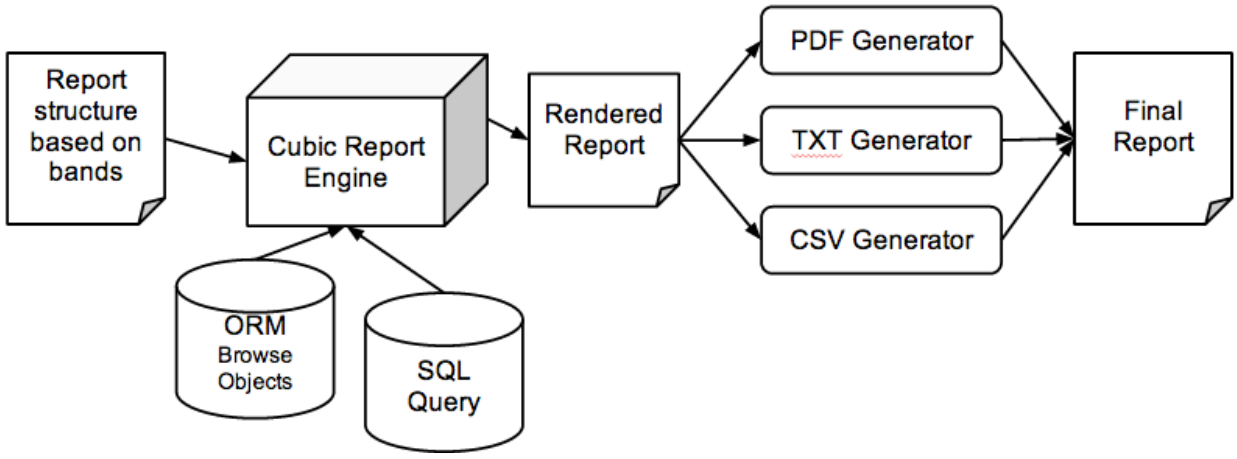
Cubic Report - Developer Guide

Table of Content

Introduction

CubicReport is a report engine fully integrated with OpenERP and based in Geraldo Reports and Report Labs libraries.



Features

- Reports based on SQL query or a Browse Object
- Company details header
- List and form mode
- Show report in menu checkbox
- Report params window
- A detail band, the report driver
- Page header, Page footer, Report begin and Report summary bands
- Grouping in multiple levels
- SubReports to print childs
- Multi-formats compatible in output (even if we only have PDF for a while)
- Graphic elements (including images and shapes)
- System fields (to show report title, page numbers, current date/time, etc.)
- Reports composition (you can reuse your bands for other reports)
- Reports inheritance (you can use the same structure for a lot of reports)
- A powerful stylizing support (colors, fonts, sizes, etc.)
- Different page sizes
- Flexibility to be easily customized
- Aggregation functions
- Hiding/showing bands and elements
- Events system

Installation

Install Cubic Reports

1. Install Python Library

```
$ sudo easy_install CubicReport
```

2. Download the module from https://www.openerp.com/apps/7.0/report_geraldo/
3. Install it from “Local Modules” menu in OpenERP

Install and patch Geraldo Reports Library (optional)

Ubuntu:

```
$ cd
$ git clone https://github.com/CubicERP/geraldo
$ sudo easy_install Geraldo
$ cd /usr/local/lib/python2.7/dist-packages/Geraldo-0.4.17-py2.7.egg
$ sudo mv geraldo geraldo.bk
$ sudo ln -s ~/geraldo/geraldo geraldo
```

CubicReport Structure

1. custom_style

This option is use to personalize the styles in the report

a. Default keys

1. fontName - Default: ‘Times-Roman’,
2. fontSize - Default: 10,
3. leading - Default: 12,
4. leftIndent - Default: 0,
5. rightIndent - Default: 0,
6. firstLineIndent - Default: 0,
7. alignment - Default: TA_LEFT,
8. spaceBefore - Default: 0,
9. spaceAfter - Default: 0,
10. bulletFontName - Default: ‘Times-Roman’,
11. bulletFontSize - Default: 10,
12. bulletIndent - Default: 0,
13. textColor - Default: black,
14. backColor - Default: None,
15. wordWrap - Default: None,
16. allowWidows - Default: 1,
17. allowOrphans - Default: 0,

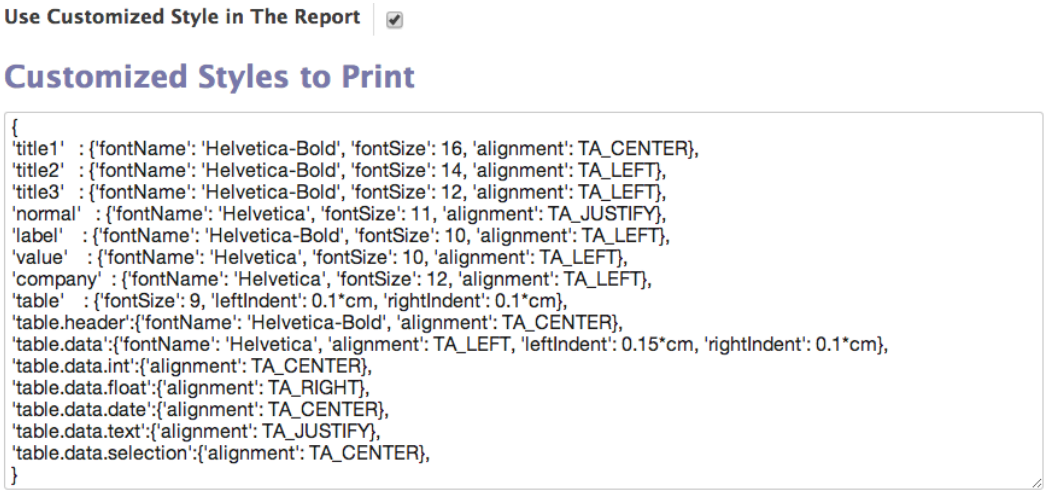
b. Predefined styles

1. 'title1' :
   {'fontName': 'Helvetica-Bold', 'fontSize': 16, 'alignment': TA_CENTER},
2. 'title2' :
   {'fontName': 'Helvetica-Bold', 'fontSize': 14, 'alignment': TA_LEFT},
3. 'title3' :
   {'fontName': 'Helvetica-Bold', 'fontSize': 12, 'alignment': TA_LEFT},
4. 'normal' :
   {'fontName': 'Helvetica', 'fontSize': 11, 'alignment': TA_JUSTIFY},
5. 'label' :
   {'fontName': 'Helvetica-Bold', 'fontSize': 10, 'alignment': TA_LEFT},
6. 'value' :
   {'fontName': 'Helvetica', 'fontSize': 10, 'alignment': TA_LEFT},
7. 'company' :
   {'fontName': 'Helvetica', 'fontSize': 12, 'alignment': TA_LEFT},
8. 'table' : {'fontSize': 9, 'leftIndent': 0.1*cm, 'rightIndent': 0.1*cm},
9. 'table.header':
   {'fontName': 'Helvetica-Bold', 'alignment': TA_CENTER},
10. 'table.data':
    {'fontName': 'Helvetica', 'alignment': TA_LEFT, 'leftIndent': 0.15*cm,
    'rightIndent': 0.1*cm},
11. 'table.data.int':
    {'alignment': TA_CENTER},
12. 'table.data.float':
    {'alignment': TA_RIGHT},
13. 'table.data.date':
    {'alignment': TA_CENTER},
14. 'table.data.text':
    {'alignment': TA_JUSTIFY},
15. 'table.data.selection':
    {'alignment': TA_CENTER},
16. 'group_by':
    {'fontName': 'Helvetica-Bold', 'fontSize': 12, 'alignment': TA_LEFT},
17. 'group_by.level_1':
    {'fontName': 'Helvetica-Bold', 'fontSize': 13, 'leftIndent': 0.4*cm},
18. 'group_by.level_2':
    {'fontName': 'Helvetica-Bold', 'fontSize': 12, 'leftIndent': 0.8*cm},
19. 'group_by.level_3':
    {'fontName': 'Helvetica-Bold', 'fontSize': 11, 'leftIndent': 1.2*cm},
20. 'group_by.level_4':
    {'fontName': 'Helvetica-Bold', 'fontSize': 10, 'leftIndent': 1.6*cm},

c. How to customize the styles

Under “Cubic Report” page check on “Use Customized Style in The Report”

d. Customize Style Example on GUI

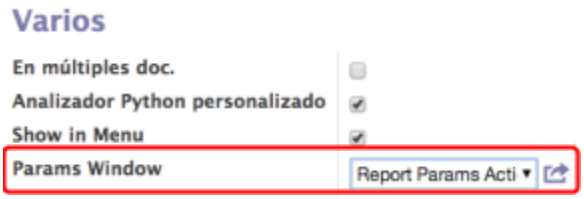


2. custom_params

Used to show a window to input some data required to filter the records of the report. This parameters would be retrieved on next ways:

- On custom_domain fields: use `%(param_name)s` as value on domain. Example: `[(‘date’,’>’,'%(date_ini)s'),(‘date’,’<’,'%(date_end)s')]`
- On custom_sql fields: use `%(param_name)s` as value in a query definition. Example:
  `...between '%(date_ini)s' and '%(date_end)s'...`
- In the reports, these params are charged on the context var using the prefix “params_”. Example:
  `{'type':'label', 'text':context.get('params_date_ini')}`
- In system fields, these params would be accessed using the expression %(var:VARIABLE_NAME)s. Example:
  `{'type':’system’, 'expression':'%(var:date_ini)s'}`

a. Custom Params Example on GUI



Customized Report Params

```
[
{'name': 'date_ini', 'type': 'date', 'args': ['Start Date'], 'kwargs':{}, 'default':time.strftime('%Y-%m-%d'), 'string':'Start Date', 'required': '1'},
{'name': 'date_end', 'type': 'date', 'args': ['End Date'], 'kwargs':{}, 'default':time.strftime('%Y-%m-%d'), 'string': 'End Date', 'required': '1'},
]
```

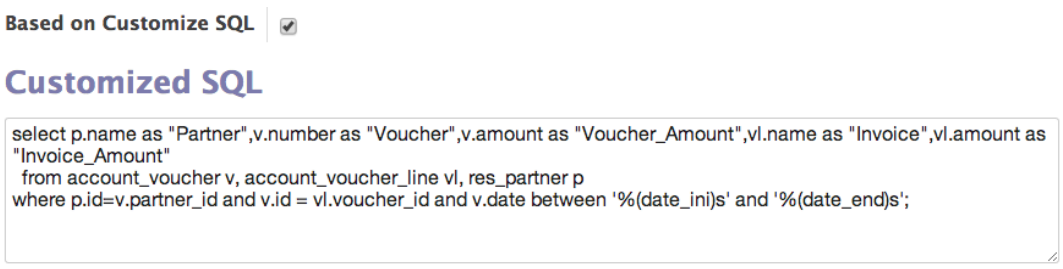
b. Custom Params Example on XML File

```
 <record id="action_account_voucher_invoice_report"
            model="ir.actions.act_window">
      <field name="name">Voucher with Invoices</field>
      <field name="res_model">base.report.params</field>
      <field name="view_type">form</field>
      <field name="view_mode">form</field>
      <field name="target">new</field>
 </record>
...
      <field name="act_window_id"
             ref="action_account_voucher_invoice_report"/>
      <field name="custom_params_src">[
                  {'name': 'date_ini', 'type': 'date',
                   'args': ['Start Date'], 'kwargs':{},
                   'default':time.strftime('%Y-%m-%d'),
                   'string':'Start Date', 'required': '1'},
                  {'name': 'date_end', 'type': 'date',
                   'args': ['End Date'], 'kwargs':{},
                   'default':time.strftime('%Y-%m-%d'),
                   'string': 'End Date', 'required': '1'},
                  {'name': 'partner_id', 'type': 'many2one',
                   'args': ['res.partner'], 'kwargs':{},
                   'relation':'res.partner', 'string': 'Partner',
                   'required': '0'},
      ]</field>
...
```

3. custom_sql

Used to generate reports based on a SQL sentence

a. Custom SQL Example on GUI



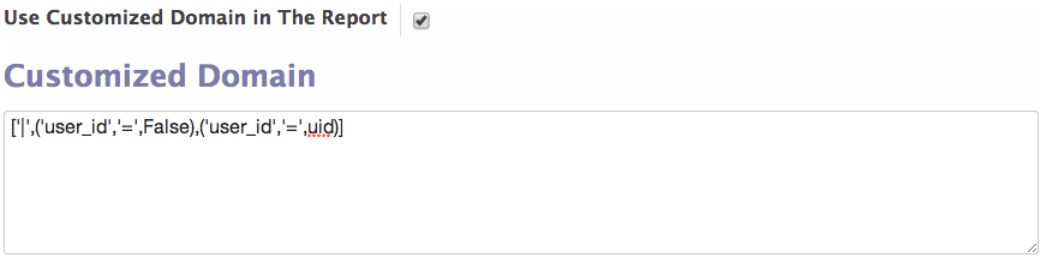
b. Custom SQL Example on XML File

```
<field name="custom_sql" eval="1"/>
<field name="custom_sql_src">
      select p.name as "Partner",v.number as "Voucher",v.amount
      as "Voucher_Amount",vl.name as "Invoice",vl.amount as
      "Invoice_Amount", v.date as "Voucher_Date"
        from account_voucher v, account_voucher_line vl,
      res_partner p
      where p.id=v.partner_id and v.id = vl.voucher_id and v.date
      between '%(date_ini)s' and '%(date_end)s';
</field>
```

4. custom_domain

Used to add a domain to filter the records when the report is based on browse records

a. Custom Domain Example on GUI



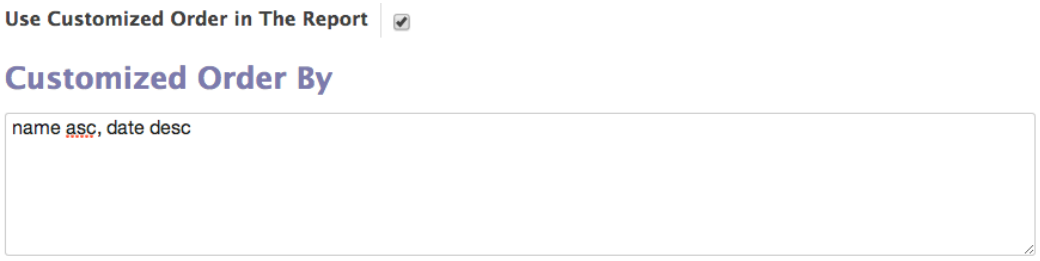
b. Custom Domain Example on XML File

```
<field name="custom_domain" eval="1"/>
<field name="custom_domain_src">[
      '|',('user_id','=',False),
      ('user_id','=',uid)
]</field>
```

5. custom_order

Used to order the records when the report is based on browse records

a. Custom Order Example on GUI



b. Custom Order Example on XML File

```
<field name="custom_order" eval="1"/>
<field name="custom_order_src">name asc, date desc</field>
```

6. custom_fields
    a. page_header
    b. begin
    c. detail
    d. summary
    e. subreports
        1. name
        2. queryset_string
        3. begin
        4. detail
        5. summary
        6. attributes
            1. subreport
            2. title
            3. begin
               Attributes of CubicReportBandBeginContent (CubicReportBand)
            4. begin.parent
               Attributes of CubicReportBandBegin (CubicReportBand)

5. detail
6. detail.parent
7. summary
8. summary.parent

f. **groups**

g. **page_footer**

7. **custom_class**
8. **custom_attributes**

CubicReport Widgets

1. **label - CubicLabel (Label, CubicWidget)**
   A label is just a simple text.

   a. **Atributes**
      1. **text** - Required
      2. **height** - Default: 0.5*cm
      3. **width** - Default: 5*cm
      4. **left** - Default: 0
      5. **top** - Default: 0
      6. **visible** - Default: True
      7. **style** - Default: table.header
      8. **get_value** - This function must have a ‘text’ argument.

   b. **label Example:**
   {'type': 'label', 'text': 'Product Details', 'style': 'title2'},
   {'type':'label', 'text':s.browse('res.users',uid).company_id.name},

2. **value - CubicObjectValue(ObjectValue, CubicWidget)**
   This shows the value from a method, field or property from objects in the queryset.
   You can provide an action to show the object value or an aggregation function on it.
   You can also use the ‘display_format’ attribute to set a friendly string formatting, with a mask or additional text.
   ‘get_value’ lambda must have an ‘instance’ argument.

   a. **Attributes**
      1. **name** - Required
         You can provide here the object attribute name you want to show in this

widget. You can provide a field, common attribute, property or just a method, since it has no required argument to provide.
You can use paths of callable or properties instead of attribute name in this attribute.

2. **height** - Default: 0.5*cm
3. **width** - Default: 5*cm
4. **left** - Default: 0
5. **top** - Default: 0
6. **visible** - Default: True
7. **style** - Defaults:
    1. if the value is int : table.data.int
    2. if the value is float : table.data.float
    3. if the value is date or datetime : table.data.date
    4. if the value is text : table.data.text
    5. if the value is selection : table.data.selection
    6. other values : table.data
8. **action** - Default: FIELD_ACTION_VALUE
   The action is where you say what Geraldo will do with the value of this field. The default action is just to show its value and nothing more.
   But if you are using an ObjectValue in a summary or footer band of report, group or subreport, you will probably have a need for aggregation functions. This is why this attribute exists.
   The choices you can use in action attributes are:
    1. FIELD_ACTION_VALUE
    2. FIELD_ACTION_COUNT
    3. FIELD_ACTION_AVG
    4. FIELD_ACTION_MIN
    5. FIELD_ACTION_MAX
    6. FIELD_ACTION_SUM
    7. FIELD_ACTION_DISTINCT_COUNT
9. **display_format** - Default: '%s'
   Use simple string formatting on the field value. The sintax supported is described on the next links:
   https://docs.python.org/2/library/string.html#formatstrings and
   https://docs.python.org/2/library/stdtypes.html#string-formatting
   You could otherwise use get_value if you want something more powerful.
10. **get_text** - Default: None.
    This 'lambda' attribute is seemed to get_value because it works after value getting. It's important to keep aware there are two moments from get a value from instance until to render it on the output generation:
    1. First the value is getted (and keeps being an unformatted value)
    2. After this, the value is formatted (and transformed to unicode string)

11. **stores_text_in_cache** - Default: True
    This is a boolean attribute you can set to False if you to force the text getting to run twice: on renderizing and on generating. If you keep it as True, it will store the text on the first getting and use the stored text on next time(s) it be requested.
12. **customCode** - Default: None
13. **customCodeMethod** - Default: None
14. **attribute_src** - Default: None
15. **value_prefix** - Default: ‘’
16. **value_suffix** - Default: ‘’

b. Events

1. **on_expression_error** - Default: None
   Is called when Geraldo tries to interpret an expression and an exception is raised. It must be a function with the following arguments:
   1. widget (or just ‘self’)
   2. exception (or just ‘e’)
   3. expression (or just ‘expr’)
   4. instance (or just ‘inst’)

   This function can returns a default value or just call ‘raise’ to raise the same exception.

c. value Example:

{'type':'value', 'name':'name', 'value_prefix':"Purchase Order - %(field)s: ", 'top':0.5*cm, 'left':0*cm, 'width':BAND_WIDTH, 'style':'Title1'}

{'type':'value', 'name':'qty', 'top':0.5*cm, 'left':0*cm, 'width':BAND_WIDTH, ‘get_text’: lambda s,i,v: float(v)<0 and "(%s)"%("{:,.2f}".format(float(v))) or "{:,.2f}".format(float(v)),
'action': FIELD_ACTION_SUM, 'style':'value'}

3. field = label + value

a. Attributes

1. **name** - Required
2. **string** - Default: name
3. **insert** - Default: False
   A list of dictionaries defining new fields to insert before this field
4. **nolabel** - Default: False
5. **style** - Defaults:
   1. for the label : label
   2. for the value : value

6. label.* - label attributes
   Use this notation to modify the attributes of the label
7. value.* - value attributes
   Use this notation to modify the attributes of the value

b. **field Example:**

```
{'type': 'field', 'name': 'requisition_id.account_analytic_id',
        'label.borders': {'top': True, 'bottom': True},
        'value.borders': {'top': True},
        'value.style': {'fontName': 'Helvetica', 'fontSize': 9,
                        'alignment': TA_CENTER},
        'label.style': {'fontName': 'Helvetica-Bold', 'fontSize': 9,
                        'alignment': TA_CENTER,
                        'backColor':HexColor(0xE0E0E0)},
},
{'type': 'field', 'name': 'requisition_id.amount',
        'value.display_format: '%.4f',
},
{'type': 'field', 'name': 'requisition_id.amount',
        'value.display_format: '{:,.4f}',
},
```

4. dummy - CubicLabel (Label, CubicWidget)

This print a label with a blank space

a. **Attributes**

1. **height** - Default: 0.5*cm
2. **width** - Default: 5*cm
3. **left** - Default: 0
4. **top** - Default: 0
5. **visible** - Default: True
6. **style** - Default: table.header
7. **get_value** - This function must have a 'text' argument.

b. **dummy Example:**

```
{'type': 'dummy', 'colSpan': 2},
```

5. system - CubicSystemField (SystemField, CubicWidget)

This shows system information, like the report title, current date/time, page number, page count, etc.
'get_value' must have 'expression' and 'fields' arguments.

a. **Attributes**

1. **expresion** - Required

This is a simple string you can write with basic macros to show system field values.
The available macros are:

1. %(report_title)s
2. %(page_number)s
3. %(first_page_number)s
4. %(last_page_number)s
5. %(page_count)s
6. %(report_author)s
7. %(now:FORMAT)s - in replace of 'FORMAT' you can use date formatting standard template filter date formatting. But if you are using your own Report.format_date method, this will use it.
   Examples:
   a. '%(now:%Y)' - shows the current year
   b. '%(now:%m/%d/%Y)' - shows something like '01/23/2009'
8. %(var:VARIABLE_NAME)s - you can inform a variable here and declare it when call generator below "Customized Report Params"

2. **width** - Default: 5*cm
3. **left** - Default: 0
4. **top** - Default: 0
5. **visible** - Default: True
6. **style** - Default: {}

b. **system - Example:**

{'type':'system', 'expression':'Pag. %(page_number)s/%(page_count)s'}

6. code - CubicObjectValue (ObjectValue, CubicWidget)

a. Attributes

1. **name** - Required
2. **src** - Required
   python source code to execute, the global variables are:
   1. cr : Cursor to the database
   2. uid : ID of the current user
   3. context : Context variable
   4. cb : CubicReport variable
   5. obj : Browse object or current record
   6. result : Value to return
3. **customCode** - Default True
4. **customCodeMethod** - Default: exec
   Method to execute the custom code, support **eval** and **exec**
5. **height** - Default: 0.5*cm
6. **width** - Default: 5*cm
7. **left** - Default: 0

8. **top** - Default: 0
9. **visible** - Default: True
10. **style** - Defaults:
    1. if the value is int : table.data.int
    2. if the value is float : table.data.float
    3. if the value is date or datetime : table.data.date
    4. if the value is text : table.data.text
    5. if the value is selection : table.data.selection
    6. other values : table.data
11. **action** - Default: FIELD_ACTION_VALUE
    The action is where you say what Geraldo will do with the value of this field. The default action is just to show its value and nothing more.
    But if you are using an ObjectValue in a summary or footer band of report, group or subreport, you will probably have a need for aggregation functions. This is why this attribute exists.
    The choices you can use in action attributes are:
    1. FIELD_ACTION_VALUE
    2. FIELD_ACTION_COUNT
    3. FIELD_ACTION_AVG
    4. FIELD_ACTION_MIN
    5. FIELD_ACTION_MAX
    6. FIELD_ACTION_SUM
    7. FIELD_ACTION_DISTINCT_COUNT
12. **display_format** - Default: '%s'
    Use simple string formatting on the field value. You could otherwise use get_value if you want something more powerful.
13. **get_text** - Default: None.
    This 'lambda' attribute is seemed to get_value because it works after value getting. It's important to keep aware there are two moments from get a value from instance until to render it on the output generation:
    1. First the value is getted (and keeps being an unformatted value)
    2. After this, the value is formatted (and transformed to unicode string)
14. **stores_text_in_cache** - Default: True
    This is a boolean attribute you can set to False if you to force the text getting to run twice: on renderizing and on generating. If you keep it as True, it will store the text on the first getting and use the stored text on next time(s) it be requested.
15. **src or attribute_src** - Default: None
16. **eval** - Default: None
    You can use src or eval

b. code - Example:

```
{'type': 'code', 'name': 'company_addres', 'colSpan': 4,
        'src': """# obj: Browse object or current record
                # result: Value to return
                result = '', ''.join([obj.company_id.street or '',
                        obj.company_id.street2 or '',
                        obj.company_id.city or '',
                        obj.company_id.state_id.name or '',
                        obj.company_id.country_id.name or '']
                )
                """
},
{'type': 'code', 'name': currency, 'eval':"obj.currency_id.name"},
{'type': 'code', 'name': 'amount',
        'eval':"pool.get('ir.translation').amount_to_text(obj.amount_total, 'pe',
                                                obj.currency_id.name)"},
```

7. image - CubicImage (Image, CubicGraphic)

a. Attributes

1. **filename** - Required
   You can provide a filename path or a Python Imaging Library Image instance. If the latter, then you have to have this library installed.
2. **file** - Required
3. **left** - Required
4. **width** - Required
5. **top** - Required
6. **height** - Required
7. **get_image** - Default: None
   You should provide a function or lambda object to this attribute when you want to work with Images or Charts based on object values or logic.

b. image - Example:

```
{'type':'image', 'file':s.browse('res.users',uid).company_id.logo,
                'left':0*cm, 'top':0*cm, 'right':4*cm, 'bottom':0.5*cm,
                'style':{'alignment':TA_LEFT}
            },
```

8. **rect - CubicRect (Rect, CubicGraphic)**

    a. **Attributes**

        1. **left** - Required
        2. **width** - Required
        3. **top** - Required
        4. **height** - Required

    b. **rect - Example**

```
{'type':'rect', 'top':0.3*cm, 'left':0.00,
        'width':BAND_WIDTH, 'height':0.85*cm,
        'fill':True, 'fill_color':HexColor(0xE0E0E0),
        'stroke': False,
    },
```

CubicReport Objects

1. CubicWidget (Widget)

Base class for reports widgets, you should use it only to inherit and make your own widgets, otherwise you shouldn't use it directly.

a. **Attributes**

1. **name** - Default: None
2. **height** - Default: 0.65*cm
3. **width** - Default: 5*cm
4. **left** - Default: 0
5. **top** - Default: 0
6. **visible** - Default: True
   Set to False if you want to make it not visible.
7. **borders** -
   Default: {'top': None, 'right': None, 'bottom': None, 'left': None, 'all': None}
   Borders values can be of three types:
   - Boolean (True/False) -
     just set if there is a border or not
   - Integer -
     set there is a border and what is its stroke width.
   - Graphic (instance of Rect, Line, RoundRect, etc.) -
     set a complex graphic to put along the border
8. **style** - Default: {}
   This is a dictionary that uses the powerful ReportLab ParagraphStyle class to set style settings for this widget.
   Default keys are:
   - fontName - Default: 'Times-Roman',

- fontSize - Default: 10,
- leading - Default: 12,
- leftIndent - Default: 0,
- rightIndent - Default: 0,
- firstLineIndent - Default: 0,
- alignment - Default: TA_LEFT,
- spaceBefore - Default: 0,
- spaceAfter - Default: 0,
- bulletFontName - Default: 'Times-Roman',
- bulletFontSize - Default: 10,
- bulletIndent - Default: 0,
- textColor - Default: black,
- backColor - Default: None,
- wordWrap - Default: None,
- allowWidows - Default: 1,
- allowOrphans - Default: 0,

9. **get_value** - Default: None.
   This is the way you can easily customize the widget output. Be careful with this attribute, because each widget has its own set of arguments.
10. **truncate_overflow** - Default: False
    When "True", truncate the widget to use only its defined height and widget attributes, with no word wrap. This means that those attributes are required.
11. **colWidth** - Default: None
12. **colSpan** - Default: None

b. Events

1. **before_print** - Default: None
   Is called by do_before_print method, before generate the widget output. Expect arguments widget and generator.
2. **after_print** - Default: None
   Is called by do_after_print method, after generate the widget output. Expect arguments widget and generator.

c. Rendering Attributes

They are read-only attributes you can use at render time.

1. **instance** - current object being rendered
2. **generator** - generator instance
3. **report** - report instance this element is in
4. **band** - band this element is in
5. **page** - current page

2. CubicGraphic (Graphic):

This is the basic graphic class. You should use it only when inheriting to create your own graphic class, never use it directly.
Its rect area is based on left-width-top-height dimensions.

a. Attributes

1. **name** - Default: None
2. **visible** - Default: True
   Set to False if you want to make it not visible.
3. **stroke** - Default: True
4. **stroke_color** - Default: reportlab.lib.colors.black
5. **stroke_width** - Default: 1
6. **fill** - Default: False
7. **fill_color** - Default: reportlab.lib.colors.black

b. Rendering Attributes

They are read-only attributes you can use at render time.

1. **instance** - current object being rendered
2. **generator** - generator instance
3. **report** - report instance this element is in
4. **band** - band this element is in
5. **page** - current page

c. Events

1. **before_print** - Default: None
   Is called by do_before_print method, before generate the graphic output. Expect arguments graphic and generator.
2. **after_print** - Default: None
   Is called by do_after_print method, after generate the graphic output. Expect arguments graphic and generator.

d. Methods

1. **set_rect**(**kwargs)
   Used to generate a hard rectangle when rendering pages. Override it to set your own rule if you need.

3. CubicReportBand (ReportBand)

A band is a horizontal area in the report. It can be used to print things on the top, on summary, on page header, on page footer or one time per object from queryset.

a. Attributes

1. **name** - Default: None
2. **height** - Default: 1*cm
3. **width** - Default: None
4. **visible** - Default: True

Set to False if you want to make it not visible.

5. **visible_force** - Default: None
   Force the visible attribute
6. **borders** -
   Default: {'top': None, 'right': None, 'bottom': None, 'left': None, 'all': None}
   Borders values can be of three types:
   - Boolean (True/False) - just set if there is a border or not
   - Integer -
     set there is a border and what is its stroke width. New on 0.4.
   - Graphic (instance of Rect, Line, RoundRect, etc.) -
     set a complex graphic to put along the border
7. **elements** - Default: []
8. **child_bands** - Default: []
9. **force_new_page** - Default: False
10. **default_style** - Default: None
11. **margin_top** - Default: 0
12. **margin_bottom** - Default: 0
13. **auto_expand_height** - Default: False
    Use 'auto_expand_height' to make flexible bands to fit their heights to their elements.
14. **cols** - Default: 4
    Define the number of columns used to print fields on form mode
15. **table_name** - Default: None
    This band is below to this table name. This attribute define a group of bands.
16. **top_position** - Default: None
    Used to print this band in a absolute top position, example: 10*cm
17. **_top_increment** - Default: 0.7*cm
    Used on automatical fields position, increments on top for form.
18. **_top_initial** - Default: 0.3*cm
    Used on automatical fields position, from this value begin the top format on fields
19. **_left_initial** - Default: 0.0*cm
    Used on automatical fields position, left initial
20. **_left_proportion_value** - Default: 2.0/3
    Used on automatical fields position
21. **_left_proportion_label** - Default: 1.0/3
    Used on automatical fields position

b. Attribute Key:

band

c. Methods:

1. **pool**(*args):
   Example: s.pool('res.partner')
2. **browse**(obj, ids, **kwargs):
   Example: s.browse('res.users',uid)
3. **search**(obj, arg):
4. **read**(obj, ids, fields):
5. **read_group**(obj, arg, fields, group):
6. **write**(obj, ids, arg):

4. CubicDetailBand (DetailBand, CubicReportBand)

An extension of ReportBand. The only difference is that this class supports more attributes, specific to detail bands.

a. Attributes:

1. **name** - Default: None
2. **margin_left** - Default: 0
3. **margin_right** - Default: 0
4. **display_inline** - Default: False
   When you use display_inline with True value and width with a valid value, the generator will try to align the detail band instances in the same way that HTML does with inline displaying of left floating elements: it will keep each detail band to the right of the last one if there is width available. This is useful for label reports.

b. Attribute Key:

detailBand

5. CubicReportBandCompany (CubicReportBand)

a. Attributes

1. **height** - Default: 0.0*cm
2. **visible** - Default: False
3. **auto_expand_height** - Default: True
4. **_top_increment** - Default: 0.6*cm

b. Attribute Key:

company

6. CubicReportBandTitle (CubicReportBand)

a. Attributes

1. **height** - Default: 1.6*cm

b. **Elements Key:**
title

c. **Attribute Key:**
title

7. **CubicReportBandBeginContent (CubicReportBand)**

a. **Attributes**
1. **height** - Default: 0.5*cm
2. **auto_expand_height** - Default: True

b. **Elements Key:**
begin

c. **Attribute Key:**
begin

8. **CubicReportBandDetailContent (CubicReportBand)**

a. **Attributes**
1. **height** - Default: 0.0*cm
2. **_top_initial** - Default: 0.07*cm
3. **auto_expand_height** - Default: True

b. **Elements Key:**
detail

c. **Attribute Key:**
detail

9. **CubicReportBandSummaryContent (CubicReportBand)**

a. **Elements Key:**
summary

b. **Attribute Key:**
summary

10. **CubicReportBandBegin (CubicReportBand)**

a. **Attributes**
1. **height** - Default: 0.0*cm

b. **child_bands**

1. CubicReportBandCompany (CubicReportBand)
This show only on tree mode and main table

2. CubicReportBandTitle (CubicReportBand)
   This show only on tree mode and main table
3. CubicReportBandBeginContent (CubicReportBand)
   This content the element widgets

c. **Attribute Key:**
   begin.parent

11. CubicReportBandDetail (CubicDetailBand)

a. **Attributes**
   1. **auto_expand_height** - Default: True
   2. **borders** - Default: {'bottom': True}

b. **child_bands**
   1. CubicReportBandCompany (CubicReportBand)
      This show only on form mode and main table
   2. CubicReportBandTitle (CubicReportBand)
      This show only on form mode and main table
   3. CubicReportBandDetailContent (CubicDetailBand)
      This content the element widgets

c. **Attribute Key:**
   detail.parent

12. CubicReportBandSummary (CubicReportBand)

a. **child_bands**
   1. CubicReportBandSummaryContent (CubicReportBand)
      This content the element widgets

b. **Attribute Key:**
   summary.parent

CubicReport Globals, Constants and Standards

1. Widget Globals

1.1. s : Self, a pointer to the current band (CubicReportBand)
1.2. cr : Cursor to database
1.3. uid : ID of the current user connected

1.4. context : Context variable

2. Alignment constants:

2.1. TA_LEFT, (0)
2.2. TA_CENTER or TA_CENTRE, (1)
2.3. TA_RIGHT, (2)
2.4. TA_JUSTIFY, (4)

3. The standard fonts:

3.1. Courier
3.2. Courier-Bold
3.3. Courier-BoldOblique
3.4. Courier-Oblique
3.5. Helvetica
3.6. Helvetica-Bold
3.7. Helvetica-BoldOblique
3.8. Helvetica-Oblique
3.9. Symbol
3.10. Times-Bold
3.11. Times-BoldItalic
3.12. Times-Italic
3.13. Times-Roman
3.14. ZapfDingbats

4. The standard Colors:

4.1. aliceblue = HexColor(0xF0F8FF)
4.2. antiquewhite = HexColor(0xFAEBD7)
4.3. aqua = HexColor(0x00FFFF)
4.4. aquamarine = HexColor(0x7FFFD4)
4.5. azure = HexColor(0xF0FFFF)
4.6. beige = HexColor(0xF5F5DC)
4.7. bisque = HexColor(0xFFE4C4)
4.8. black = HexColor(0x000000)
4.9. blanchedalmond = HexColor(0xFFEBCD)
4.10. blue = HexColor(0x0000FF)
4.11. blueviolet = HexColor(0x8A2BE2)
4.12. brown = HexColor(0xA52A2A)
4.13. burlywood = HexColor(0xDEB887)
4.14. cadetblue = HexColor(0x5F9EA0)
4.15. chartreuse = HexColor(0x7FFF00)
4.16. chocolate = HexColor(0xD2691E)
4.17. coral = HexColor(0xFF7F50)
4.18. cornflowerblue = cornflower = HexColor(0x6495ED)
4.19. cornsilk = HexColor(0xFFF8DC)
4.20. crimson = HexColor(0xDC143C)

4.21. cyan = HexColor(0x00FFFF)
4.22. darkblue = HexColor(0x00008B)
4.23. darkcyan = HexColor(0x008B8B)
4.24. darkgoldenrod = HexColor(0xB8860B)
4.25. darkgray = HexColor(0xA9A9A9)
4.26. darkgrey = darkgray
4.27. darkgreen = HexColor(0x006400)
4.28. darkkhaki = HexColor(0xBDB76B)
4.29. darkmagenta = HexColor(0x8B008B)
4.30. darkolivegreen = HexColor(0x556B2F)
4.31. darkorange = HexColor(0xFF8C00)
4.32. darkorchid = HexColor(0x9932CC)
4.33. darkred = HexColor(0x8B0000)
4.34. darksalmon = HexColor(0xE9967A)
4.35. darkseagreen = HexColor(0x8FBC8B)
4.36. darkslateblue = HexColor(0x483D8B)
4.37. darkslategray = HexColor(0x2F4F4F)
4.38. darkslategrey = darkslategray
4.39. darkturquoise = HexColor(0x00CED1)
4.40. darkviolet = HexColor(0x9400D3)
4.41. deeppink = HexColor(0xFF1493)
4.42. deepskyblue = HexColor(0x00BFFF)
4.43. dimgray = HexColor(0x696969)
4.44. dimgrey = dimgray
4.45. dodgerblue = HexColor(0x1E90FF)
4.46. firebrick = HexColor(0xB22222)
4.47. floralwhite = HexColor(0xFFFAF0)
4.48. forestgreen = HexColor(0x228B22)
4.49. fuchsia = HexColor(0xFF00FF)
4.50. gainsboro = HexColor(0xDCDCDC)
4.51. ghostwhite = HexColor(0xF8F8FF)
4.52. gold = HexColor(0xFFD700)
4.53. goldenrod = HexColor(0xDAA520)
4.54. gray = HexColor(0x808080)
4.55. grey = gray
4.56. green = HexColor(0x008000)
4.57. greenyellow = HexColor(0xADFF2F)
4.58. honeydew = HexColor(0xF0FFF0)
4.59. hotpink = HexColor(0xFF69B4)
4.60. indianred = HexColor(0xCD5C5C)
4.61. indigo = HexColor(0x4B0082)
4.62. ivory = HexColor(0xFFFFF0)
4.63. khaki = HexColor(0xF0E68C)

4.64. lavender = HexColor(0xE6E6FA)
4.65. lavenderblush = HexColor(0xFFF0F5)
4.66. lawngreen = HexColor(0x7CFC00)
4.67. lemonchiffon = HexColor(0xFFFACD)
4.68. lightblue = HexColor(0xADD8E6)
4.69. lightcoral = HexColor(0xF08080)
4.70. lightcyan = HexColor(0xE0FFFF)
4.71. lightgoldenrodyellow = HexColor(0xFAFAD2)
4.72. lightgreen = HexColor(0x90EE90)
4.73. lightgrey = HexColor(0xD3D3D3)
4.74. lightpink = HexColor(0xFFB6C1)
4.75. lightsalmon = HexColor(0xFFA07A)
4.76. lightseagreen = HexColor(0x20B2AA)
4.77. lightskyblue = HexColor(0x87CEFA)
4.78. lightslategray = HexColor(0x778899)
4.79. lightslategrey = lightslategray
4.80. lightsteelblue = HexColor(0xB0C4DE)
4.81. lightyellow = HexColor(0xFFFFE0)
4.82. lime = HexColor(0x00FF00)
4.83. limegreen = HexColor(0x32CD32)
4.84. linen = HexColor(0xFAF0E6)
4.85. magenta = HexColor(0xFF00FF)
4.86. maroon = HexColor(0x800000)
4.87. mediumaquamarine = HexColor(0x66CDAA)
4.88. mediumblue = HexColor(0x0000CD)
4.89. mediumorchid = HexColor(0xBA55D3)
4.90. mediumpurple = HexColor(0x9370DB)
4.91. mediumseagreen = HexColor(0x3CB371)
4.92. mediumslateblue = HexColor(0x7B68EE)
4.93. mediumspringgreen = HexColor(0x00FA9A)
4.94. mediumturquoise = HexColor(0x48D1CC)
4.95. mediumvioletred = HexColor(0xC71585)
4.96. midnightblue = HexColor(0x191970)
4.97. mintcream = HexColor(0xF5FFFA)
4.98. mistyrose = HexColor(0xFFE4E1)
4.99. moccasin = HexColor(0xFFE4B5)
4.100. navajowhite = HexColor(0xFFDEAD)
4.101. navy = HexColor(0x000080)
4.102. oldlace = HexColor(0xFDF5E6)
4.103. olive = HexColor(0x808000)
4.104. olivedrab = HexColor(0x6B8E23)
4.105. orange = HexColor(0xFFA500)
4.106. orangered = HexColor(0xFF4500)

4.107. orchid = HexColor(0xDA70D6)
4.108. palegoldenrod = HexColor(0xEEE8AA)
4.109. palegreen = HexColor(0x98FB98)
4.110. paleturquoise = HexColor(0xAFEEEE)
4.111. palevioletred = HexColor(0xDB7093)
4.112. papayawhip = HexColor(0xFFEFD5)
4.113. peachpuff = HexColor(0xFFDAB9)
4.114. peru = HexColor(0xCD853F)
4.115. pink = HexColor(0xFFC0CB)
4.116. plum = HexColor(0xDDA0DD)
4.117. powderblue = HexColor(0xB0E0E6)
4.118. purple = HexColor(0x800080)
4.119. red = HexColor(0xFF0000)
4.120. rosybrown = HexColor(0xBC8F8F)
4.121. royalblue = HexColor(0x4169E1)
4.122. saddlebrown = HexColor(0x8B4513)
4.123. salmon = HexColor(0xFA8072)
4.124. sandybrown = HexColor(0xF4A460)
4.125. seagreen = HexColor(0x2E8B57)
4.126. seashell = HexColor(0xFFF5EE)
4.127. sienna = HexColor(0xA0522D)
4.128. silver = HexColor(0xC0C0C0)
4.129. skyblue = HexColor(0x87CEEB)
4.130. slateblue = HexColor(0x6A5ACD)
4.131. slategray = HexColor(0x708090)
4.132. slategrey = slategray
4.133. snow = HexColor(0xFFFAFA)
4.134. springgreen = HexColor(0x00FF7F)
4.135. steelblue = HexColor(0x4682B4)
4.136. tan = HexColor(0xD2B48C)
4.137. teal = HexColor(0x008080)
4.138. thistle = HexColor(0xD8BFD8)
4.139. tomato = HexColor(0xFF6347)
4.140. turquoise = HexColor(0x40E0D0)
4.141. violet = HexColor(0xEE82EE)
4.142. wheat = HexColor(0xF5DEB3)
4.143. white = HexColor(0xFFFFFF)
4.144. whitesmoke = HexColor(0xF5F5F5)
4.145. yellow = HexColor(0xFFFF00)
4.146. yellowgreen = HexColor(0x9ACD32)
4.147. fidblue=HexColor(0x3366cc)
4.148. fidred=HexColor(0xcc0033)
4.149. fidlightblue=HexColor("#d6e0f5")