor toolbar to print the current report class. From these tools you can print to the current printer, a page preview window, or to the screen.

### Report Destination Dialog

You can select the output destination or device for your reports from the Print Destination dialog, available from the File>>Print Destination option on the main Omnis menu bar. This dialog may also contain any custom devices, such as the HTML device. The devices in the Print Destination dialog include

1. **Printer**  
   the current printer
2. **Preview**  
   reports are sent to a report preview window in Omnis
3. **Screen**  
   reports are sent to a report window in Omnis; the default destination
4. **Disk**  
   the report is sent to the file specified in the Parameters pane of the Print Destination dialog; the file is stored in a cross-platform proprietary binary format
5. **Clipboard**  
   reports are sent to the clipboard in text format
6. **Port**  
   the report is sent to the port specified in the Parameters pane
7. **File**  
   the report is sent to a text file specified in the Parameters pane
8. **RTF**  
   the report is sent to an RTF file specified in the Parameters pane
9. **HTML**  
   the report is sent to an HTML file (UTF-8 encoded)
10. **PDF**  
    the report is sent to a PDF file; for the JavaScript Client the file can be viewed in the browser on the end user’s mobile device, or for the fat client the file is opened on the end user’s computer using the default PDF viewer
11. **Memory** and **DDE/Publisher**   
    are also available but by default are not visible in the Print Destination dialog (you can set their $visible device preference to make them visible; see below)

To set the report destination

1. Select File>>Print Destination from the main Omnis menubar
2. Select a report destination and click OK, or double-click on a destination

### Printer Report Destination

The Printer option sends the report to the current printer. Under Windows, selecting the printer as destination opens a list of installed printers, and changing to a new printer does not affect the default printer setup as defined in the Windows Control Panel. Note that you can change the page setup with the File>>Page Setup menu item.

### Page Preview Destination

The Page Preview option displays a full page on the screen. Text is “Greeked” if the screen size is too small, dots representing the characters so that the whole page fits the available screen area.

Hyperlinks are supported in Page Previews: see PDF for details.

### Screen Report Destination

The Screen option is the default report destination at startup. It displays the report on the screen without the normal page margins. You can have more than one screen/page preview report open at a time. When you display reports on the screen, you can use the horizontal and vertical scroll bars to view the report, and the Page up, Page down, and arrow keys as well.

Omnis displays a screen report window as soon as it has prepared the first page of data, that is, normally it does not wait for the report to finish. Therefore as you scroll a long report it may take a few seconds to print each page.

#### Screen and Preview toolbars

Screen and Preview reports have a toolbar at the top of the window to allow end users to print the report to the current Printer or Save the report to a file.

The Omnis root preferences $reporttoolbarscreen and $reporttoolbarpagepreview (in the $root.$prefs devices group) allow you to select the buttons on the Screen report and Preview toolbar for user reports.

#### Copying from Screen and Preview reports

You can copy graphics in Windows metafile, bitmap or Mac PICT format from the screen and page preview report window by selecting an area with the mouse and using the Edit>>Copy menu item. You can copy text in the same way, and you can select and copy more of the report than is displayed on screen. When you drag-and-select data in your report, the window will scroll. The Select All option in the edit menu selects the whole report.

The Omnis root preference $disablereportcopy (in the $root.$prefs devices group) allows you to disable the copy via selection feature of screen reports and page previews for user reports.

#### Copy from Print Preview

You can use the tab key to tab through the page list, page and search field in the Print Preview screen. Therefore, to copy content from the preview screen, you can tab to the page if necessary, press Cmd+A to Select all and then Cmd+C to copy content.

When the page has the focus, you can use the Escape key to clear the selection.

#### Save PDF on Print Preview

The Page Preview window has a Save PDF button. You can control whether this button is present for user reports, using the root device preference $reporttoolbarpagepreview: the constant kRBsavePDF controls whether or not the button is present.

### Disk Report Destination

The Disk file report device sends the report output to a file on disk in a cross-platform binary format. If you double-click on the Disk icon in the Report Destination dialog Omnis prompts you for a disk file name.

You can print to the Disk device on one platform, reload the file in Omnis and print it on another platform. Alternatively, you can print the output from the Disk device using the File>>Print Report From Disk menu option, or using the *Print report from disk* command.

### Clipboard Report Destination

The Clipboard option sends the current report to the clipboard as an unpaged, text-only report suitable for pasting as text into other applications.

### Port Destination

The Port option sends the current report to a Unix or Windows serial or parallel port, or a Mac Modem or Printer port specified in the Parameters pane. This device also uses the settings in the Page sizes pane: see the File device.

Only one program can have a particular port open; if a port is open in Omnis and it is also, for example, the port used by the Spooler, then the Spooler will not be able to function.

Under macOS, there is an option on the Parameters pane to Convert for Imagewriter. When selected, the characters beyond ASCII 127 convert to a combination of a character, backspace, and accent character so that the report can print accented characters and umlauts.

Modern Mac computers do not have serial ports, but you can plug a serial adaptor into a USB port. Once the drivers for your serial adapter are installed correctly, Omnis Studio will automatically recognize any new serial ports when you plug the adapter into your machine. Any new serial ports that are discovered will be displayed in the parameters panel for you to select.

NOTE: Serial adapters and other USB devices which do not support the Comms Toolbox standard will not be recognized by Omnis Studio.

You can save the settings for the Port device to a profile using the Port profile editor: see later in this chapter for a description of the Port Profile editor. You can also setup the port using the *Set port parameters* command.

### File Report Destination

The File print destination sends the current report to a file. If you double-click on the File icon in the Report Destination dialog Omnis prompts you for a file name. Omnis does not close the file at the end of the report so you can append multiple reports into a single file. This option enables the Page sizes pane in the Report Destination dialog.

In the Page sizes pane you can specify the number of lines per page to use in reports printed to a file or a port. Omnis stores the setting in the Omnis configuration file.

If you check the Generate paged reports check box, you can also check the Send form feed check box, which tells Omnis to terminate pages with a form feed, or fill in the Line per page field with a number of lines to which to pad out each page. Checking the Restrict page width option lets you enter the number of Characters per line.

### RTF Report Destination

The RTF print device sends the current report to an RTF file. If you double-click on the RTF icon in the Report Destination dialog Omnis prompts you for a file name. You can also set the file name under the Parameters pane in the Report Destination dialog, as well as control the how images are embedded or linked. The default behavior is to embed images in the RTF file, but you can link the images in which case any images are stored as separate files and linked to the RTF file; you can also ignore the images altogether.

The following code allows you to set the name of the RTF file:

Do $devices.RTF.$setparam(kDevRtfFileName,'/Users/test/Desktop/test.rtf')

### HTML Report Destination

The HTML report device is a custom device that prints a report to an HTML file on disk. When installed and loaded it appears in the print destination dialog and behaves like any other standard printing device. You can send any Omnis report to the HTML device and access and change the device using the notation. If you double-click on the HTML icon in the Report Destination dialog Omnis prompts you for a file name. The HTML output uses UTF-8: if you use a template file, that must also be UTF-8 encoded.

The HTML printing device uses HTML tables and standard HTML tags to position and structure the output of the Omnis report. The default background color of the HTML file is white. The color of the text in the original report class is retained in the HTML output file. Where possible, the device converts any image or picture data into JPEG images, which are written to disk and linked to the output HTML file.

### PDF Report Destination

The PDF report destination sends the report output to a PDF file which can be viewed in a browser or on the end user’s desktop using the default PDF viewer.

The alternative of using the Printer report destination with $macosdesttype set to PDF uses bitmaps to render background objects, and their appearance in a scaled PDF will vary depending on the scaling factor and the algorithm used by the PDF viewer to scale the bitmap.

#### Hyperlinks in PDF Reports

Hyperlinks are supported in PDF and Page Preview report destinations. To add a hyperlink to a report, use the HTML link external component (text, picture and icon are all supported), and set the $address property to the target of the link, for example:

https://omnis.net

mailto:bob.smith@omnis.net

or for Page Preview reports only, and ignored for PDF reports:

omnis:p1,p2,p3,p4

where the data after omnis: is a comma separated list of parameter values, which can be integer or character, and must not contain ".

In the latter case, when the user clicks the omnis: link, Omnis looks for a method called $previewurlclicked. Firstly, if the report has been sent to a window field, Omnis looks for the method in the window instance containing the screen report field, otherwise if the first test fails, Omnis looks for the method in the task that printed the report. If Omnis finds the method, it calls $previewurlclicked passing it the following parameters:

1. An item reference to the report instance.
2. The ident of the report object.
3. A row created by adding a column for each comma delimited item in the data after omnis: in the link.

You can create a method called $previewurlclicked in either the window or task, as above, and react to the parameters passed.

#### Printing Background Images to PDF

Background images on reports need to be shared pictures (PNGs) to print on the Linux headless server. To solve this, go to the library prefs and set $sharedpictures to kSharedPicModeTrueColor. Then open each affected report class in the report class editor; Omnis will ask if pictures are to be converted to shared, so respond with Yes. The reports will now print to PDF in the headless server, and additionally the report classes are much smaller.

Omnis will report an error if the headless server cannot print to PDF.

#### PDF Font Mapping

When using custom fonts for PDF printing there may be a mis-match between the name of a font and its Window registry entry, which results in the font not being found and the report not being rendered correctly. To rectify this, you can add mappings to the "pdf" entry in config.json (that apply to the Windows platform only), to map a font name to its entry in the registry. For example, you can map the font name "Proxima Nova Rg" to its registry entry "Proxima Nova Regular", using the following item in the config.json file.

"pdf": {

"plainSuffixes": "Regular,Standard,Normal,Normale",

"Proxima Nova Rg": "Proxima Nova Regular",

"Proxima Nova Rg Bold": "Proxima Nova Bold"

},

### Memory Report Device

The Memory device lets you send a report to a binary variable or field, which you can hold in memory or save in a database. At a later date you can reload the contents of the binary variable or field and print the report to the printer or any other destination.

You can access the Memory device using the notation only via the $root.$devices notation group. By default this device *is* not shown in the Print Destination dialog, but you can show it using the following method (although in practice you would not use this destination for end users):

###### Do $root.$devices.Memory.$visible.$assign(kTrue)

You can print the output from the Memory device using the *Print report from memory* command.

### DDE/Publisher Device (Obsolete)

*This feature is no longer supported, since it relates to very old Windows and Mac operating systems, but is still available for backwards compatibility.* The DDE/Publisher lets you send a report via DDE under Windows, or to an Edition under macOS.

You can access the DDE/Publisher device using the notation only via the $root.$devices notation group. By default this device *is not shown* in the Print Destination dialog, but you can show it using the following method (although in practice you would not use this destination for end users):

###### Do $root.$devices.//DDE/Publisher//.$visible.$assign(kTrue)

### Printing Errors

The print manager reports the following error codes and text. You can setup error handlers to manage these errors.

|  |  |
| --- | --- |
|  |  |
| 1001650 | Non-fatal print manager error |
| 1001670 | Fatal print manager error |
| 1001680 | Print manager system error; the code is shown in the error text |
| 1001681 | Other Omnis error reported by print manager |

### Tabs in Reports

There is an entry in config.json to handle tabs in reports better, so they are displayed properly on reports, e.g. on the preview thumbnail in page preview. The new option "replaceTabsInRTFwithSpacesWhenAddingToReport" is in a new group called "docview". This can be a value from zero to 32 inclusive, the default is 2. Zero means leave the text unchanged (the behavior prior to this update). 1-32 means replace each tab character found in the text with 1-32 spaces, when adding the text to a print job.

### Using style() in Reports

In versions prior to Studio 10.x, if the style() function was used in a report instance belonging to a remote task incompatible results were generated (but this has been fixed).

The style() function usually generates different results when used in a remote task instance, and the resulting style is suitable for use with the JavaScript client. The fix allows normal, non-JS client results, to be generated by style() when running in a report instance inside a remote task.

However, even with this fix, you should note that the call to print the report from the remote form, which passes the results of style() from some remote form code, will not work: you need to pass the icon id as the parameter, and call style() from within the report instance to make this work with this fix.

## Report and Field Methods

You can create a report class, add fields and objects to the report from the Component Store, but to print sophisticated reports you will need to add some programming behind the fields and sections on your report. To do this, you need to write code that uses the Omnis print commands or methods. You can add *class methods* to the report itself to control printing, and you can add *field methods* to each field or section marker on your report to control things like the interval breaks and subtotals.

You can add up to 501 methods to each field or section on your report, and a further 501 methods to your report class. You enter the methods for a report class and its fields using the method editor.

When you create a report from the Component Store it contains a $construct() and $destruct() method by default. You can add code to these methods to control the opening and closing of the report instance. For example, if your report uses a list you can build the list in the $construct() method in the report. You can use the $open() method or the *Prepare for print* command to open a report instance, you can finish a report using $endprint() or the *End print* command, and you can close a report instance using the $close() method. You can send a list of parameters to the $construct() method when you open the report instance using the $open() method. You can send data to a report instance record by record using the *Print record* command, or print an entire report using the *Print report* command. Alternatively, you can send print messages to a report instance using the notation. For example, you can send a $printrecord() message to print a record to the report instance, or send an $endprint() message to finish the report; there is no equivalent method for the *Print report* command. You can override the default handling for these messages by writing your own custom methods with the same name. You enter these custom methods in the class methods for the report class.

To add a method to a report class

1. Open your report class
2. Right-click on the report background to open the report context menu
3. Select the Class Methods option
4. Right-click in the method list in the method editor and add your method

To add a method to a report field or section

1. Open your report class
2. Right-click on the field or section to open its context menu
3. Select the Field methods option
4. Right-click in the method list in the method editor and add your method

The next section in this chapter describes the type of methods you can add to report classes and objects.

## Print Devices and the Current Device

The following properties under $root handle the group of currently installed print devices or destinations, and the current printing device.

1. $devices  
   group of currently installed printing devices, including Printer, Preview, Screen, Disk, Memory, Clipboard, Port, File, DDE/Publisher, and any custom devices you may have installed. You can set a reference to a device by using

###### Set reference MyRef to $devices.Screen

1. $cdevice  
   the current printing device or report destination. You can change $cdevice by assigning a device from the $devices group, for example, to specify the screen as the current device use one of:

###### Calculate $cdevice as kDevScreen

###### Calculate $cdevice as $devices.Screen

###### Calculate $cdevice as $devices.$findident(kDevScreen)

### Print Devices

The $root.$devices group contains the currently installed printing devices plus any custom devices you may have installed. A device has the following properties; $canassign for these properties is true unless specified.

1. $name  
   the name of the device; $canassign is false
2. $title  
   the string used to identify the device in the Print destination dialog
3. $iconid  
   the id of the icon for the device as displayed in the Print destination dialog, zero by default which means the device uses the default icon
4. $ident  
   a unique numeric identifier for the device in the $devices group; $canassign is false
5. $visible  
   if true, the device is shown in the Print destination dialog
6. $isopen  
   returns true if the device is open and in use; $canassign is false
7. $istextbased  
   returns true if the device is text-based, otherwise, the device is image-based, such as Printer, Screen, or Preview; $canassign is false
8. $cangeneratepages  
   this is a read only property. If it returns true, the device can generate pages and all normal page headers and footers will be printed. If it returns false, only the report header and first page header are printed. No footer section is printed.
9. $cankeepopen  
   returns true if the device can be kept open for long periods of time, such as the File device; otherwise, the device should be opened prior to printing and closed immediately after printing has finished, for example you must do this for the Printer; $canassign is false

You can use the following methods for a device; $cando() returns true if a device supports the method.

1. $open()  
   opens the device ready for printing or transmitting text or data. Some devices such as the Screen or Preview can only be opened from a print job when printing a report
2. $close()  
   closes the device, if the device is open

The following example prints two reports in the same print job, and uses the $open() and $close() methods to initialize the Printer.

###### Set reference theDevice to $devices.Printer If theDevice.$isopen ## check if printer is in use   If theDevice.$canclose()     Do theDevice.$close()   Else     Quit method kFalse ## if device can’t be closed   End if End If Do theDevice.$open() Returns ok ## open the printer If ok ## print the reports   Set report name reportOne   Print report   Set report name reportTwo   Print report   Do theDevice.$close() Returns ok ## close the printer End If

1. $canclose()  
   returns true if the device can be closed. If you opened the device using $open(), this method returns true; if you opened the device via a print job and the job is still in progress, it returns false
2. $sendtext( cText, bNewLine, bFormFeed )  
   sends the text in cText to the current device; all normal character conversion takes place. If bNewLine is true, the device advances to a new line or sends an end of line character; if bFormFeed is true, a new page is started, or a form feed character is sent. Data is sent in parameter order: first text, then the new line, then the form feed.

The following example sends some text to the File device.

###### Set reference theDevice to $devices.File If theDevice.$sendtext.$cando()   Do $prefs.$printfile.$assign(‘HD:MyFile’)   Do theDevice.$open() Returns ok   If ok     Do theDevice.$sendtext(‘Some text’,kTrue) Returns ok     Do theDevice.$sendtext(‘More text’,kTrue) Returns ok     Do theDevice.$close() Returns ok   End If End If

1. $senddata( cData[,cData1]… )  
   sends the specified data in a binary format to the device; no character conversion takes place unless the data is of type kCharacter. If more than one parameter is specified the data is sent in individual packets

When using the $senddata() method you must consider type conversion. The method expects binary data, and therefore any data which is not in a binary format is converted to binary. For example, if you pass an integer variable, the data is converted to a 4 byte binary. In some cases, due to cross platform incompatibilities, if you want to be certain of the order in which the data is sent, and of the values which are sent, you should use variables of type Short integer (0 to 255), for example

###### Calculate myShortInt1 as 13 Calculate myShortInt2 as 10 Do myDevice.$senddata( myShortInt1, myShortInt2 )

You can send raw data to the Port or File device. The following example prints a report to a binary variable and sends the binary data to the Port device.

###### # print a report to a binary variable Do $cdevice.$assign($devices.Memory) Do $prefs.$reportdataname.$assign(‘myBinaryField’) Set report name myReport Print report # now send the report to the port Set reference theDevice to $devices.Port If theDevice.$senddata.$cando()   Do theDevice.$open() Returns ok   If ok     Do theDevice.$sendata(myBinaryField) Returns ok     Do theDevice.$close() Returns ok   End If End If

1. $flush()  
   flushes the device. For the File device $flush() will ensure all data is written to disk; you can safely call $flush() for devices which do not support this method; $cando() returns true for all devices that support $senddata() or $sendtext()

## Global Printing Preferences

There are a number of Omnis preferences under $root.$prefs that handle the print devices and their parameters. You can set these using the Property Manager or using the notation.

1. $reportfile  
   the full path and file name for the Disk device
2. $printfile  
   the full path and file name for the File device
3. $editionfile  
   the full path and file name for the DDE/Publisher device
4. $pages  
   the page or page numbers to be sent to the device; all devices support this property. You can specify pages as a comma-separated list or range of pages separated by a hyphen, or any combination. Prefixing a range with an “e” will print even pages within the range, or with an “o” will print odd pages within the range. For example   
   1,3,7,10-15,25-20,e30-40,o30-40
5. $reportdataname  
   the name of the binary field for the Memory device
6. $reportfield  
   the name of the window field for a Preview or Screen report; if you specify this property the report is redirected to the window field
7. $windowprefs  
   the optional title and screen coordinates for a Screen or Preview window; the syntax is the same as the *Open window command*, such as My Title/50/50/400/300/STK/CEN; the title is also used as the document name when printing to the Printer
8. $waitforuser  
   if true, method execution is halted until the user closes the Screen or Preview window
9. $hideuntilcomplete  
   if true, a Screen or Preview window remains hidden until the report is finished

The following example specifies the Screen as the current device and sets up the preferences for the report window.

###### Do $cdevice.$assign(kDevScreen) Do $prefs.$windowprefs.$assign(‘MyTitle/20/20/420/520/CEN’) Do $prefs.$waitforuser.$assign(kFalse) Do $prefs.$hideuntilcomplete.$assign(kTrue)

1. $charsperinch  
   the number of characters per inch when printing to a text-based device
2. $linesperinch  
   the number of lines per inch when printing to a text-based device
3. $generatepages  
   if true, reports generate paged output when printing to text-based devices, that is, page headers and footers are generated as normal; otherwise if false, only one report header and page header is printed at the beginning of the report
4. $linesperpage  
   the number of lines per page when $generatepages is true
5. $restrictpagewidth  
   if true, the width of a page is restricted when printing to text-based devices
6. $charsperline  
   the number of characters per line when $restrictpagewidth is true
7. $sendformfeed  
   if true, form feeds are sent to text-based devices after each page
8. $appendfile  
   if true, data is appended to the current print file specified in $printfile, otherwise if false, the file is overwritten when printing to the File device; note if the device is already open prior to printing a report, the file is appended to regardless
9. $istext  
   if true, forces a non-text device to behave like a text-based device using the same preferences as text-based devices
10. $portname  
    the name of the port when printing to the Port device (Not supported on macOS)
11. $portspeed  
    the port speed setting when printing to the Port device (Not supported on macOS)
12. $porthandshake  
    the handshake when printing to the Port device; this can be kPortNoHandshake, kPortXonXoff, or kPortHardware (Not supported on macOS)
13. $portparity  
    the parity checking when printing to the Port device; this can be kPortNoParity, kPortOddParity, or kPortEvenParity (Not supported on macOS)
14. $portdatabits  
    the number of databits when printing to the Port device; this can be kPort7DataBits, or kPort8DataBits (Not supported on macOS)
15. $portstopbits  
    the number of stop bits to be used when printing to the Port device. This can be kPort1StopBit or kPort2StopBits (Not Supported on macOS)

The following example sets up the preferences for the Port device.

###### Do $prefs.$portspeed.$assign(9600) Do $prefs.$porthandshake.$assign(kPortNoParity) Do $prefs.$portdatabits.$assign(kPort8DataBits) Do $prefs.$portstopbits.$assign(kPort1StopBit) Do $prefs.$porthandshake.$assign(kPortXonXoff) Do $prefs.$charsperinch.$assign(10) Do $prefs.$linesperinch.$assign(6) # Note $charsperinch and $linesperinch are used for # all text-based devices

There is also a group of Page setup properties under $root.$prefs giving access to the global page settings. These are

1. $orientation  
   the page orientation; this can be kOrientDefault, kOrientPortrait, or kOrientLandscape
2. $paper  
   the paper type or size, a constant; one of 50 or so paper sizes or types including US Letter, European A sizes (from A6 upwards), envelope sizes, custom sizes, and so on
3. $paperlength  
   the length of the paper in cms or inches depending on the $usecms preference
4. $paperwidth  
   the width of the paper in cms or inches depending on the $usecms preference
5. $scale  
   the scaling factor in percent
6. $copies  
   the number of copies

The Omnis root method $getprinterlist() allows you to get a list of printers available to the current user. For example, the following method will return a list of printers:

###### # define local vars printer (Char), mylist (List) Do mylist.$define(printer) Do $root.$getprinterlist(mylist)

The current line of the list indicates the current printer.

The Omnis preference $disablereportworkingmessage allows you to disable working messages for reports, which you might want to do when printing to a Print Review window.

1. $disablereportworkingmessage  
   If true, the 'Sending report to...' working message is not shown when printing a report.

This property only applies to reports being printed on the main thread (as reports in JavaScript client threads do not show a working message). Note that you cannot cancel the report if you set this property to true. You may also need to use $modes.$fixedcursor and $modes.$ccursor if you want to display a cursor other than the busy cursor while printing the report.

## Report Instances

Report instances have the following methods.

1. $printrecord()  
   prints the Record section; same as the *Print record* command
2. $printtotals(*section*)  
   triggers a subtotal or totals section; *section* is the highest level subtotal to be printed, a constant, such as kSubtotal5 or kTotals
3. $printsection(iSection[,bPosnIsRecord=kFalse])   
   prints a report section, bPosnIsRecord applies to positioning sections only; *section* is a constant, kRecord, kTotals, and so on, or a reference to a field on the report instance: If bPosnIsRecord is kFalse (the default), this method prints a section based on the position of the previous section; otherwise, when true the section is printed as a record
4. $accumulate(*section*)  
   accumulates the subtotals and totals section, and is sent during the printing of a record section
5. $checkbreak()  
   checks if a subtotal break is required, returns a constant: kSubtotal1 to kSubtotal9 or kNone if subtotal break is not required
6. $skipsection()  
   skips the current section; if you call this during $print() for a field, no further fields will be printed for that section
7. $startpage(*pagenumber*)  
   starts a new page; adds the page header section to the page, and for the first page also adds the report header section
8. $endpage(*pagenumber*)  
   ends a page and adds the footer section to the page; without parameter ends all pages which have been started
9. $ejectpage(*pagenumber*)  
   ejects a page; without parameter ejects all pages which have been ended and not ejected; this method ejects pages which have an active section intercepting their boundary when the section has finished printing
10. $endprint()  
    finishes the report; prints the final subtotals and totals sections and ejects all the remaining pages
11. $openjobsetup()  
    opens the job setup dialog. You can call this method immediately after $open() for a report; if it returns kFalse as the result, the user has selected Cancel, and you should close the report instance. You cannot call $openjobsetup() during $construct() since a print job is not created until $construct() finishes.
12. $cdevice  
    reference to the printing device for the instance; if you wish to change the device, you must do so before returning from $construct() of the report instance, and before you start printing the first record. For example, execute the following at the start of $construct() to specify the page Preview device for the current instance

###### Do $cinst.$cdevice.$assign($devices.Preview)

The device preferences are listed under the global printing preferences. Report instances have their own printing preferences which are local to the instance. They take their initial values from the global printing preferences. You can only assign values to these properties in $construct(), and before you start printing the first record.

The $firstpage property always returns 1, and $canassign() is false. The $lastpage property returns the last page. You cannot set $lastpage in the report instance to reduce the number of pages generated, that is, once pages have been generated they cannot be removed from a print job.

Note that the $pageheight property of a report instance returns the height of the printable area excluding the margins, headers, and footer areas of the report.

The following method generates subtotal breaks every fifth record. The $reccount property is incremented and the subtotals accumulated manually.

###### Do $reports.Report2.$open('\*') Returns Myreport For lineno from 1 to Mylist.$linecount step 1   Do Mylist.[lineno].$loadcols()   Calculate Myreport.$reccount as Myreport.$reccount+1   Do Myreport.$printsection(kRecord)   Do Myreport.$accumulate()   If mod(lineno,5)=0     Do Myreport.$printtotals(kSubtotal1)   End If End For Do myreport.$endprint()

### Page Setup Report instance properties

You can change the page setup information of a report instance without effecting the global settings. The properties which can be set are;

1. $pagesetupdata  
   this can only be calculated prior to printing the first record; the best time is during $construct. When a print job has started, $canassign returns kFalse.
2. $orientation, $paper, $paperlength, $paperwidth, $scale, and $copies  
   any of these properties can be changed at any time during a print job, and will effect the next page to be generated. When $startpage for a page has been called, changing these properties will take effect from the next page onwards. A good time to make changes for the next page is during a $endpage for the current page, but it can be done from anywhere prior to the $startpage call for a page to be effected
3. $loadpagesetup  
   When true, the page setup information stored with the report is automatically loaded and used when the report is printed. This will *not* affect the global page setup information. The page setup information is applied to the page setup information of the report instance prior to calling the $construct method or opening the job setup dialog. It is also possible to load the page setup data from the $construct method of the report instance by assigning $loadpagesetup ( $cinst.$loadpagesetup.$assign(kTrue) ). Assigning kFalse has no effect., For new reports, this property will be set to kTrue by default. For existing reports, this property will be false.

Once a print job is complete and $endprint has been called, $canassign returns kFalse for all these properties.

### Printing a report from a list

The following method prints a report from a list and uses a *For* loop to print the report record by record.

###### Do $reports.Report1.$open('\*') Returns Myreport For lineno from 1 to Mylist.$linecount step 1   Do Mylist.[lineno].$loadcols()   Do Myreport.$printrecord() End For Do Myreport.$endprint()

Note that you should not use hash variables to define the columns in list if it is going to be used as the basis of a report.

### Screen Report Fields

Screen Report Fields have two methods that allow you to zoom or search the screen report. Once the screen report has been sent to the field you can use $zoom() to scale the report from 25% to 200%, and the $searchreport() method allows you to search for the specified text within the screen report content.

#### Zoom

The method $zoom(*iZoom*) zooms the screen report field, where iZoom can be positive (indicating a percentage between 25 and 200% inclusive), or 0 meaning zoom to fit, or negative (-1 to -7) where -iZoom indexes the 7 standard zoom factors from smallest to largest.

#### Search

The method $searchreport(*cText*[,*bIgnoreCase*=kTrue,*bNext*=kTrue]) searches the report for cText. Further calls with the same cText and bIgnoreCase search for the next (bNext kTrue) or previous (bNext kFalse) match. Empty cText clears the search.

There is an event which works in conjunction with $searchreport (needed because the search occurs in a background thread). The event enables you to manage next and previous buttons, and status text. The next and previous buttons are assumed to start in disabled state. The event evReportSearchStatus is sent to the report field when the report search status changes: this has one event parameter pReportSearchStatus which is a row with 4 columns, as follows:

|  |  |
| --- | --- |
| next | If true,search next can be enabled as there is another search result later in the report |
| prev | If true,search previous can be enabled as there is another search result earlier in the report |
| count | The count of search results |
| index | The 1-based index of the current search result |

## Report Field and Section Methods

Report fields and sections contain a $print() method that controls that particular field or section when it is printed. Every time a field or section is encountered during printing its $print() method is called, so for fields in the report Record section $print() is called for every row of data. You must end your own custom $print() methods with a *Do default* command to carry out the default processing for that line after your code has executed.

For example, the following $print() method for a report field prints the field in bold if its value is greater than 1000.

###### If parm\_value>1000   Do $crecipient.$fontstyle.$assign(kBold) Else   Do $crecipient.$fontstyle.$assign(kPlain) End If Do default

## Report Object Positioning

When a report field prints, its position and data are passed to its $print() method; when a report section prints its position only is passed. You can set up parameter variables of type Field reference in the $print() method for a report section or field to receive its position and data. You can manipulate the position variable using the report object positioning notation. If you change the position of a section all objects in that section are affected together with all subsequent sections in the report. Making changes to the position of an object does not affect other objects.

A report position variable has the following properties.

1. $inst  
   the report instance to which the position belongs
2. $posmode  
   the mode of the report position, which is one of the following constants; assigning $posmode does not change the physical position of the object, but it does change its coordinates to the new coordinate system.

|  |  |
| --- | --- |
|  |  |
| kPosGlobal | the position is global to the print job, relative to the top-left of the local area of the first page |
| kPosPaper | the position is relative to the top-left of the paper edge of the page specified by $posvertpage and $poshorzpage |
| kPosPrintable | the position is relative to the top-left of the printable area of the page specified by $posvertpage and $poshorzpage |
| kPosLocal | the position is relative to the top-left of the local area (excluding the header and footer sections, and the margins) of the page specified by $posvertpage and $poshorzpage |
| kPosHeader | the position is relative to the top-left of the header area (union of report and page header sections) of the page specified by $posvertpage and $poshorzpage |
| kPosFooter | the position is relative to the top-left of the footer area of the page specified by $posvertpage and $poshorzpage |
| kPosSection | the position is relative to the top-left of the section specified by $possectident |

In addition, you can set $posmode to one of the following values to return the coordinates of an area on the page specified by $posvertpage and $poshorzpage.

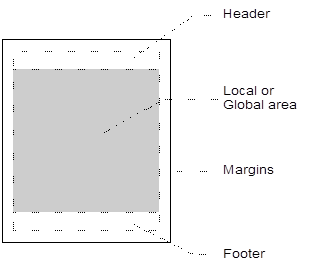
|  |  |
| --- | --- |
|  |  |
| kBndsGlobal | returns kPosGlobal coordinates. The top, left, width, and height are calculated to global coordinates of the local area of the page |
| kBndsPaper | returns kPosPaper coordinates. The top and left are zero, and the height and width are calculated to the height and width of the paper of the page |
| kBndsPrintable | returns kPosPrintable coordinates. The top and left are zero, and the height and width are calculated to the height and width of the printable area of the page |
| kBndsLocal | returns kPosLocal coordinates. The top and left are zero, and the height and width are calculated to the height and width of the local area of the page |
| kBndsHeader | returns kPosHeader coordinates. The top and left are zero, and the height and width are calculated to the height and width of the header area of the page |
| kBndsFooter | returns kPosFooter coordinates. The top and left are zero, and the height and width are calculated to the height and width of the footer area of the page |

1. $possectident  
   the $ident of the section when $posmode is kPosSection
2. $posvertpage  
   the vertical page number when $posmode is *not* kPosGlobal or kPosSection
3. $poshorzpage  
   the horizontal page number when $posmode is *not* kPosGlobal or kPosSection. This will usually be set to 1. Horizontal page numbers apply when horizontal pages are enabled
4. $top  
   the top of the position in cms or inches local to its $posmode
5. $left  
   the left of the position in cms or inches local to its $posmode
6. $height  
   the height of the position in cms or inches
7. $width  
   the width of the position in cms or inches

Measurements are in either cms or inches depending on the setting of the **usecms** Omnis preference which you can change in the Property Manager using the Tools>>Options/Preferences menu option.

### Page layout

To understand the positioning notation it helps to look at the layout of the report on paper or screen. The area available for printing is limited to the printable area on the paper as determined by the printer or device. Within this space Omnis reports print to the header, footer, and local or global areas, that is, the space remaining after subtracting the header, footer, and margins specified in the class. Note that Omnis subtracts the margins specified in the class from the paper edge, rather than the boundary of the printable area.



The position of a report object, either a section or report field, is relative to the local area on the current page, or the global area for the entire report.

|  |  |
| --- | --- |
|  |  |
| Local coordinates are relative to the local area on the current page | Global coordinates are relative to the global area for the entire report |

The following example method produces a report with multiple columns by configuring itself according to the current paper size and orientation. The report class contains various instance variables including iCurColumn, iMaxColumns, iLeftAdjust to handle columns and a global left adjustment. The data is taken from a list, but your data can be from any source. The Record section contains one field that gets its data from the list. Note the code for this method does the positioning and the *Do default* command prints the section.

###### # $print() method for Record section in a column report # Declare Parameter var pThePos of type Field reference # and Local var posBnds of Row type # pThePos is in global coordinates and does not contain the page # number, so make a copy and convert it to page-based coordinates Calculate posBnds as pThePos Calculate posBnds.$posmode as kPosLocal # Fetch the global boundaries for the page: we can do this now since # setting $posmode to kPosLocal set $poshorzpage and $posvertpage Calculate posBnds.$posmode as kBndsGlobal # Check if the bottom of the section will fit on the page If (pThePos.$top+pThePos.$height)>(posBnds.$top+posBnds.$height)   # if it doesn't fit is there room for a new col on current page   If (iCurColumn<iMaxColumns)     Calculate iCurColumn as iCurColumn+1   Else     # put the section at the top of the next page for column one     Do posBnds.$offset(0,posBnds.$height)     Calculate iCurColumn as 1   End If   # now calculate the section’s top position based on posBnds.$top   Calculate pThePos.$top as posBnds.$top   # calculate the section’s left pos based on current column number   Calculate pThePos.$left as (iCurColumn-1)\*$cinst.$labelwidth+iLeftAdjust Else If not(pThePos.$left)   Calculate pThePos.$left as iLeftAdjust End If Do default ## this prints the Record section

You can use the $offset(x,y) method to move the object horizontally by x units (unless $horzpages is enabled) and vertically by y units. The units are defined by $prefs.$UseCMS and maybe either positive or negative.

## Report Inheritance

When a report is based on a report superclass several items are inherited including report header and footer sections (e.g. page header, subtototals, footer, etc), the objects within these sections, as well as the sort fields in the report superclass. The objects within the record section of the report superclass are not inherited. You can, therefore, create a report template containing your company identity and base all other reports on the template to ensure a uniform design and layout is maintained across all your reports.

Inherited sections cannot be manipulated in any way. You cannot resize an inherited section by moving the section object immediately below the inherited section, or change the properties of the section or its objects. When a section is inherited, the height of the section is determined from the superclass, and all other sections below are moved up or down accordingly.

For inherited sections, the text of the section bar and the text of the section connection lines are shown in blue. The background is shown in gray to indicate that this section cannot be manipulated.

### Inheriting Sections

#### Modifying a Superclass

When modifying the section of a superclass, e.g. adding, removing or changing objects within a section, these changes will be reflected in any report class which inherits the section.

Removing an entire section from a superclass will also affect all subclasses. However, adding a new section to a superclass will not affect any of its subclasses. To inherit the new section, subclasses need to be modified to inherit this new section.

Any changes made to sort fields will be reflected in all subclasses.

#### Inheriting/Overload a section

To inherit or overload a section, right click on the section in the Property Manager. A context menu opens allowing you to inherit or overload the property. You cannot manipulate inherited sections, or inherited section objects, or to add more objects to an inherited section.

#### Subtotal Sections and Sort Fields

A subclass inheriting a subtotal section will also inherit the sort field from the super class, unless the subclass has already specified a sort field for the subtotal section in question. Sort fields can be overloaded or inherited via the context menu of the sort field. The properties of an inherited sort field cannot be changed. If you want to change the properties of an inherited sort field you must first overload the sort field.

When overloading a sort field, the properties of the sort field originally inherited are maintained. If a sort field is inherited which has not been specified by its super class, the name will be displayed as #???.

#### Positioning Sections

If an inherited section contains positioning sections, these are also inherited. It is *not* possible to add additional positioning sections to an inherited section. It is not possible to inherit position sections without inheriting their “main section”.

#### Notation

You can inherit or overload sections using notation. To do this you assign $isinherited of the section notation. For example:

###### Do $clib.$reports.myreport.$pageheader.$isinherited.$assign(kTrue)

To inherit sort fields using notation:

###### Do $clib.$reports.myreport.$sorts.1.$isinherited.$assign(kTrue)

The objects of inherited sections are not part of the subclass and as such will not appear in the list of objects of the report class, but will appear in the list of objects of a report instance.

The following section properties can be inherited: $reportheader, $pageheader, $subtotalhead, $subtotalhead1 to …9, $subtotal9 to …1, $pagefooter, $totals.

## Report Fonts

If you are developing an application for a cross platform environment (for desktop apps), you may want to set up the system font tables to allow the fonts used in your application to map correctly across the different platforms. There is a system font table in a library for report classes and window classes for each platform supported in Omnis.

The fonts in the report font table will appear in the $font property for report objects. So even if you are developing an application for a single platform, you may still want to edit the report font table to add fonts to those already available for report objects.

|  |  |
| --- | --- |
|  |  |
| #WIRFONTS | Report font table for Windows OS |
| #MARFONTS | Report font table for Mac (9 & earlier) |
| #UXRFONTS | Report font table for Unix |
| #MXRFONTS | Report font table for OSX/macOS (10 onwards) |

To view the report fonts system table

1. Use the Browser Options dialog (press F7/Cmnd-7 while the Studio Browser is on top) to make sure the system tables are visible
2. Double-click on the **System Tables** folder
3. Double-click on **#WIRFONTS** or the report font table for your platform

The #WIRFONTS system table contains a list of fonts that are available in Omnis by default. Each row in the font list displays the corresponding font for each platform supported in Omnis. To change the font mapping, replace the name of a font either by typing its name or selecting it from the list of fonts. To add a font, click in the next available line in the font list and add the name of the font. Add a font name for each platform.

The font table editor loads or creates a font table for each platform (corresponding to each column in the editor) and allows you to edit them all simultaneously. Therefore when you edit the report font table for the first time, and click OK to finish editing it, a new system table is added to your library for each platform supported in Omnis, other than your current platform.

### Unknown macOS Fonts

You can map unknown macOS fonts in reports, such as the New York legacy Mac font, to alternative fonts. You can add the "unknownMacOSFonts" item to the "pdf" section of config.json to specify the font mapping. For example:

"pdf": {  
 "unknownMacOSFonts": {  
 "New York": "Times New Roman",  
 "default": "Lucida Grande"  
 }  
 },

The members of the unknownMacOSFonts object are the names of the unknown fonts to be mapped, and the name of the replacement font. A “default” member can be included to map all other fonts not listed in unknownMacOSFonts to the specified font.

## Report Text Labels

You can use the standard built-in Text and Label objects in your report classes to label report fields and so on. In addition, you can use the String Label object which allows you provide multi-language labels on your reports: for information about the use of String Labels and String tables, see the *Localization* chapter.

The Wrapping List and Data Grid report fields support the use of styled text, provided their $::styledtext property is enabled. You can insert styling characters or “text escapes” into text objects and other fields in Omnis reports using the style() function.

## Port Profiles

A port profile is a named collection of information sufficient to completely describe the operating system configuration of a port. Under Windows, the port profile information corresponds to the information required to set the fields in a DCB for a serial port (under Win16 the DCB is different to Win32). There is no port profile for a parallel port, since it requires no operating system configuration information.

Under macOS, the port profile information corresponds to the information required to make the SerReset and SerHShake API calls. (Note: Port profiles are not currently supported on macOS.)

The information required to completely configure a port therefore comprises:

1. The port profile (not required for parallel ports)

2. Characters per inch.

3. Lines per inch.

### Port Profile Management

Each port profile is stored in a file. The Ports folder in the Omnis tree contains the port profiles. The port profile file contains:

1. An indicator of the platform to which the profile corresponds - Win16, Win32 or macOS. This allows for runtime checking.
2. The name of the profile, to be used in the report destination dialog, and as an argument to the *Set port parameters* command. Note that profile names are not case sensitive.
3. The profile data.

Note that this means that each profile file only contains the data for a single platform.

You can create and edit profiles using the Port Profile Editor on the Add-ons submenu in the Tools menu (it is not available on some older operating systems). The profile editor must run on the platform for which the profiles are to be created. If you wish to edit port profiles using a runtime version, there is a folder in the root of the Studio CD, called RTTools, containing the editor library, which you copy to your hard disk, and open to run the editor.

### Port Profiles at Runtime

#### Report Destination Dialog

When the report destination is set to Port, the parameters tab of the dialog displays the port configuration information. There is a dropdown list on this tab, which contains a list of profile names, and one additional entry, ‘Use options below’. When this is selected, you can specify the port parameters in the dialog. When a profile is selected, Omnis configures the port using the information in the profile. Note that the dropdown list and configuration fields on this tab are disabled if you select a parallel port.

#### Set port parameters command

You can use a port profile name in place of the port parameter list:

###### Set port parameters {Profile name}

When this command executes, it first checks the entire parameter string against the list of profiles. If it matches an entry in the list, the command uses the profile to configure the port; otherwise, the command treats the parameter string as a parameter list.

#### Notation

The relevant $port… notation such as $portparity is unassignable when a port profile is selected. The $portprofile property is the port profile for the current port on the macOS, or just the single port profile under Windows.

### Printer Escapes

When sending to printer, escape characters are not interpreted correctly, because the printer driver attempts to draw them as data. You must send reports containing printer escapes to port to ensure escape sequences are interpreted correctly.

The Omnis root preference $exportencoding ($root.$prefs) determines how the data is converted before Omnis sends it to the port. Set this to kUniTypeAnsiLatin1 for printer escapes to be interpreted correctly.

The escapes generated by the style() function cannot be mixed (in a report field's data) with escapes specific to a printer.

## Labels

To print labels in Omnis you need to create a report class and set up its properties for label printing. You can create the report class using the standard SQL or Omnis report wizards, or you can create an entirely New Report and add the fields yourself. This section uses the Omnis Report wizard as the basis for a Customer address label, but the process is the same for any label report.

To create the basis of your label report

1. Create a new report class using the Omnis Report wizard and include the fields you want in your label
2. Open the report class to modify it
3. Delete the header section and any labels the report wizard places on the report class, but leave the data fields; your report class should look something like the following

To change this report class into a label report you need to change some of its properties, set the properties of the Record section to position your labels on the printed page, and as a further enhancement you can set the properties of some of the fields on the report to exclude empty lines. Note that all measurements use the current units set in the **usecms** Omnis preference.

To set the label properties of a report class

1. Click on the background of your report class to view its properties
2. Set the **islabel** property to kTrue
3. Set the **labelcount** property to the number of labels across the page; for example, for standard 3 x 8 laser labels you set **labelcount** to 3
4. Specify the width of a single label in the **labelwidth** property; if there are spaces between your labels, include the space in the label width, that is, the labelwidth is the distance between one record and next across the sheet of labels
5. If you want to print more than one label for each row or record of data set the **repeatfactor** property, otherwise leave it set to 1 for a single copy of each label

To specify the distance between each row of labels down the page, you change the properties of the Record section in your report class.

1. Click on the Record section to view its properties
2. Set the **startmode** property to kFromTop, and in the **startspacing** property enter the distance between the top of one row of labels and the next going down the label sheet

Excluding Empty Lines

When you print your labels some of the fields may be empty and a blank line is printed. However you can stop a field from printing and move up all subsequent lines by setting the **nolineifempty** property for the field. For example, if your label includes two lines for the address you can set the **nolineifempty** property to kTrue for the second address field. In this case if the second address line is empty for a particular record, the line is not printed and subsequent fields move up one line. If any address field on your label is likely to be empty you ought to set its **nolineifmpty** property.

Using a Calculated Field

Rather than putting two separate fields on your label report for the Firstname and Lastname data, you can use a single calculated field and the *con()* function.

To create a calculated field

1. Create a field on your report and view its properties
2. Leave the **$dataname** property empty, and set the **$calculated** property to kTrue
3. Enter the calculation in the **text** property, something like the following

###### con(CU\_FNAME,’ ‘,CU\_LNAME) ## note space char is in quotes

The *con()* function concatenates the current values in the CU\_FNAME and CU\_LNAME fields and separates them with a single space character.

Using all the features described in this section, your label report should look something like the following when printed to the screen.

## HTML Report Device

The Omnis Studio Print Manager API has been made public, allowing you to create your own custom printing devices as external components and place them in the XCOMP folder. You can show your own custom printing devices in the Print Destination dialog, and use the printing preferences and notation to control your own devices. The HTML printing device is an external component and shows what you can do with custom devices. You can use the HTML report device in exactly the same way as the standard report destinations; there is no difference between internal and external output devices.

When the HTML external component is loaded in Omnis, it registers an external output device with the Omnis Studio Print Manager and shows the HTML icon in the Report Destination dialog. To print to the HTML output device you can set it up via the Report Destination dialog, or access it via the notation using the print device methods.

You can specify the HTML device using the notation as follows.

###### Calculate $cdevice as kDevHtml Calculate $cdevice as $devices.Html

You can also set an item reference to the HTML device:

###### Set reference myDevice to $devices.Html

The constant kDevHtml is supplied by the HTML component at registration together with some other constants.

### Setting the HTML Device Parameters

The HTML output device has several parameters which affect the overall appearance of the HTML document generated by the device. You can change some of these parameters in the Report Destination dialog and the notation, while some can be manipulated by the notation only. The HTML output uses UTF-8: if you use a template file, that must also be UTF-8 encoded.

The HTML device parameters are represented by constants which you can use in the notation. Some of them correspond to parameters in the Report Destination dialog.

|  |  |
| --- | --- |
| Constant | Description |
| kDevHtmlFileName | the pathname of the destination HTML file |
| kDevHtmlFont1 | largest point size which maps to HTML font size 1 |
| kDevHtmlFont2 | largest point size which maps to HTML font size 2 |
| kDevHtmlFont3 | largest point size which maps to HTML font size 3 |
| kDevHtmlFont4 | largest point size which maps to HTML font size 4 |
| kDevHtmlFont5 | largest point size which maps to HTML font size 5 |
| kDevHtmlFont6 | largest point size which maps to HTML font size 6 |
| kDevHtmlFont7 | largest point size which maps to HTML font size 7 |
| kDevHtmlImageBorder | whether JPEG images have a single pixel border |
| kDevHtmlUseRects | whether background rectangles are to be used to determine the background color of the HTML table cell which intersects the background rectangle |
| kDevHtmlBackcolor | background color of the HTML document |
| kDevHtmlTextcolor | default text color; any black text received from the print manager will be changed to this color |
| kDevHtmlLinkcolor | color for HTML text or pictures which are HTML links |
| kDevHtmlVLinkColor | color for links which have been visited |
| kDevHtmlALinkColor | color for links which are currently active |
| kDevHtmlTemplate | full path and file name of a template HTML file which must be UTF-8 encoded; it must already contain the basic framework for an HTML file, that is  <html><head>  <meta http-equiv="content-type" content="text/html; charset=UTF-8"></head> <BODY bgcolor= …etc…> </BODY> </HTML> |
| kDevHtmlTemplateChars | the place holder contained within the template file which marks the point at which the report output will be inserted into the template, that is, “$$$$” the template file must contain this text, that is  <html><head>  <meta http-equiv="content-type" content="text/html; charset=UTF-8"></head> <BODY bgcolor= …etc…> <p>$$$$</p> </BODY> </HTML> |
| kDevHtmlScaleFont… | an additional single font scale factor; the following constants are available  kDevHtmScaleFontNone: no scaling kDevHtmScaleFontVSmall: reduce HTML size by 2 kDevHtmScaleFontSmall: reduce HTML size by 1 kDevHtmScaleFontLarge: increase HTML size by 1 kDevHtmScaleFontVLarge: increase HTML size by 2 |

You can get and set the value of the device parameters using the following methods.

1. $getparam(param constant)  
   returns the value of the specified parameter
2. $setparam(param constant, value [,param constant, value, …] )  
   sets the value(s) of the specified parameter(s)

For example

###### Do $devices.Html.$setparam(KDevHtmlFont1,6,KDevHtmlFont2,8) Do $devices.Html.$getparam(kDevHtmlFileName) Returns MyPath

The following example, sets up a template file (which must be UTF-8 encoded):

###### Calculate lPath as 'C:\<path>\Template.htm'

###### Do $devices.HTML.$setparam(kDevHtmlTemplate,lPath)

###### Do $devices.HTML.$setparam(kDevHtmlTemplateChars,'$$$$')

The value of all device parameters is stored in the Omnis configuration file.

### Sending Text or Data

It is possible to send text or data to some internal and external devices. The HTML device supports both. You can use the methods $sendtext() and $senddata() to send text and data, respectively.

When sending text, the HTML output device surrounds the given text with the correct HTML syntax, that is, it places begin paragraph and end paragraph statements around the text. You can send text with more than one call to $sendtext(), but still have the text appear in one paragraph. To do this, specify kFalse for the line feed parameter of the $sendtext() method. The device buffers the text separately, before adding a single paragraph to the document when you call $sendtext() with the line feed parameter set to kTrue.

When sending data, the device writes the data directly to the current position in the HTML file without any modification.

You can send text or data between reports, but not during printing, that is, while a report is being printed calls to $sendtext() and $senddata() are ignored.

The following method uses $senddata() to send data to an HTML file.

###### Set reference myDevice to $devices.Html Calculate $cdevice as myDevice Do myDevice.$setparam(kDevHtmlFileName,”C:\Omnis\REPORT.HTM”) Do myDevice.$open() Returns ok If ok   Do myDevice.$senddata(myData1)   Set report name Report1   Print report   Do myDevice.$senddata(myData2)   Set report name Report2   Print report   Do myDevice.$senddata(myData3)   Do myDevice.$close() End If

### HTML Report Objects

HTML report objects are special objects that you can use to insert objects, such as other HTML documents, pictures, DLLs, or web site addresses, into your HTML reports. The HTML report objects are part of the HTML printing device and, when the report editor is the top window, they appear in the Component Store under the External Components button.

The HTML report objects have an $address property which you can set to the address or location of an HTML document, picture, DLL, or web site, for example, “results/result1.htm” or “http://www.rainingdata.com/”. The text can contain square bracket calculations, such as “www.[lvWebName]”.

If the HTML objects are printed to any other device other than the HTML device, they behave like their equivalent Omnis field types, a standard picture or text field.