Wallboard Plugin Installation

You can install the plugin via the FOP2 Manager as any other plugin. But for it to work fully, you also must install Asternic Call Center Stats PRO, available at http://www.asternic.net/download.php (http://www.asternic.net/download.php).

The key factor is the queue_log parser named 'asterniclog' that will work wherever the software is licensed or not.

As such, there is no need to have an Asternic CCStats PRO license, but if you do have it licensed, then you will have some nicer calculations by having tracks of outbound queue calls and separate sla times per queue.

To install Asternic Call Center Stats PRO follow its installation guide (http://www.asternic.net/docs/#Gettingthesoftware)

After installing Asternic you will need to allow access to the CDR database to the MySQL user that connects to Asternic qstats database also, if you installed with default instructions, the following command will do the trick:

```
mysql –u root –p –e "grant select on asteriskcdrdb.* to
qstatsUser@localhost identified by 'qstatsPassw0rd'"
mysql –u root –p –e "grant select on asterisk.users to
qstatsUser@localhost identified by 'qstatsPassw0rd'"
```

Wallboard Plugin Configuration

The wallboard plugin is totally customizable, thanks to the power of html/css/javascript and the dustjs library/template system. This makes it extremely powerful, but also complex to setup. Here you will find a list of configuration options and variables, together with some samples, so you can try to adapt the wallboard to your liking/needs.

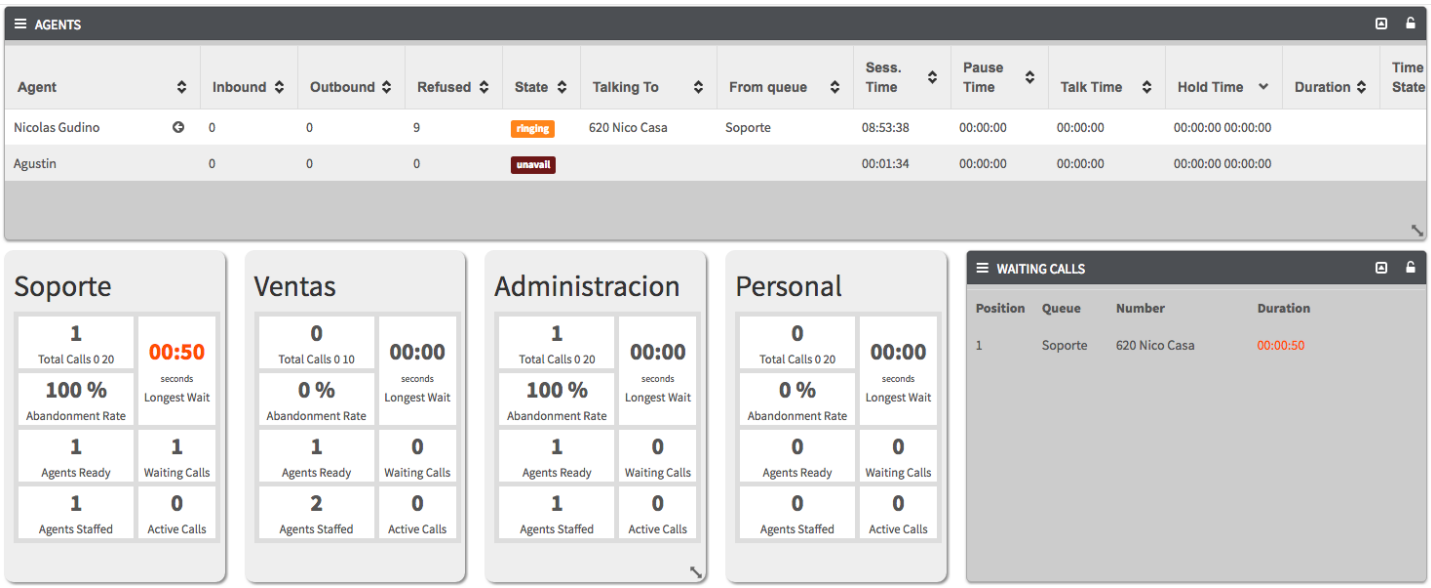
In order to configure or modify the wallboard configuration and templates, you must log into the FOP2 Manager, select the Plugins section, scroll and find the wallboard plugin and click on the *Settings* button to display its configuration form.

Sections

Wallboard has 3 main components, that are isolated from each other:

- Waiting Calls Widget
- Agent List Widget
- Queue Box Widgets

Here is a screen shot showing all sections in action:



The waiting calls and queue box widgets have one template.

The Agent Status has actually two templates to configure: the Agent List template, and the Agent Line template). The reason for this is twofold: to simplify its configuration as the template might become quite large and contain lots of data, also to make fast updates when a single agents change status, as you do not want the full agent widget to be rendered when only one agent needs to be updated. So, the Agent List template contains the whole table and iteration call for agents (it is basically a table header), and the Agent Line template is the individual agent row (a table row).

Settings

Agent List Template:

```
<table id='wbtableagents' class='tablesorter tablesorter-bootstrap table table-bordered table-striped'>
  <thead>
    <tr>
      <th style='width:200px;'><span class='wbAgent'>Agent</span></th>
      <th style='width:100px;'>Inbound</th>
      <th style='width:110px;'>Outbound</th>
      <th style='width:100px;'>Refused</th>
      <th style='width:80px;'><span class='wbState'>State</span></th>
      <th style='width:140px;'>Talking To</th>
      <th style='width:140px;'>From queue</th>
      <th style='width:100px;'>Sess. Time</th>
      <th style='width:100px;'>Pause Time</th>
      <th style='width:120px;'>Talk Time</th>
      <th style='width:120px;'>Hold Time</th>
      <th style='width:100px;' class='wbDuration'>Duration</th>
```

Agent Line Template:

```
<td class='col-md-2'>
  <div>
    <div class='pull-left'>{NAME}</div>
    <div class='pull-right'>
```

Templates

Templating is done using the Dustjs library http://www.dustjs.com/ (http://www.dustjs.com/)

This system lets you mix html with variables and some logic into it. This is a sample waitings call template:

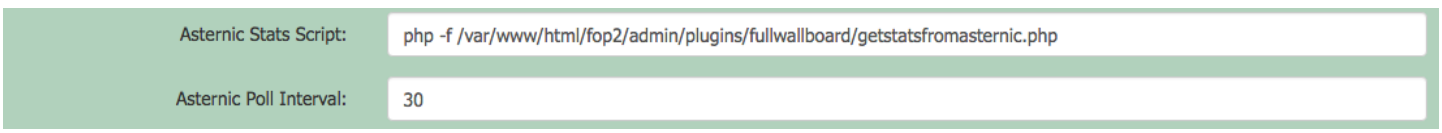
```
<table class='table table-bordered agent-table'>
  <thead>
    <tr>
      <th><span class='wbPosition'>Position</span></th>
      <th><span class='wbQueue'>Queue</span></th>
      <th class='wbNumber'>Number</th>
      <th class='wbDuration'>Duration</th>
    </tr>
  </thead>
  <tbody>
    {#calls}
    <tr>
      <td class='col-md-1'>{POSITION}</td>
      <td class='col-md-2'>{QUEUE}</td>
      <td class='col-md-4'>{CLIDNUM} {CLIDNAME}</td>
      <td class='col-md-5 blink_me_red'>{DURATION|s}</td>
    </tr>
    {/calls}
  </tbody>
</table>
<br/><br/>
```

As you can see, its straight html code with some variables being show. Variables are always indicated between curly brackets.

At time of rendering, variables will show the actual numbers on the system. Notice there is also an iteration section named {#calls} that ends with {/calls}, that will repeat for every call that is found to be waiting on your PBX.

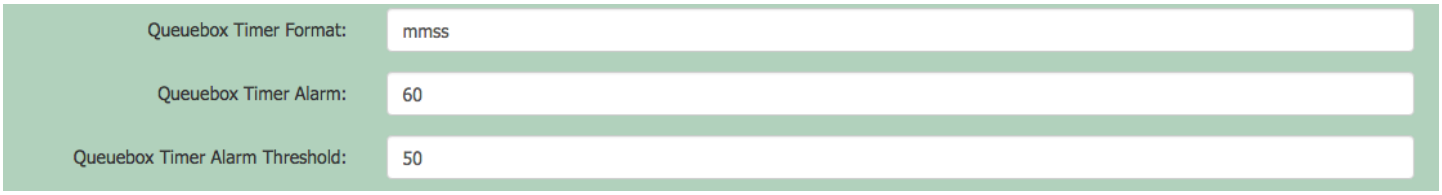
Wallboard Stats Script

In order to get statistical information, the wallboard plugin uses Asternic CCStats PRO database in order to acquire data and display it on its widgets. The plugin comes bundled with a script that will use Asternic configuration and database to generate the proper events to update the widgets. That script name is *getstatsfromasternic.php* located in the same directory the plugin is installed, usually /var/www/html/fop2/admin/plugins/fullwallboard. That is script is also run periodically to update information on widgets. By default it runs every 30 seconds. Both parameters (script and poll interval), can be changed via the plugin configuration form.



Queuebox Waiting Timer Alarm

In the Queuebox widget it is possible to define a timer alarm that will change the timer color from yellow to red depending on how long the caller is waiting.



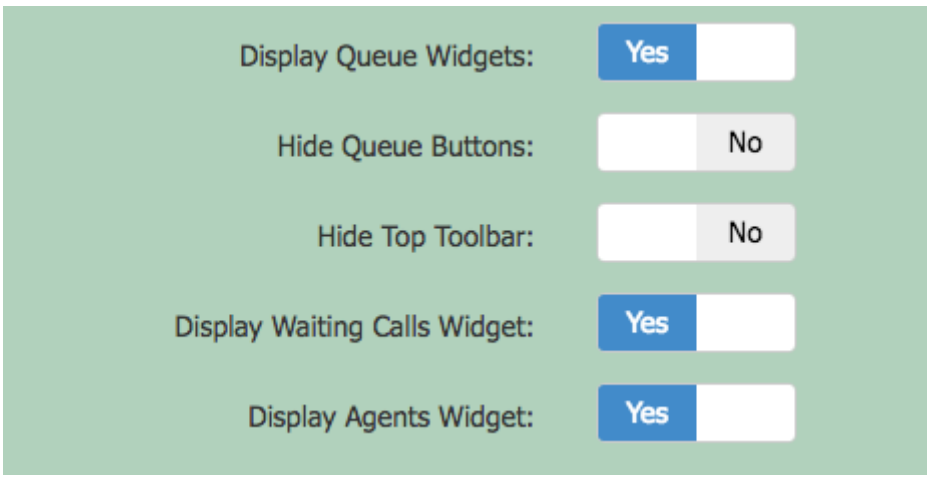
As you can see you have 3 options for fully configuring the alarm:

- Queuebox Timer Alarm (value in seconds the call is considered to be in red alarm)
- Queuebox Alarm Threshold (percentage of alarm on to which start coloring timer in yellow)
- Queuebox Timer Format (format to show the max waiting timer, possible values being hhmmss, mmss or ss)

In the above example, when a call reaches 60 seconds of waiting time it will be red colored, but at 50% that time (30 seconds) will be colored yellow and will be turning more redish when aproaching the alarm value of 60. The format for the waiting time will show only minutes and seconds, like 00:34.

Display Options

You also have some other options that affects the display and rendering of the panel when the wallboard is used.



You can chose to show or not any of the thre main sections (queue widgets, agent widget, waiting calls widget).

You can also chose to hide the normal queue buttons when the wallboard is active, and to hide the top toolbar (so you can really fill the screen width your widgets in any layout you want with no superfluos content).

Login into the wallboard automatically

FOP2 can be accessed directly with no authentication prompt if you pass the extension and password in the http request with the 'exten' and 'pass' options.

Imagine you want to display the wallboard and nothing but the wallboard showing information from 2 of all of the available queues, say the queue 'Support' and queue 'Sales'. In order to do so you will need to perform this steps:

- Create a Group: Select the Groups tab in the FOP2 Manager, create a new group with the name 'SupportAndSales' (or any other name you like) and selct only the Support and Sales queues from the list. Submit Changes.
- Create a User: Select the Users tab in the FOP2 Manager, create a new user, for example '1000' with password 'wallboard', and be sure to click the Full Wallboard plugin from the Plugins section, and select only the group 'SupportAndSales' in the Groups section for that user. Submit Changes.
- Relaod FOP2 via the Actions - Reload FOP2 menu.

Now point your browser to:

http://your.server/fop2/?exten=1000&pass=wallboard (http://your.server/fop2/?exten=1000&pass=wallboard)

And you should be logged directly into FOP2 and seeing only the wallboard on screen.

Wallboard Agent Variables:

Here is a list of variables you can use on the Agent List or Agent Line templates. All of this variables are calculated for the begining of the current day, starting at 00:01 AM:

{CA}

type: integer

Number of CompleteAgent calls. Answered calls that were completed by the agent (agent hangs up first).

Example dustjs code to get the sum for complete caller and complete agent numbers and get the total number of calls answered by the agent:

{@math key="{CA}" method="add" operand="{CC}"/}

{CC}

type: integer

Number of CompleteCaller calls. Answered calls that were completed by the caller (caller hangs up first).

Example dustjs code to get the sum for complete caller and complete agent numbers and get the total number of calls answered by the agent:

```
{@math key="{CA}" method="add" operand="{CC}"/}
```

{CO}

Type: integer

Number of Complete Outbound calls. Outgoing calls that were performed by the agent.

{RA}

type: integer

Number of Ring no Answer calls. The number of times a call was offered to the agent but the agent failed to pick up.

{ST}

type: integer

Session Time, expressed in seconds. Time since the agent is logged into the queue.

{PT}

type: integer

Pause Time, epxressed in seconds: Time in seconds the agent was paused

{TT}

type: integer

(TalkTime) Talk time in seconds: Time in seconds the agent was in conversation (starting from the current day)

This time includes any time the caller might have been on hold. If you want to show true talktime, deducting held times, use this formula:

```
{@formula value="{TT}-{HT}"/}
```

The value is printed out in seconds. It is possible to change the representation format by using builtin filters. Available filters are:

- hhmmss (shows seconds in HH:MM:SS format)
- mmss (shows seconds in MM:SS format)

If you want to change the format for the result of the previous formula, use something like this:

```
{@filter type="hhmmss"}{@formula value="{TT}-{HT}"/}{/filter}
```

If you want to apply the format for a variable, you can use the standard dustjs filter format:

```
{TT|mmss}
```

{HT}

type: integer

Hold time in seconds: Time in seconds the agent put a caller on hold. For this to work the holdreport plugin for FOP2 must be installed.

If you want to change the format this value is printed, you can use the hhmmss or mmss display filters, for example:

```
{HT|mmss}
```

{WT}

type: integer

Wrapup time in seconds: Time in seconds the agent was paused with the Wrapup reason. For this to work the auto wrapup plugin for FOP2 must be used with a reason of 'Wrapup' (or you can modify whatever tool you use to pause agents to use reason Wrapup).

If you want to change the format this value is printed, you can use the hhmmss or mmss display filters, for example:

```
{WT|mmss}
```

{DIRECTION}

type: string

Direction of the call. Returns a string containing either inbound or outbound.

Return values:

- inbound

- outbound

{PAUSED}

type: integer

Return values:

- 0 = agent is not paused
- 1 = agent is paused in some queues
- 2 = agent is paused in all queues he is a member of

{STATE}

type: string

Agent State

Possible return values: unavail, free, ringing, busy, hold

{CALLTIMER|s}

type: html/string

Call duration timer

{LASTSTATETIMER|s}

type: html/string Text representation for time of last status change for the agent. The event must occur while the panel is open. Any event occurring with the wallboard closed won't be tracked/recorded.

{LASTCALLAGO|s}

type: html/string

Text representation of how much time passed since the last call was taken by the agent

{LASTCALL|s}

type: html/string

Date/Time since the last call was taken by the agent

{REASON}

type=string

Paused reason. If the agent was paused with a reason, this variable will contain that reason

{NAME}

type=string

Agent Name

{QUEUE}

type=string

Queue for the active call

{CLIDNUM}

type=string

Caller id or dialed string of the active call

Many times, the callerid number and callerid name are populated with the same value. If that is the case, and your template shows both, then you might want to condense the view by showing only one of the variables. For doing so you can use the 'eq' dustjs helper, like this:

```
{@eq key=CLIDNUM value=CLIDNAME}{CLIDNUM}{:else}{CLIDNUM} {CLIDNAME}
{/eq}
```

{CLIDNAME}

type=string

caller id name for the active call

Many times, the callerid number and callerid name are populated with the same value. If that is the case, and your template shows both, then you might want to condense the view by showing only one of the variables. For doing so you can use the 'eq' dustjs helper, like this:

```
{@eq key=CLIDNUM value=CLIDNAME}{CLIDNUM}{:else}{CLIDNUM} {CLIDNAME}
{/eq}
```

{AGENTID}

type=string

html id for the agent element.

{CALLTIMERID}

type=string

html id for the call timer element

{LASTSTATETIMERID}

html id for the last state timer element

Example Formulas

Sample dustjs code to calculate the percentage of time an agent has been paused in the current session:

```
{@eq key="{ST}" value="0"}0{:else}{@formula value="{PT}*100/{ST}"/}{/eq}
```

(If session time is ZERO then return zero, otherwise calculate percentage and return that).

--

Example dustjs code to show different icons based on paused status (if value is 1 show a half hourglass icon, if value is 2 show a full hourglass icon):

```
{@select key="{PAUSED}"}
    {@eq value="0"} {/eq}
    {@eq value="1"}<i class='fa fa-hourglass-half'></i>{/eq}
    {@eq value="2"}<i class='fa fa-hourglass'></i>{/eq}
{/select}
```

Wallboard Waiting Calls variables

The Waiting Calls Widget template is a bit more simpler, as there is not much information to show for a waiting call. Here is a list of available variables for it:

{CLIDNUM}

type=string

Callerid of the caller waiting in the queue

Many times, the callerid number and callerid name are populated with the same value. If that is the case, and your template shows both, then you might want to condense the view by showing only one of the variables. For doing so you can use the 'eq' dustjs helper, like this:

```
{@eq key=CLIDNUM value=CLIDNAME}{CLIDNUM}{:else}{CLIDNUM} {CLIDNAME}
{/eq}
```

{CLIDNAME}

type=string

Callerid name of the caller waiting in the queue

Many times, the callerid number and callerid name are populated with the same value. If that is the case, and your template shows both, then you might want to condense the view by showing only one of the variables. For doing so you can use the 'eq' dustjs helper, like this:

```
{@eq key=CLIDNUM value=CLIDNAME}{CLIDNUM}{:else}{CLIDNUM} {CLIDNAME}
{/eq}
```

{CHANNEL}

type=string

The channel name of the waiting call

{DURATION}

Type=string

duration of the waiting call as an incremental clock

{POSITION}

type=integer

Position for the call in the queue

{QUEUE}

type=string

Name of the queue

Wallboard Queue Variables

The queue widget boxes are good to summarize activity for one queue. As a bonus you can set alarms to highlight timers after a certain threshold is met. Also, it is possible to group more than one queue into just one widget by adding a section "Queue Group" in the FOP2 Manager plugin configuration that can have any name, and in its value a comma separated list of queues to include in that group.

{ACTIVECALLS}

type=integer

Number of active (ongoing) calls on that queue.

{ABANDONED}

type=integer

Number of abandoned and unanswered calls

{AGENTSPAUSED}

type=integer

Number of agents that are paused in the queue

{AGENTSREADY}

type=integer

Number of agents logged in the queue that are ready to take a call (not busy or paused)

{AGENTSSTAFFED}

type=integer

Number of agents logged into the queue

{COMPLETED}

type=integer

Number of completed calls. Sum of COMPLETECALLER, COMPLETEAGENT and TRANSFER dispositions.

{COMPLETEDSLA}

type=integer

Number of completed calls, answered before the SLA time was reached. SLA time is defined in Asternic Call Center Stats PRO with setting 'sla_answered'. If you do not have Asternic PRO installed, you can edit the script getstatsfromasternic.php and configure a default SLA_ANSWERED value at the top from that script.

{MAXWAIT|s}

type=html/string

Timer showing the max wait time for waiting calls on that queue

{QUEUE}

type=string

The Name of the queue

{TALKTIME}

type=integer

Total talk time in seconds. The sum of time agents talked on the phone for that queue. Talk Time includes also the duration callers were put on hold. Because of Asterisk limitations, held times are not deducted from queue stats (but you can get them to show in agent status by using the Hold Report FOP2 plugin).

{WAITTIME}

type=integer

Total wait time in seconds. The sum of time callers waited in the queue before being connected to an agent or exited/abandoned the queue.

{SERVICELEVEL}

type=integer

Number of seconds configured as inside service level for that queue. Configured via Asternic Call Center Stats PRO sla_answered setting or modifying getstatsromasternic.php script SLA_ANSWERED[''] value.

{WAITINGCALLS}

type=integer

Number of waiting calls on the queue

Example formulas

Here is a list of sample dustjs formulas you can use to get some special information in your queue widget

Show total number of calls (sum of completed + abandoned):

```
{@formula value="{COMPLETED}+{ABANDONED}"/}
```

Show percentage abandonment rate:

```
{@math key=ABANDONED method="add" operand=COMPLETED}
      {@eq value=0 type="number"}0
      {:else}
      {@formula value="({ABANDONED}/({COMPLETED}+{ABANDONED}))*100"/}
{/eq}
{/math}
```

(first we sum both abandon + completed under a @math section so we can use the result for conditions, if the result value is zero then returns zero because we do not want to divide by zero and burn the universe, otherwise do the math by using the @formula value that can take any valid javascript math evaluation string.)

Same as above but rounded, with no decimals:

```
{@math key=ABANDONED method="add" operand=COMPLETED}
      {@eq value=0 type="number"}0
      {:else}
      {@formula value="round(({ABANDONED}/({COMPLETED}+
{ABANDONED}))*100)"/}{/eq}
{/math}
```

Dust Filters

Filters are used to transform a variable before outputting it. You can attach one or many filters to a variable. There are some built-in filters in Dust, and some others specific for FOP2 and this Wallboard.

Filters are attached to a Dust reference by adding a pipe | and the filter name after the reference name. You can chain filters by adding multiple pipes. The filters will be run from left-to-right.

```
{QUEUE}{~n}
{QUEUE|s}{~n}
```

Another way to apply a filter is by using the @filter helper. This comes handy when you want to apply a filter to a result from some helper like @formula:

```
{@filter type="hhmmss"}{@formula value="{TT}-{HT}"/}{/filter}

{@filter type="piechart"}{@formula
value="round(({ABANDONED}/({COMPLETED}+{ABANDONED}))*100)"/}{/filter}
```

Built-In Filters

- `h` - HTML Encode
- `s` - turn off HTML enconding
- `j` - javascript string encode
- `u` - encode Uri
- `uc` - encode Uri Component
- `js` - JSON Stringify
- `jp` - JSON Parse

FOP2 Specific Filters

- `hhmmss` - Presents a value in seconds into format HH:MM:SS
- `mmss` - Presents a value in seconds into format MM:SS
- `piechart` - Presents an single integer value as a simple pie chart

The piechart filter can read data attributes from the parent element to change some styles, you can set the bar, track and scale colors, the form of the line cap (butt, round or square) and the line width:

```
<span class='number' data-barColor='#DD1144' data-trackColor='#777'
data-scaleColor="#777" data-lineCap="butt" data-lineWidth="10">
{COMPLETED|piechart}
</span>
```

Dust Helpers

Dust helpers extend the templating language and allow you to do some manipulations or calculations.

Helpers look like `{@helper}`.

Logic Helpers

The helpers library comes with the following logic helpers:

- `{@eq}` : strictly equal to

- `{@ne}` : not strictly equal to
- `{@gt}` : greater than
- `{@lt}` : less than
- `{@gte}` : greater than or equal to
- `{@lte}` : less than or equal to

These helpers allow you to print content if one value compared in a certain way to another value is true. For each helper, you specify the first value with the key attribute and the second value with the value attribute. Both key and value can point to a reference or a literal value. Wrap literal values in quotes and leave references un-quoted.

In the following example, the first helper looks for the value PAUSED for an Agent and checks if its set to one. The second checks to see if the number of abandoned calls (ABANDONED) is too high, if conditions are met, a string is printed out:

```
{@eq key=PAUSED value="1"}You are paused{/eq}

{@gt key=ABANDONED value=10}Too many abandons!{/gt}
```

Else

For all logic helpers, you can create an {:else} block that will render if the test is false.

```
{@eq key=ABANDONED value="0"}
  Great! No abandons for now!
{:else}
  You need more agents, some customers are not being served!
{/eq}
```

need more agents, some customers are not being served!

Math Helper

The `{@math}` helper can perform simple math operations in a template and then either output or truth-test the result. It accepts key and method parameters as well as an operand parameter for operations that require two values, like adding or subtracting.

You can nest logic helpers inside a math helper to test the result of the operation.

In the followint example we use math to evaluate the result in a logical expression (to avoid division by zero). We also use the `{@filter}` helper to display the result as a pie chart.

```
{@math key=ABANDONED method="add" operand=COMPLETED}
    {@eq value=0 type="number"}
        {@filter type="piechart"}0{/filter}
    {:else}
        {@filter type="piechart"}{@formula
value="round(({ABANDONED}/({COMPLETED}+{ABANDONED}))*100)"/}{/filter}
    {/eq}
{/math}
```

Formula Helper

The `{@formula}` helper is another (simplified and possibly more powerful) way to do calculations. Unless the {@math} helper, it will only print out the result, you cannot test it with logic conditions afterwards. The {@formula} helper takes the value parameter and evaluates the mathematical expression that contains. For example:

```
{@formula value="round(({ABANDONED}/({COMPLETED}+{ABANDONED}))*100)"/}

{@formula value="1+1"/}

{@formula value="43/2"/}
```

If the result is a float, number will be printed with only two decimals. You can use the round function in order to remove decimals and show a rounded integer, You can also use any Dust reference insde the mathematical expression.