Configurable POD Reports

Aptean Ltd

Contents

1 Configurable POD Reports

For each completed collection or delivery job, POC or POD reports may be produced from the system. These reports are now defined with configurable attributes, allowing reports to be designed and distributed to the customer without requiring code changes.

At times, very bespoke requirements for a report layout mean that a bespoke report format will be written. In the main, however, any new reports should be produced as configurable reports, adding all required new functionality to the generic report producer, ensuring that the report becomes more feature-rich in time.

Note: All new reports (that have no bespoke components) should be created using this report.

1.1 Overview

The report is reliant on EPOD_CONFIG_REPORT data table. This controls:

- The report title
- The HTML of the report, for each of the sections
  - Page header
  - Initial Header
  - Content
    - title row
    - row, indicating columns to be totalled (/T?) and formatted (/F??)
    - empty row
    - total row
  - Final footer
  - Page footer
  - Specific javascript functions to help with bespoke formatting of data.
- The page-specific styling CSS, for display and printing
- Parameters for:
  - the number of content lines per page of each type
  - the number of images displayed per row on the Images page. **Note:** This is deprecated - use styles to control this instead.
- Flags to control whether certain aspects are included in the report:
  - Alt rows - are alternate rows printed with a different format.
  - Inc Products - whether content rows are produced per product or container.
  - Inc Cancelled - whether cancelled products or containers are included.
  - Inc Images - whether an images page is included.
  - Email Images - whether the images page is included when emailing the PDF.
  - PDF Orientation - portrait or landscape.

1.2 Designing

Data Field	Description
ECR_CSS	The styling (minified CSS) controlling the general layout and format of each element on the report.
ECR_CSS_PRINT	The styling (minified CSS) controlling the additional requirements of layout and format for printing the report.
ECR_PAGE_HEADER	HTML for the page header, appearing once at the top of every page.
ECR_INITIAL_HEADER	HTML for the initial header, appearing once at the top of the first page only, under the page header.
ECR_CONTENT_TITLE	HTML for the title row of the details section of the report.
ECR_CONTENT_ROW	HTML for the content row of the details section of the report.
ECR_CONTENT_TOTAL	HTML for the summary row (if required) of the details section of the report.
ECR_CONTENT_EMPTY	HTML for the empty rows of the details section on each page of the report. If not provided, empty rows will not be generated.
ECR_FINAL_FOOTER	HTML for the final footer of the report, produced on the last page only, above the page footer.

Data Field	Description
ECR_PAGE_FOOTER	HTML for the page footer, appearing once at the top of every page.
ECR_FUNCTIONS	Javascript for custom (bespoke) functions, for bespoke formatting.

1.3 Embedding Data

All report sections are built from the HTML in the sections in the database, replacing data tags ([...]) with real data from a selection of tables.

Tables allowed:

Data Field	Description
EPOD_SITE	The site, typically the carrier executing the job or the depot.
EPOD_SITE_CUSTOMER	The customer address linked to the site ID.
EPOD_LOAD	The load on which the job was completed.
EPOD_USER	The user that is assigned to the load.
EPOD_VEHICLE	The vehicle that is assigned to the load
EPOD_JOB	The job itself.
EPOD_JOB_GROUPS	The Job Group of the job.
EPOD_JOB_GROUP_CUSTOMER	The customer address linked to the job group.
EPOD_CUSTOMER	The customer address of the job.
EPOD_JOB_ADDRESS	The job address. If not present, this will show the customer address.
EPOD_COLLECTION_JOB	The collection job associated to the job selected.
EPOD_COLLECTION_CUSTOMER	The customer address of the collection job.
EPOD_COLLECTION_JOB_ADDRESS	The job address of the collection job. If not present, this will show the customer address.
EPOD_DELIVERY_JOB	The delivery job associated to the job selected.
EPOD_DELIVERY_CUSTOMER	The customer address of the delivery job.
EPOD_DELIVERY_JOB_ADDRESS	The job address of the delivery job. If not present, this will show the customer address.
EPOD_PRODUCT	The products of the job.
EPOD_CONTAINER	The containers (items) of the job.
EPOD_SIGNATURE	Any pre-job signatures associated to the job.
EPOD_TNC	Any pre-job terms and conditions associated to the job.
LXJ_JOB_1	Related to Last 3 Jobs functionality.
LXJ_JOB_2	Related to Last 3 Jobs functionality.
LXJ_JOB_3	Related to Last 3 Jobs functionality.
LXJ_DELIVERY_JOB_1	Related to Last 3 Jobs functionality.
LXJ_DELIVERY_JOB_2	Related to Last 3 Jobs functionality.
LXJ_DELIVERY_JOB_3	Related to Last 3 Jobs functionality.
LXJ_PRODUCT_1	Related to Last 3 Jobs functionality.
LXJ_PRODUCT_2	Related to Last 3 Jobs functionality.
LXJ_PRODUCT_3	Related to Last 3 Jobs functionality.

Example: [EPOD_JOB.EPL_JOB_CODE]

Note: Regarding links between EPOD_JOB/EPOD_COLLECTION_JOB/EPOD_DELIVERY_JOB: ONLY EPOD_JOB connects to EPOD_CONTAINERS and EPOD_PRODUCTS

Special formatting is applied to:

- Date/Time fields have standard OBS formatting applied by default, although this can be overridden by the Format modifier e.g. [table.EPL_DATE/Fdd MMM yyyy] for explicit date formatting.
- Telephone Numbers - selects the indicated telephone number EPOD_CUSTOMER.EPL_TELEPHONE(0)
- UDF Fields - selects the UDField from the specified UDF e.g. EPOD_JOB.EPL_UDF_JOBDETS.WEIGHBRIDGE_NO (normal field) or

EPOD_JOB.EPL_UDF_JOBDETS.EPOD_JOB.TRAILER_ID (data-bound field).
    - Note that accessing an X type field doesn't return the value - there are many items. Therefore it returns formatted HTML of the checkbox items and labels, with the class UDFFieldX. Note you should not format these fields.
    - Note that the entire UDF form can now be accessed by simply referencing the UDF directly without a field i.e. EPOD_JOB.EPL_UDF_JOBDETS
- TNCs - gets the T&Cs (in either format). Fields other than labels (i.e. the basic text) can be removed by EPOD_JOB.EPL_TNCS(N).
    - If these are UDF TNCs, a specific field can be requested e.g. EPOD_JOB.EPL_TNCS.EMAIL_REQD.
    - Note that, for UDF T&Cs for pre-job signatures, they are accessible through EPOD_SIGNATURE.SIG_TNC_CHECKS.
    - Accessing an 'X' type UDF field does not return the value - there are many items. Therefore it returns formatted HTML of the checkbox items and labels, formatted with the class UDFFieldX. Note you should not format these fields.
- Signatures - automatically displays as No Image Found.
- Job Addresses - automatically uses Customer if the job address doesn't exist.
- Weight fields are automatically formatted correctly with 3 decimal places. This can be overridden with an explicit format.
- Description fields have new line and returns changes to HTML linebreak (BR) instructions.
- Status and Job Type automatically translate to descriptions. This can be overridden by a pipe-delimited set of descriptions in parenthesis following the field e.g. (Collecting|Delivering|Servicing).

Explicit field formatting is controlled with a /F modifier:

- N? - numeric no of digits or a pattern e.g. /FN##,##0.00;-##,##0.00;#.
- C? currency no of digits or a pattern e.g. /FC£##,##0.00;-£##,##0.00;#

Any other value is considered to be a date or time format e.g. dd/mm/yyyy HH:mm

Totals are indicated on fields with a /T? modifier, indicating which total this is. They are used with a special data tag [TOT?], identifying the total to use.

These may also be formatted i.e. [TOT0/FN2]

- Up to 8 totals may be specified in the sections, accessed using 0-7.
- TOT8 specifies the total product count.
- TOT9 specifies the total container count.
- The content section is always calculated first, which means the totals are available in the header and footer sections.

Calculations may be identified as follows: [CALC: ...], with the fields inside the calculation NOT in square barckets For example:

- [CALC: EPOD_PRODUCT.EPL_PRODUCT_QTY_ACTUAL * EPOD_PRODUCT.EPL_WEIGHT/FN3]

Note: The above is formatting the calculation to 3 decimal places.

Warning: You may include parentheses, BUT YOU MUST LEAVE SPACES AROUND THEM. For example:

- [CALC: EPOD_PRODUCT.EPL_PRODUCT_QTY_ACTUAL * (EPOD_PRODUCT.EPL_UNIT_PRICE + EPOD_PRODUCT.EPL_UNIT_VAT)/FN3]

Calculations can be used to make decisions based on values, using the IIF function call, for example:

- [CALC:IIF('EPOD_JOB.EPL_STATUS'='X','CANCELLED - EPOD_JOB.EPL_REASON_DESCRIPTION','')]

Note: Everything must be enclosed in strings, as these are substituted for literal values.

This example will control adding some text if the status code of the job is 'X'.

This can also be used to check multiple values, with logical OR and AND, for example:

- [CALC:IIF('EPOD_JOB_ADDRESS.EPL_ADDRESS_5'='GB' OR 'EPOD_JOB_ADDRESS.EPL_ADDRESS_5'='GBR','UKCA','CE')]

Note: When putting these in SQL packages to update or create a configurable report, the single quotes in the text must be doubled, for example:

- [CALC:IIF("EPOD_JOB.EPL_STATUS"="X","CANCELLED - EPOD_JOB.EPL_REASON_DESCRIPTION","")]

Information for the last jobs completed on the trailer are available through the following 'table' names:

- LXJ_JOB_1 - the collection job object for the first job (nearest to date that this job was completed). This object includes all details)containers, products, etc)
- LXJ_JOB_2
- LXJ_JOB_3 - the collection job object for the third job (furthest of the 3 away from the date this job was completed).
- LXJ_DELIVERY_JOB_1 - the delivery job linked to LXJ_JOB_1 by Job Code
- LXJ_DELIVERY_JOB_2
- LXJ_DELIVERY_JOB_3
- LXJ_PRODUCT_1 - the first product on LXJ_JOB_1
- LXJ_PRODUCT_2
- LXJ_PRODUCT_3

In order for this to be used, the jobs MUST have trailer ID entered against them, either set in Admin, or entered by the user against e load (through Load Start Metrics) and inherited onto the jobs.

The data can be accessed in exacty the same way as any other field on any other table. For example, accessing a UDF Field on the 2nd delivery job is as follows:

- [LXJ_DELIVERY_JOB_2.EPOD_UDF_JOBDETS.CLEANING]

Customers may now have any number of parameters attached to them. The customers retrieved in this report (through the EPOD_CUSTOMER DAL object) include a collection EPOD_ADM_PARAMETER_VALUES.

This is accessible through that base EPOD_CUSTOMER DAL objects created, using the field name EPOD_ADM_PARAMETER_VALUES. The data can be accessed in exacty the same way as any other field on any other table. For example, accessing a parameter "OPERATOR" collection customer is as follows:

- [EPOD_COLLECTION_CUSTOMER.EPOD_ADMIN_PARAMETER_VALUES.OPERATOR]

The process will find the matching parameter record and return the value.

Barcode printing may be added as in the following example:

- </p><script>JsBarcode('#barcode', "[EPOD_JOB.EPL_JOB_CODE]", {height: 40, displayValue: false});</script>

You can set many parameters of the barcode being printed - see the JSBarcode documentation for details

Reason Code Descriptions for the right level of reason code can be accessed directly using [table.EPL_REASON_DESCRIPTION]. Depending on the table selected and the status of the table, the right reason code will be decoded (i.e. JOB, DET or CLA).

No other types are currently supported.

The reason code and description can be returned delimited by "-" using [table.EPL_REASON_DESCRIPTION(Y)].

Note: If the item has been cancelled through the C-ePOD Admin console, the reason code is set to "AdminCancel". EPL_REASON_DESCRIPTION will return "Admin Cancellation" as the reason code.

Current Page and Page Count can be inserted with [RPTPAGE] and [RPTPAGES].

Current line number in the content/detail sections can be obtained using [LINE].

1.4 Custom (Bespoke) Functions

Although the configurable POD eport supports many functions, some are so bespoke as to not require supporting within the standard functions. In this case, custom javascript (jQuery) functions may be added to the report to modify the report after it has been created, for example:

- Extract portions of a particular field
- Apply unusual formatting

Furthermore, data replacements can be made in the same way as the report format as well, to further increase functionality. For example:

```
if('[EPOD_JOB_ADDRESS.EPL_ADDRESS_5]'.indexOf('GB')>0) {
  alert('ITS GB');
} else {
  alert('NOT GBR');
}
```

In this example, the code will replace the data tag with the actual value before executing the script.

When adding the elements, ensure that the element is classed with the name of the function you want to apply to that element, or a common class to all functions that need this applying.

Add the functions to amend items of those classes.

When the report loads, the functions are run and the changes applied.

Large-scale modifications to layout can be achieved easily this way.

Entire sections can be altered based on data value by applying the class based on the data value itself.

To use automatically and have the functions applied at document ready:

- Do not include the <script> tags.
- Do not include the jQuery document ready calls.

Alternatively, include all code, including the HTML <script> tags and all will be added and executed as coded.

Examples:

The following are examples of applying a format to every element classed as 'col1', and creating a barcode from an ID'd element:

```
function makeBold() {
    $(this).css('font-weight', 'bold');
}
function padValue() {
    $(this).text('000'+$(this).text());
}
$('.col1').each(function() {
    $(this).css('background-color', 'yellow');
});
$('.col1').each(makeBold);
$('.col1').each(padValue);
$('.col1').css('color', 'red');
var jobCode = $("#tdJobCode").text();
var barcode = JsBarcode('#barcode', jobCode,  {height: 40, displayValue: false});
```

These functions will be called on document ready.

The below code will achieve the same thing as the above, but in this case, the designer is in full control of when the code is run and how.

```
<script type="text/javascript">
```


```
    $(document).ready(function() {
        function makeBold() {
            $(this).css('font-weight', 'bold');
        }
        function padValue() {
            $(this).text('000'+$(this).text());
        }
        $('.col1').each(function() {
            $(this).css('background-color', 'yellow');
        });
        $('.col1').each(makeBold);
        $('.col1').each(padValue);
        $('.col1').css('color', 'red');
        var jobCode = $("#tdJobCode").text();
        var barcode = JsBarcode('#barcode', jobCode,  {height: 40, displayValue: false});
    });
</script>
```

1.5 Notes on Styling and Classes

In the general POD document, the designer is in control of what classes and styles are in use. Most of the POD notes created in the system use a standard class set to make changing POD notes as similar as possible, whilst also allowing great flexibility and cross-browser compatibility. Each POD report can add to these classes and styles.

Certain common elements can be controlled through use of standard styles and classes, whereas some styling depends on predefined class definitions. The following sections show how to achieve styling of these elements.

1.5.1 Coloured Elements based on Status Code

This is achieved through basic classes:

- Create classes for the specific status, for example status_X
- Apply the class to the element that requires the formatting, by adding the status to the class name, for example:

```
<span class="status_[EPOD_CONTAINER.EPL_STATUS_CODE]">MyText</span>
```

1.5.2 Styling the Photo Page

The photo page, however, is not able to be directly designed for the report. It is, however, very customizable.

The photo page can be basically changed from one photo per page to two using the setting against the report.

The designer now has far more control now through styling the classes.

The basic photo page class styling is already included in the aspx page. This can be over-ridden by the CSS in the report itself

The designer now has control over:

- Number of photos per row.
- Size of detail and image cells.
- The text included against the header, the job photos, the container photos and the product photos.
- The labels for all included text.

Note: The photo page exports perfectly in the PDF conversion, but does NOT work well when printing from Chrome.

Samples:

Displaying 1 photo per page

- .photo { width: 100%;}

- .photo {page-break-after: always;}
- .photoSection div:first-child {page-break-after: always !important;page-break-before: avoid !important;}

Displaying 2 photos per row

- .photo { width: 49.5%;}

Displaying 3 photos per row

- .photo { width: 33%;}

Displaying text above photo (rather than to the left)

- .photoDetails { width: 100%; display: block; float: none;}
- .photoImage { width: 100%; display: block; float: none;}

Throwing a page break after every even/3rd/4th photo

- .photoSection > div:nth-child(even) {page-break-after: always !important;}
- .photoSection > div:nth-child(3n+0) {page-break-after: always !important;}
- .photoSection > div:nth-child(4n+0) {page-break-after: always !important;}

1.5.3 UDF Form/Field Classes

UDF forms may be retrieved as a whole from the database (e.g. [EPOD_JOB.EPL_UDF_JOBSDETS]). These are strongly classed and ID'd. These classes and IDs can then be used by the designer to specifically format certain sections.

The following classes are available:

- .UDFForm {}
- .UDFForm > div {}
- .UDFField {}
- .UDFFieldL {}
- .UDFFieldT {}
- .UDFFieldN {}
- .UDFFieldB {}
- .UDFFieldX2 {}
- .UDFFieldTSC {}
- .UDFFieldDDL {}
- .UDFFieldX {}
- .UDFFieldA {}
- .UDFFieldTitle {}
- .UDFFieldValue {}
- .UDFFieldSubtext {}
- .UDFFieldPost {}
- .UDFFieldItem {}

Each field can be styled by ID or class, with class per field or class per field type.

Each element of a field type can be classed individually by chaining the styles. So, title can be styled generically through ".UDFFieldTitle", and specifically for Text Areas using ".UDFFieldA .UDFFieldTitle".

As forms are ID'd by the UDF type, each UDF form can be styled completely separately by chaining the the styles. So, a field in one form may have a black border, whilst a field in another may have a red border. This would be styled as follows:

```
div#UDFForm_Returns_Only .UDFField { border: solid black 1px;}

div#UDFForm_Difficult_Delivery_Disclaimer .UDFField { border: solid red 1px;}
```

This allows for generic styling of all fields uniformly, with exceptions for certain forms, field types or specific fields.

1.6 Setting Up the Report

Settings for the configurable report are stored on EPOD_CONFIG_REPORT and are as follows:

Data Field	Description
ECR_NAME	The name of the report.
ECR_ROWS_FIRST_PAGE	The number of detail rows on the first page of a multi-page report, including the title line.
ECR_MAX_ROWS_SINGLE_PAGE	The number of detail rows on a single page report, including the title line.
ECR_ROWS_CONTENT_PAGE	The number of detail rows on a (non-first or last) page of a multi-page report.
ECR_ROWS_LAST_PAGE	The number of detail rows on the last page of a multi-page report
ECR_ALT_ROWS_IND	Whether odd and even detail rows are formatted differently (1) or not (0).
ECR_INC_CANCELLED_IND	Controls whether to include cancelled details lines (containers or products) on the report. If set to 0, cancelled lines will not appear on the report in any circumstances. If set to 1, cancelled lines will appear on the report for completed jobs and cancelled jobs. If set to 2, cancelled items will appear on the report only if the job was cancelled. If set to 3, cancelled lines will always appear on the report for complete and cancelled jobs.
ECR_INC_IMAGES_IND	Whether to include images on the viewed report (1) or not (0). If value 2, images are produced, but the automatic email to the customer is not sent until all photos have been received.
ECR_EMAIL_IMAGES_IND	Whether to include images on the PDF report generated for emails (1) or not (0). Value 2 means images are required in a split file in the POD production only.
ECR_INC_PRODUCTS_IND	Whether Products are included on the report (1) or not (0).
ECR_PDF_ORIENT	The orientation of the report: (P)ortrait or (L)andscape.
ECR_CONTENT_SORT	The fields on which to sort detail (product or container) records, delimited by commas. For example: "EPOD_CONTAINER:EPL_CONTAINER_PACKAGE_DESC".
ECR_CONTENT_GROUP	The fields on which to group detail (product or container) records, delimited by commas. For example: "EPOD_CONTAINER:EPL_CONTAINER_PACKAGE_DESC".
ECR_SINGLE_HEADER	If set, only produces the fixed header on the first or single page.
ECR_SINGLE_FOOTER	If set, only produces the fixed footer on the last or single page.
ECR_PHOTOS_PER_ROW	How many columns are shown on the Images page per row. The default is 2. **Note:** This is deprecated - use styling instead.

When a report format has been created, an entry must be created on EPOD_LIST_ITEMS for the EPOD_LISTS ID that controls POD and POC report formats. This may differ per installed system.

EPOD_LIST_ITEMS fields:

- Description - The description of the report, to appear on the drop-down lists within the C-ePOD Admin system.
- Value - the ECR_ID of the report, in square brackets. So, if the ID is 17, the value here would be "[17]"