

Breato

Designer User Guide

Ian Jamieson

This document has been prepared by Breato Limited.

©2011 Breato Ltd

Glossary

The following terms appear in this document.

Term	Meaning
Breato	The Breato web development and delivery platform.
Widget	An element which appears on a page. These include standard form elements, such as input boxes and dropdown lists, decorative elements, such as images, and bespoke elements, such as the Breato Interactive Report Table.
Data groups	The types of object held in the system. These are cognate with "entities" or "tables" in other systems.
Data attributes	The attributes of the objects. Each data group automatically has name and description attributes. Data attributes are cognate with "attributes", "columns" or "fields" in other systems.
Data relationships	The relationships between data groups.
Data views	Pre-packaged views of the data groups, their attributes and relationships.
Container	<p>A container of information. It will consist of one or more widgets. A container can be one of:</p> <ul style="list-style-type: none"> • <i>page</i> - a container which can be minimised and which often provides a display of information as a gateway to data maintenance • <i>dialogue</i> - a container which is normally used to display data maintenance information • <i>context</i> - a context menu which is normally displayed when the user "rolls over" the top of a widget and which will normally disappear when the user "rolls out" but which can also be configured to remain displayed until the user explicitly closes it. <p>Pages and dialogues can be moved around the screen by clicking on the header and can be resized by dragging the icon at the bottom right of the footer.</p>
Application	A logical collection of containers and dependent data groups.

Conventions

The following conventions have been adopted in this document:

- text in <> indicates a generic name which should be substituted for an actual name in practice



Caveats

The strings "breato" and "br_" are reserved and should not be used when naming widgets. The following escape sequences are defined and can be used when creating Javascript:

Sequence	Converts to
~bslash	\
~colon	:
~semicolon	;
~que	?
~dq	"
~sq	'
~quo	'
~ob	[
~cb]
~oc	(
~cc)
~op	{
~cp	}
~nl	\n
~and	&
~lt	<
~gt	>
~hash	#
~pc	%
~plus	+
~ed	"
~eq	'

The designer-related containers should not be altered as they are required to allow you to create and maintain the rest of your system and they have been configured specifically. Should you inadvertently modify the designer login page to the extent that you can no longer login, you can use the emergency login page, *elogin.html*.

Overview

This document provides a summary of the functionality provided by the Breato Designer. It describes the underlying design philosophy, features and options. It is intended to be read by system administrators and developers who have the responsibility for maintaining and enhancing the operational functionality of their Breato installation. The Public Configuration user guide provides details of how to set up a series of static, public-facing pages to complement your operational system.

This document contains images illustrating the use of the designer package. Please note that your installation may differ cosmetically from that shown here as "re-skinning" is a feature of Breato.

Architecture

Breato consists of three main components:

- Client software: a browser which supports CSS2, Javascript, XMLHttpRequest
- Web server software
- A data repository

Of these, you are primarily concerned with the first, the client software. The software is intended to be cross-platform and cross-browser compatible. The software has been developed using Mozilla Firefox v3. Most other popular browsers will work. Please see the section on browsers contained in the Breato Wiki at <http://www.breato.biz/tikiwiki> for more details of browser compatibility.

Template Applications

Your Breato installation includes a number of template applications. These can be modified and extended through the Designer. New applications can also be created.

Styling

Default style attributes, such as fonts and colours, are specified using Cascading Style Sheets (CSS). These can be further modified through the Designer for each widget.

Wizards

The Designer Wizard provides the easiest way of creating and maintaining functionality. It is perfectly possible to use the Designer without understanding any of the underlying technologies in any depth. However, it is recommended that you become familiar with CSS, Javascript and ANSI standard SQL in order to get the most from the system.

The whole system that you see has been designed from the ground up using the tools which you have access to.

Parameter Substitution

Breato adopts a "late substitution" policy for parameters. This means that parameters are not substituted until the point at which code is executed. All SQL,

Javascript and multi-line text properties for widgets can refer to the values of other widgets and global variables using the "{}" notation. There are three forms of this notation:

- `{<name>}` - this refers to a value to be found in the current container. If the current widget is an *interactive table* the columns defined in the SQL are inspected first for the value. Next, if it exists, the value of a widget in the container is substituted. Finally, the set of parameters with which the container was invoked is inspected for the given value.
- `{<container name>.<name>}` - this refers to the value of the named widget in the given container. If it does not exist, the container's parameters are inspected.
- `{globals.<name>}` - this refers to a named global variable.

Table of contents

Glossary.....	2
Conventions.....	2
Caveats.....	2
Overview.....	4
Architecture.....	4
Template Applications.....	4
Styling.....	4
Wizards.....	4
Parameter Substitution.....	4
Figures.....	8
Change History.....	9
Configuration.....	10
Styling.....	10
Configuration Options.....	10
Images.....	10
Universes.....	10
Page Layout.....	10
Restricted Access.....	11
Getting Started.....	11
Page Explorer.....	11
Designer Wizard.....	13
Data group maintenance.....	13
Date attribute maintenance.....	14
Data relationship maintenance.....	15
Data view maintenance.....	16
Data page generator.....	18
Managing a Container.....	19
Context menu.....	20
Widget Properties.....	21
Simple properties.....	21
Multiline text properties.....	21
WYSIWYG properties.....	22
Javascript properties.....	22
Sql properties.....	23
Tab index.....	24
Page Properties.....	26
Simple call.....	27
Complex call.....	28
API actions.....	28
Javascript.....	30
Helper Functions.....	30
brAPI.....	30
brAppend.....	30
brClearRadio.....	31
brCommand.....	31
brDisable.....	31
brEmail.....	31
brExportHTML.....	31
brGetGlobal.....	31
brHideDlg.....	32
brInf.....	32
brKeyPress.....	32
brListText.....	32
brMSMove.....	32
brOpenWindow.....	33

brROI	33
brRead	33
brReadSelectionText	33
brRefreshDD	33
brRefreshDDs	33
brRefreshOp	34
brRefreshOps	34
brSetAttribute	34
brSetDate	34
brSetGlobal	35
brSetHidden	35
brSetStyle	35
brShowCtxt	35
brShowDeDlg	36
brShowDlg	36
brShowFB	36
brShowPCal	37
brShowPDF	37
brShowPTCal	37
brShowPage	37
brShowPagelet	38
brShowWYSIWYG	38
brTextSelected	38
brWrite	38
Supported Widgets	39
Form elements	39
Label	39
Textbox	40
Textarea	41
Dropdown	42
Combobox	44
Multiple select	45
Scrolling select	46
Listbox	47
Checkbox	49
Radio button	50
Password	51
Text	52
Free text	53
Hidden text	54
Image	55
Decorative background	56
Plain background	57
Group box	58
Tab control	59
Reporting	59
Interactive Table	59
Plain Table	62
Chart	64
Interactive chart	65
Google App	66
Objects	66
Embedded page	66
RSS	67
WWW	67
Flash	68
Embedded object	69

Google Ads	69
Google Maps	69
Directory listing	71
Directory tree	72
Directory file list	72
Controls	72
Generic Button	72
Save Button	73
New Button	74
Delete Button	74
Refresh Button	75
Inline Calendar	76
Popup calendar	78
File browser	79
Diary	80
Tree view	81
Horizontal resizer	82
Vertical resizer	82
Acknowledgements	84

Figures

Illustration 1: Default page	11
Illustration 2: Page Explorer	12
Illustration 3: Designer Wizard	13
Illustration 4: Data Group Maintenance	13
Illustration 5: Data Attribute Maintenance	14
Illustration 6: Data Relationship Maintenance	15
Illustration 7: Data View Maintenance	16
Illustration 8: Data Page Generator	18
Illustration 9: Container Designer	19
Illustration 10: Bounding Box	19
Illustration 11: Widget Context Menu	20
Illustration 12: Adding a Widget	21
Illustration 13: Widget Properties Dialogue	21
Illustration 14: Text Editor	22
Illustration 15: WYSIWYG Editor	22
Illustration 16: Javascript Editor	23
Illustration 17: Javascript Helper Functions Dialogue	23
Illustration 18: Database Designer Dialogue	24
Illustration 19: Tab Index Dialogue	25
Illustration 20: Page Properties Dialogue	26

Change History

Version	Purpose	Released
V01-001	First release	14 th June 2007
V01-002	Updates to Functions	15 th June 2007
V01-003	Updates to Functions	
V02-001	Second release	1 st July 2008
V02-002	Updates to the description of the page designer.	1 st July 2008
V03-000	Major rewrite.	15 th December 2010
V03-001	Updates to the supported widgets and the helper functions	16 th December 2010

Configuration

Styling

Your installation is supplied with a number of themes (skins) which can be selected or modified as required. Three main files are provided:

- *local/css/breato.css* - cascading style sheet file
- *local/css/breatocss.php* - the file which defines the theme to be adopted
- *local<theme>/breatocssvar.php* - the file which supplies theme-specific information to the *breato.css* file.

These can be modified in accordance with the instructions contained in the file. The style of individual widgets can be further modified through the Designer.

Configuration Options

Two files are supplied which you can modify as required:

- *local/config/header.html* - provides default system settings including html meta tags. This file is included at the top of the index.php and designer.php files (to which you do not have access).
- *local/config/footer.html* - by default, this provides "noscript" tag to display when the browser does not support Javascript. This file is included at the top of the index.php and designer.php files.

Images

Your installation comes with a large number of small icons which you are at liberty to use in your applications. These can be found in the local/images directory and sub-directories. You should replace the "logo.png" file with your own organisation's logo. Renaming or deleting images may cause presentation issues in the designer or the operational system.

Universes

The system is constructed around the concept of **Universes**. Each Universe defines how to interact with a different data repository. Your system comes pre-configured with a default Universe, **Breato**, which provides access to your own PostgreSQL repository on the hosted server. The access details for this Universe cannot be changed and are not displayed.

You may define as many other Universes as you require. Currently, PostgreSQL, MySQL and Microsoft SQL Server repositories are supported with more planned.

The Universes allow reporting information to be read from a number of different sources. Currently, there is no mechanism for automatically combining information from disparate Universes in a single report.

When creating bespoke SQL statements we recommend that you adhere to the ANSI standard. Additionally, when accessing your default Universe, you may use PostgreSQL fuzzy matching and crosstab extensions, if required.

Page Layout

Your installation consists of multiple containers. Containers consist of a number of widgets. Some of the widgets are purely decorative whilst others provide reporting or data input functionality.

Each widget is positioned absolutely on a page. Thus, you can always guarantee how a page is going to look on any browser with a high degree of confidence. The Designer provides a WYSIWYG (What You See Is What You Get) interface.

Restricted Access

Access to the designer software (designer.php) is restricted to the username supplied to you.

Getting Started

To access the designer login page, browse to <your domain>/designer.php and enter the username and password supplied.

On successfully logging in you will see the following:

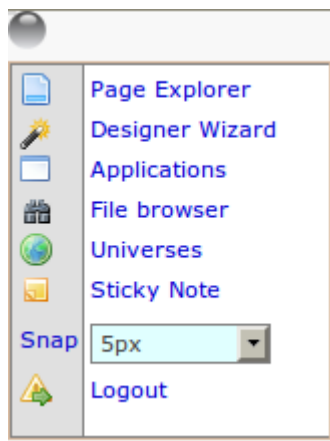


Illustration 1: Default page

The icon at the top left can be clicked to display / hide the designer menu. The menu comprises the following options:

- *Page Explorer* - click to open the page explorer container displaying the containers already defined.
- *Designer Wizard* - click to open the designer wizard container displaying the existing data groups.
- *Applications* - click to open the applications container showing all the existing applications.
- *File browser* - click to open the Breato file manager which allows you to view and manage the files and folders in your installation to which you have access.
- *Universes* - click to open the universe container showing all the user-defined universes.
- *Sticky note* - click to create a new Sticky Note.
- *Snap* - change the grid displayed in the designer. Widgets will automatically snap, when moved, to the top left corner of the nearest grid square.
- *Logout* - click to reset the page to the designer login page.

Page Explorer

The page explorer displays an overview of existing containers and provides the gateway to maintaining existing containers and to creating new containers manually.

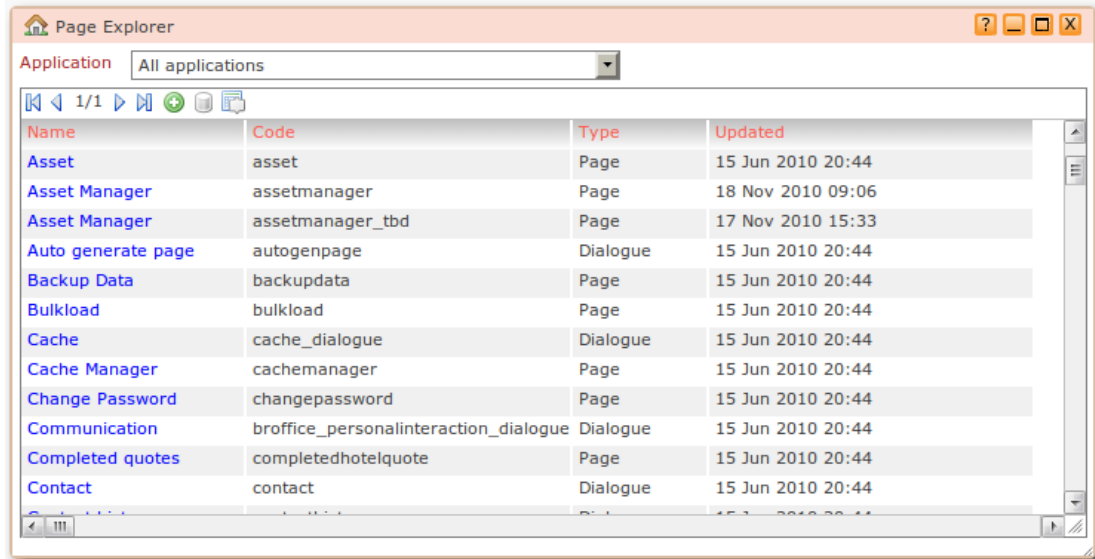


Illustration 2: Page Explorer

The page explorer shows, for each container, its name, code, type and when it was last updated. You can restrict which containers are shown by selecting an application from the dropdown list. Click on the name of a container to display the container in designer mode, or click on the add button to display a blank container.

Designer Wizard

The designer wizard provides a structured, semi-automated method of creating data groups, attributes and relationships, (re-)generating data views and creating containers based on the data groups.

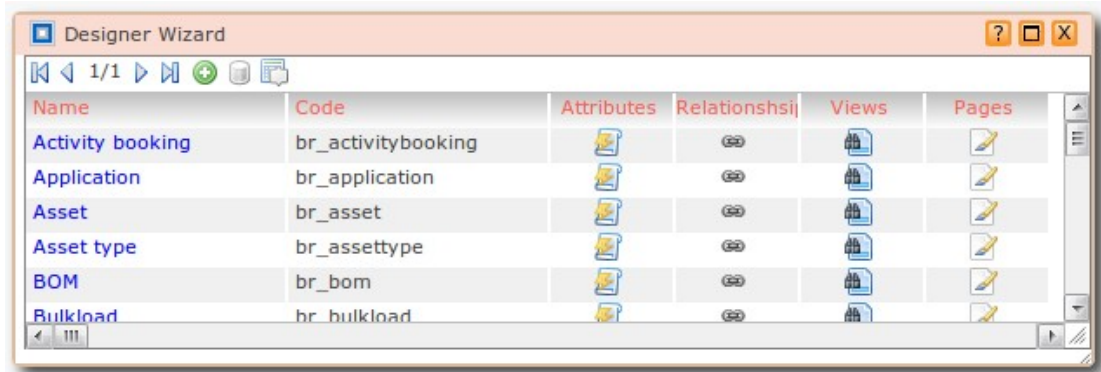


Illustration 3: Designer Wizard

Click on the name of a data group to display the data group maintenance dialogue. Click on the add button to create a new data group.

Click on the attributes icon to display the data attributes maintenance dialogue. Click on the relationships icon to display the data relationship maintenance dialogue. Click on the views icon to display the data views maintenance dialogue. Click on the pages icon to display the auto-generate container dialogue.



Please note that changing anything to do with data groups which have been supplied to you, particularly renaming or deleting groups, attributes and relationships, should only be undertaken by trained users.

Data group maintenance

The data group maintenance dialogue allows you to create and maintain data groups.



Illustration 4: Data Group Maintenance

You can define the name and record a description. The code is automatically generated for you from the name you supply and a prefix which is specific to your installation.

Please note that if you change the name of an existing data group, any containers which reference it will need to be changed manually.

Date attribute maintenance

The data attribute maintenance dialogue allows you maintain the attributes of the given data group.

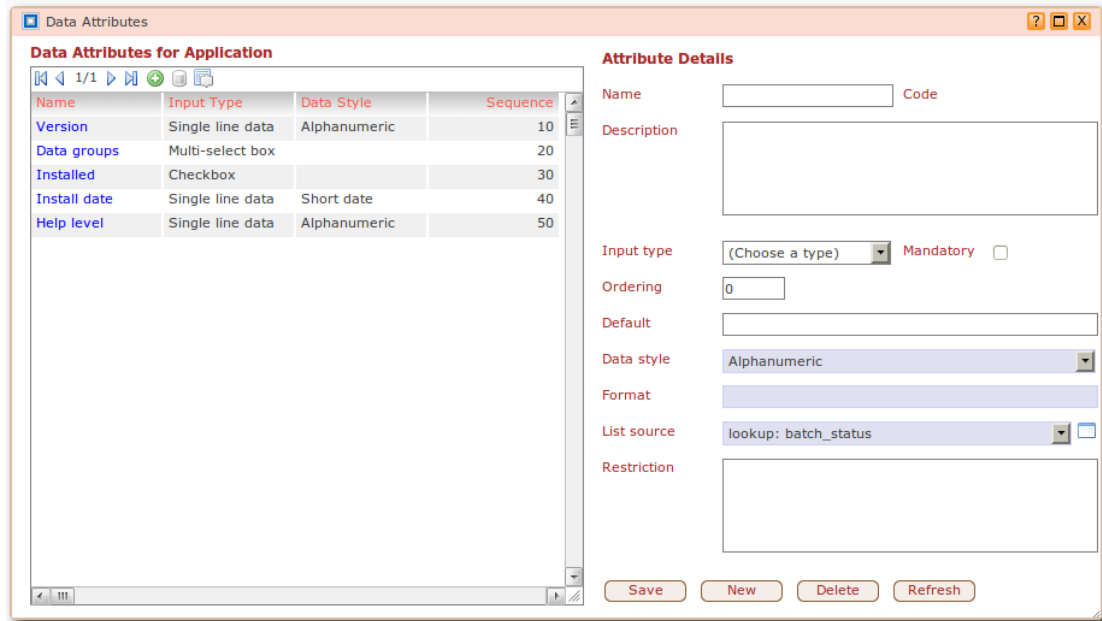


Illustration 5: Data Attribute Maintenance

You can define the following details for an attribute:

- *name* - the human-readable name. A code is automatically generated based on the name. The code should be used to refer to the attribute. The name is used as a label when a container is automatically generated.
- *description* - details of what the attribute is for. The description is used as the tooltip for the label when a container is automatically generated.
- *input type* - this defines the type of widget which should be used to manage the attribute.
- *mandatory* - this indicates that a blank value is not allowed.
- *ordering* - this defines where on an automatically generated container the attribute should appear.
- *default* - this defines the default value to be used if one is not supplied
- *data style* - this defines what type of data are expected.
- *format* - this defines the format string to apply to the value
- *list source* - this defines how to obtain static values for dropdowns
- *restriction* - this contains a Javascript snippet, if required, which can be run to validate the attribute further. A generic warning message will be produced if the value fails the test. The error message can be made more specific by editing the associated widget in a container.



Please note that currently the *format* option is not used.

Data relationship maintenance

The data relationship maintenance dialogue allows the relationships between different data groups to be maintained.

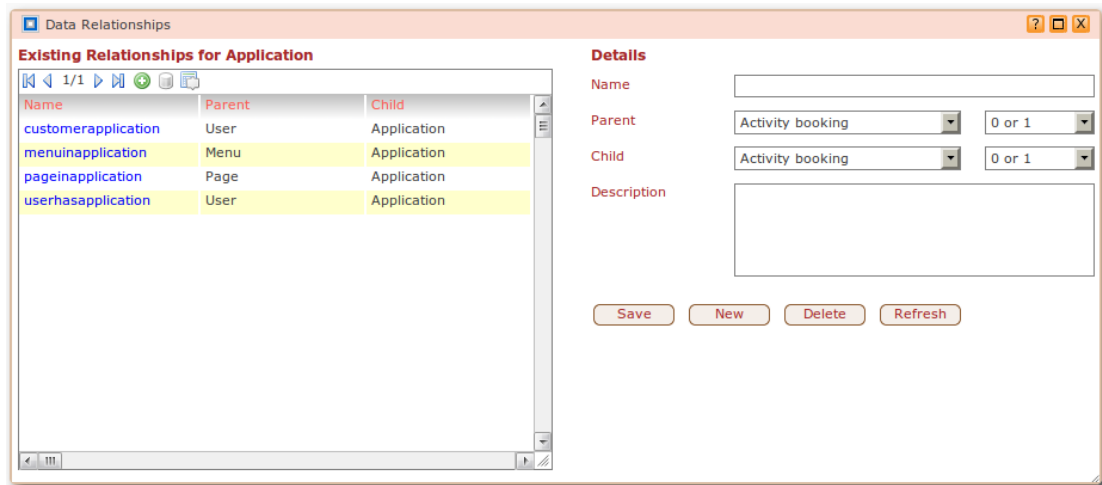


Illustration 6: Data Relationship Maintenance

You can specify:

- *name* - for ease of understanding, this should be short, meaningful and consist of a single string of alphanumeric characters.
- *parent* - the data group which is considered to be the parent or master in a relationship. In a "one to many" relationship, the parent should be the "one". In a "many to many" relationship, the assignment may be arbitrary. The associated dropdown shows how many of the parent there must be in the relationship.
- *child* - the data group which is the child or slave. The associated dropdown shows how many of the child there must be in the relationship.
- *description* - details of what the relationship achieves.



The name should be unique within the total set of relationships, for ease of understanding, and must not duplicate any attribute names for the data group.

Any one data group can be related to any other data group with multiple relationship types.

Data view maintenance

The data view maintenance dialogue allows you to generate the database views of the data group.

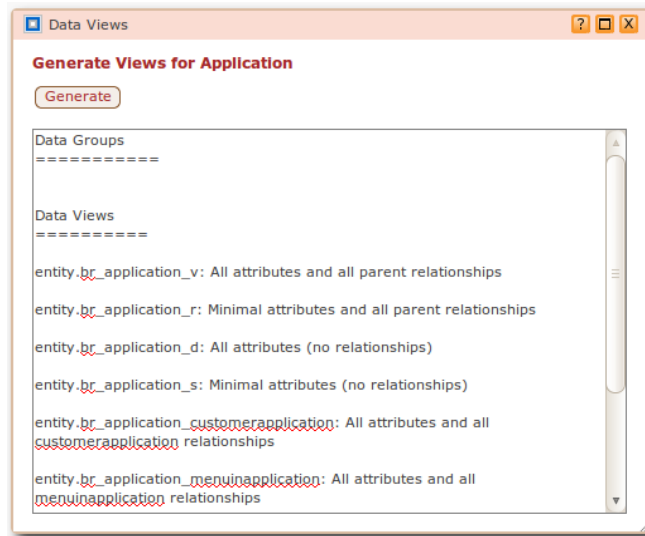


Illustration 7: Data View Maintenance

The data views are a convenient means of access to data and mean that you do not have to understand the architecture of the database.

The following four views are always generated:

- `<code>_s` - the simple view. It comprises the following columns:
 - `id` - the unique identifier of the record in the database
 - `entity_type` - an internal value
 - `entity_status` - a numeric flag which typically has the following values:
 - 1 => active record
 - 3 => active record but not the current one in a sequence
 - 0 => inactive (effectively deleted)
 - 2 => *inactive (effectively deleted)*
 - `datestamp` - a date and time stamp showing when the record was created
 - `updatestamp` - a date and time stamp showing when the record was last updated
 - `xid` - a foreign key (internally generated)
 - `xref` - a reference to another record, possibly in another repository (internally managed)
 - `tag` - a free form field for allowing a record to be identified uniquely (internally managed)
 - `bruserid` - the unique identifier of the user who created or last modified the record (internally managed)
 - `brorgid` - the unique identifier of the organisation to which the user belongs (internally managed)
 - `name` - the name field for the record
 - `description` - the description field for the record
- `<code>_d` - full data view. This view builds on the simple view and includes all defined attributes. For each attribute, three values are included:
 - `<attribute code>` - the value stored in that field
 - `<attribute code>_value` - *the value stored in that field (present for legacy reasons)*

- `<attribute code>_id` - the numeric identifier for the attribute type definition
- `<code>_r` - minimal data view with all parent relationships (i.e. references to all other data groups where this data group is the child in the relationship). This view builds on the simple view. Each relationship is defined in terms of:
 - `<relationship>` - the unique identifier of the record to which it is related
 - `<relationship>_status` - the status value of that relationship
- `<code>_v` - maximal data view. This builds on the full data view and includes all relationships.

Additionally, for each relationship where the data group is the child, there is a specific view of the form `<code>_<relationship>` which has the full data and just the given relationship.



Please note that if you use the "_r" or "_v" relationships without fully qualifying the relationships, you may get multiple sets of data returned where you were only expecting one.

Data page generator

The data page generator allows proforma containers to be generated from the data views.

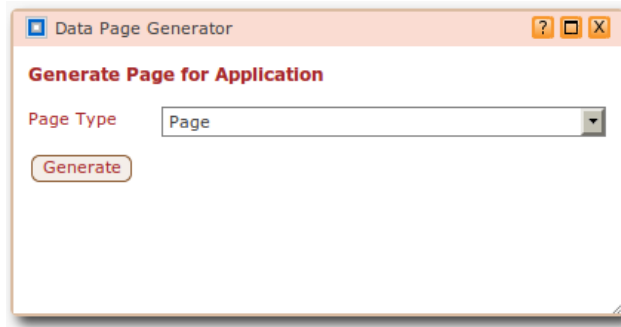


Illustration 8: Data Page Generator

You may generate one of four types of container:

- *page* - a data maintenance form in a page container
- *display page* - a simple interactive table on a page which displays details of the data group instances and which links to the automatically generated dialogue.
- *dialogue* - a data maintenance form in a dialogue container.
- *context* - a data maintenance form in a context menu container.



Please note that the containers are generated on the basis of the views which exist. Thus, if you have modified anything to do with a data group, its attributes or relationships, those changes will not be incorporated until the data views have been (re-)generated.

Managing a Container

However you generate a container - automatically or manually - you are likely to want to customise it. The image below is of a blank container which you can generate by clicking on the add button on the *Page Explorer*.

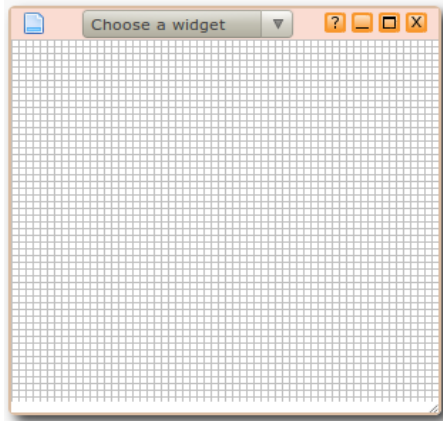


Illustration 9: Container Designer

The grid shown is determined by the *Snap* option on the main menu.

Typically, a container will comprise a number widgets. Details of the widgets supported are shown below in section ????.

The dropdown list at the top allows you to select a widget in the container. Alternatively, you can click on it to select it.

If you left-click on a widget, it will be highlighted and the properties dialogue displayed (see below). If you use the *Ctrl* button you can select multiple widgets, in which case a common subset of properties is shown in the dialogue. Changing one of the properties will affect all the selected widgets. You can also select multiple widgets using the "bounding box". Click on a part of the container where there is no widget and drag the cursor around to create a bounding box. All widgets within the box will be selected.

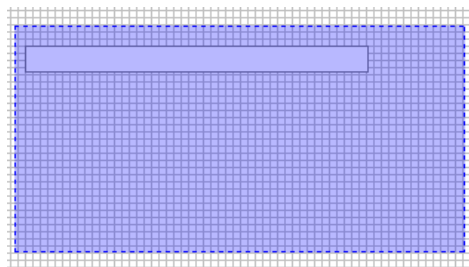


Illustration 10: Bounding Box

You can reposition a widget by dragging it around the container. If a grid is displayed, it will snap to the appropriate top left corner. If you have multiple widgets selected you can move them all simultaneously.

If you right-click anywhere in the body of the container, the widget context menu is shown, as illustrated below. You can use the *Ctrl* button to select multiple items to operate on simultaneously.

Context menu

The designer context menu allows you to add, copy and remove widgets.

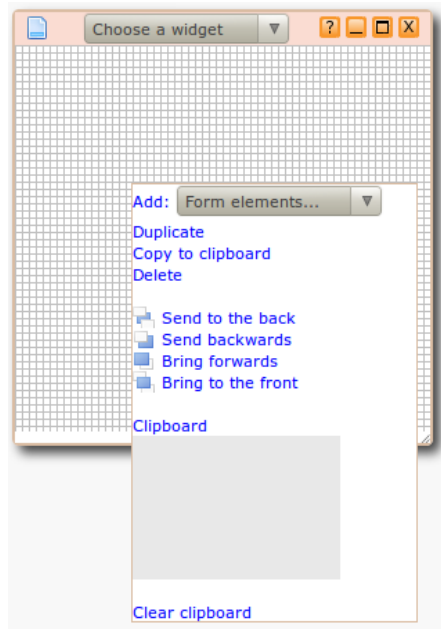


Illustration 11: Widget Context Menu

You have the following options:

- *add* - add a new widget (see below).
- *duplicate* - create an exact copy of any widgets selected.
- *copy to clipboard* - copy any widgets selected to the built-in designer clipboard. Clicking on a widget in the clipboard will cause it to be placed on the current container at the current cursor point. Thus, this provides a mechanism for copying widgets between containers.
- *delete* - delete any currently selected widgets
- *send to the back* - change the selected widgets' z-order to be the lowest of all the widgets currently displayed
- *send backwards* - demote the selected widgets' z-order by one position
- *bring forwards* - promote the selected widgets' z-order by one position
- *bring to the front* - change the selected widgets' z-order to be the highest of all the widgets currently displayed
- *clear the clipboard* - remove all widgets from the clipboard

If you choose to add a new widget, a dropdown of supported widgets is displayed.

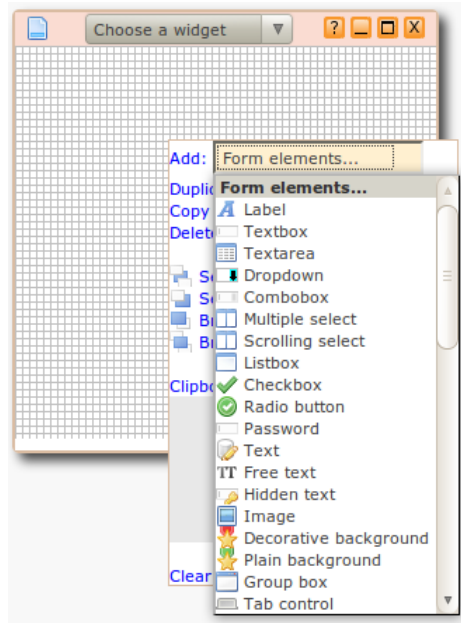


Illustration 12: Adding a Widget

Widget Properties

Each widget has a set of properties which you can change which affect its appearance or its functionality. See below for a description of all the supported widgets and their properties.

Clicking on a widget will cause the properties dialogue to be displayed.

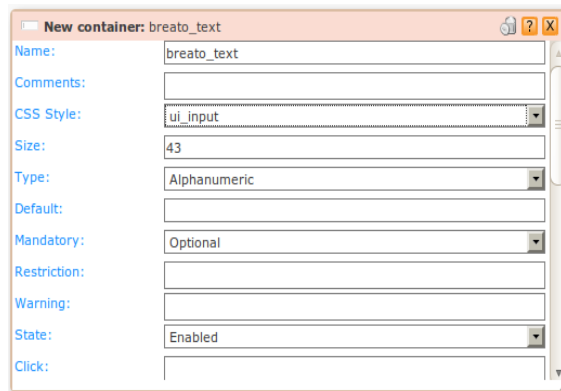


Illustration 13: Widget Properties Dialogue

Simple properties

Some properties, such as the *name* can be typed in directly to the entry box. press return to ensure that the change is put into effect. Others are controlled by dropdown lists which can be changed as required.

Multiline text properties

Some properties, such as *warning*, are maintained by double-clicking in the input box to display the *Text Editor* dialogue. You can type text directly into the body

of the editor. The text can include references to widgets using the "{}" notation. These will be substituted with the actual value of the given widget when the text is displayed.

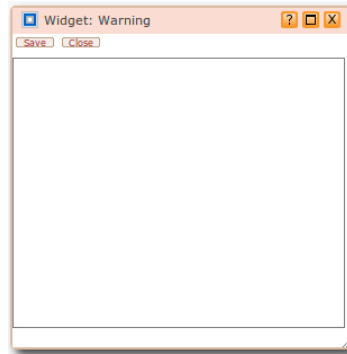


Illustration 14: Text Editor

WYSIWYG properties

Some properties, such as *comments*, are maintained by double-clicking in the input box to display a WYSIWYG editor (based on ckEditor).

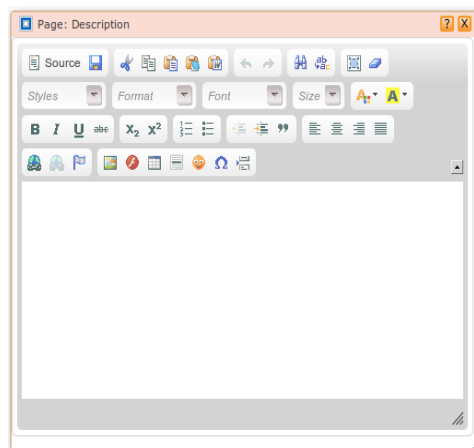


Illustration 15: WYSIWYG Editor

You can type directly into the body of the editor and, when you are satisfied, click on the "save" button to commit the changes.

Javascript properties

Some properties, such as *click*, require snippets of Javascript code. Double-clicking in the input box causes the Javascript editor to be displayed.

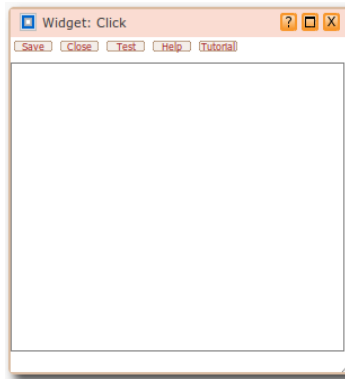


Illustration 16: Javascript Editor

You can type Javascript code in the body of the editor. Use *Ctrl-t* to start indenting by a (further) tab; delete the tab character to stop indenting.



You can test the syntactic validity of your code (or the part highlighted) by clicking on the *Test* button. Please note that a very strict validation is applied and that passing the test is no guarantee that the code will do what you want. You can press the *Close* button to close the editor without saving any changes, or press the *Save* button to save and close.

Two help options are provided. You can click on the *Tutorial* button to open the online W3 Javascript tutorial in a new window. You can get additional help on how to use the editor and the built-in helper functions by clicking on the *Help* button. This displays the *Javascript Helper Functions* dialogue. The *bookmarks* on the left-hand side provide quick access to the individual functions. Clicking on the actual helper function reference on the right-hand side will cause a proforma version of the function to be inserted at your current cursor insertion point in the editor.

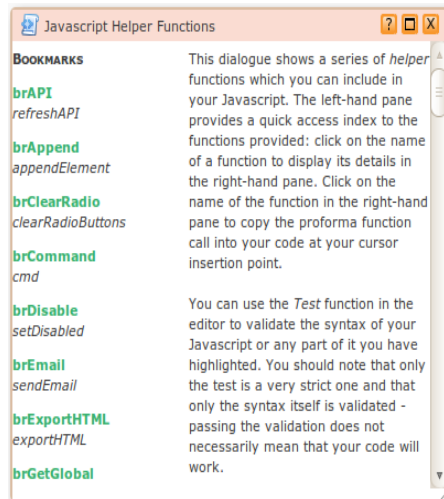


Illustration 17: Javascript Helper Functions Dialogue

Sql properties

The SQL property for widgets which retrieve their data from a data repository is maintained by double-clicking in the input box to display the *Database Designer* dialogue.

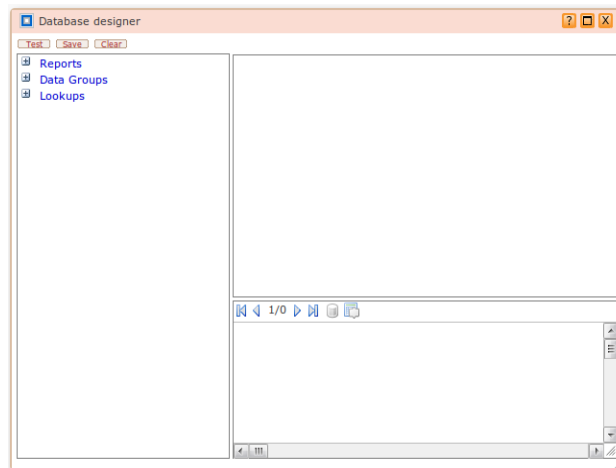


Illustration 18: Database Designer Dialogue

You can resize the different panes by clicking and dragging on the space in between them.

The left-hand pane provides a number of options for accessing pre-defined SQL. The *Reports* tree shows all the defined reports. The *Data Groups* tree provides access to various options for all the data groups. The *Lookups* tree provides options for all the groups of options defined in the *lookups* table. Clicking on an option causes the corresponding SQL to be placed at the insertion point in the top right-hand pane.

You can press the *Test* button to test the (selected) SQL. The results, including details of any errors, are shown in the table in the bottom right pane.

Pressing the *Clear* button will cause the SQL pane to be cleared. Pressing the *Save* button will cause the changes to be committed back to the properties dialogue.

For legibility, you can indent your code by pressing *Ctrl-t* at the start of a line. You can undo this by deleting the resulting tab character.

Your SQL should be appropriate for the *Universe* selected. It can include references to other values using the `{}` notation.

Tab index

The order of tabbing around the screen is controlled by the *tab index* property. This should be a numeric value or blank. The widget with the lowest non-blank tab index will receive the focus by default when the container is opened. A blank value indicates that a widget will not be tabbed to. You can enter a value directly into the input box.

A wizard is provided to help with setting all tab indexes in one go. You can invoke this by clicking on the tab button next to the input box.

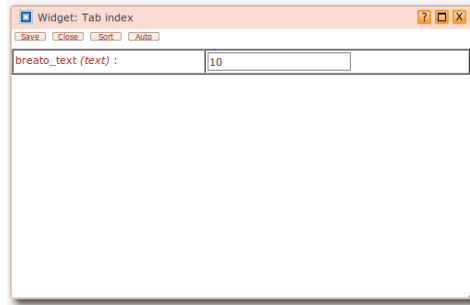


Illustration 19: Tab Index Dialogue

All widgets are displayed and initially sorted by tab index value. You can type values into the input boxes and press *Sort* to reorder the widgets by tab index.

You can press the *Auto* button to have the designer take a best guess at the appropriate tab orders. In this case, all labels, decorations, hidden fields and images without any associated events are ignored. Other widgets are ordered on the basis of their vertical offset and then horizontal offset from the top left corner of the container.

Press the *Close* button to close the wizard without saving, or press the *Save* button to commit the changes back to the properties dialogue and then close the wizard.

Page Properties

Each page has its own set of properties. The *Page Properties* dialogue is displayed / hidden by clicking on the icon at the top left in the container's header.

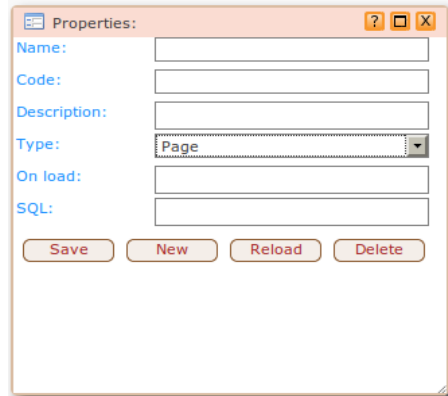


Illustration 20: Page Properties Dialogue

You can change the following:

- *name* - a human-readable name for the container. This is displayed, by default, in the header when the container is displayed operationally.
- *code* - a unique code to use to refer to the container. This is used when referring to widgets in a container and when performing actions on the container as a whole, such as opening and closing it.
- *description* - details of what the container does. This information is maintained using the *WYSIWYG* editor and is displayed from the help button of the container.
- *type* - defines whether the container is a page, dialogue or context menu. This definition controls some of the display features, such as whether to show the header and footer. Only "pages" are shown in the menu maintenance dialogue available in the operational system.
- *on load* - this is a Javascript snippet which defines one or more actions to be taken when the container is (re-)displayed. The code is maintained by the *Javascript* editor.
- *SQL* - the sql which controls how data are loaded and maintained for a data maintenance page or dialogue. This is maintained using the *Database Designer* dialogue. This particular SQL statement is referred to in the documentation as the container's API.

The SQL is of particular note and needs to be understood in detail if you are creating a data maintenance container from scratch or materially altering an existing container. When a container is auto-generated, this will be created for you and if you do not want to change which fields are displayed and stored you need do nothing with this.

There are two forms the API may take. To illustrate these, we will use the following example which is also shown on the tutorial video:

Data group: Pet - code: br_pet

Data attributes: Type - code: type
Sex - code: sex

Data relationships: petowner : links to the br_user data group as the child
petseller : links to the br_organisation data group as the

child

Assume we have created a data maintenance form with the following widgets on it:

- *name* - the name of the pet
- *description* - any comments about the pet
- *type* - a dropdown specifying the type of animal
- *sex* - a dropdown specifying the sex of the animal
- *petowner* - a dropdown of users recorded
- *petseller* - a dropdown of organisations recorded

Simple call

This is a call to a simplified stored procedure. It hides some of the complexity of the database interaction and is, as a result, not as flexible as its complex counterpart.

The call is of the form:

```
select * from simple_entity('{action}', coalesce({id}, 0), '<data group
code>', '{name}', '{description}',
ARRAY['<data attribute code 1>', ..., '<data attribute code n>',
      '<data relationship code 1>', ..., '<data relationship code n>'],
ARRAY['{data attribute code 1}', ..., '{data attribute code n}',
      '{data relationship code 1}', ..., '{data relationship code n}'])
```

The first ARRAY defines the names of the attributes and relationships which this container is interested in. In the case of a *select* action, or where a record is returned in response to some other action, these names are used to identify widgets in the container to populate. If there is no corresponding widget, the value is stored in the widgets list of parameters.

Our proforma API for the given example would look like:

```
select * from simple_entity('{action}', coalesce({id}, 0), 'br_pet', '{name}',
'{description}',
ARRAY['type', 'sex', 'petowner', 'petseller'],
ARRAY['{type}', '{sex}', '{petowner}', '{petseller}'])
```

From the earlier description we can see that when the API is invoked, the elements in {} will be substituted by actual values.

If we invoked this on our example data group to save the record with unique identifier 21, we might have a call as follows:

```
select * from simple_entity('save', coalesce(21, 0), 'br_pet', 'Fido', 'Faithful
companion',
ARRAY['type', 'sex', 'petowner', 'petseller'],
ARRAY['1', '2', '5', '3'])
```

where 1 refers to the id of a value in the *type* lookup table

2 refers to the id of a value in the *sex* lookup table

5 refers to the id of a user in the *br_user* table

3 refers to the id of an organisation in the *br_organisation* table

If we did not want to record or display details of the *type* we would simply leave the appropriate references out of the API. Any values already stored for the

type would be maintained. Similarly, if we did not want to record or display the *petseller* we would leave that reference out.

Using this form of API we can only manage relationships where the given data group is the child.

Complex call

This call provides the maximum amount of control.

The call is of the form:

```
select * from manage_entity('{action}', coalesce({id}, 0), '<data group code>', '{name}', '{description}',
ARRAY['<data attribute code 1>', ..., '<data attribute code n>'],
ARRAY['{data attribute code 1}', ..., '{data attribute code n}'],
ARRAY[['<data relationship code 1>', '<relationship type>', '<widget name 1>', '<optional relationship action>'], ..., ['<data relationship code 1>', '<relationship type>', '<widget name n>', '<optional relationship action>']],
ARRAY['{widget name 1}', ..., '{widget name n}'])
```

The first ARRAY defines the set of attributes this container is interested in. The second ARRAY defines the corresponding values.

The third ARRAY defines the set of relationships the container is interested in. The fourth ARRAY defines the corresponding values. Each relationship is defined in terms of three parameters with an optional fourth:

- *data relationship code* - corresponds to the name of the relationship.
- *relationship type* - 1 indicates that the subject data group of this API is the parent in the relationship, 2 indicates that it is the child.
- *widget name* - refers to the name of a widget in the container. Normally this will be the same value as the *data relationship code*.
- *relationship action* - this parameter is optional. It can be *replace* (default): indicates that any existing relationships of this type should be replaced; *save*: indicates that existing relationships will be maintained but their statuses will be set to 3.

If we invoked this on our example data group in the same situation as described above, we might have a call as follows:

```
select * from manage_entity('save', coalesce(21, 0), 'br_pet', 'Fido', 'Faithful companion',
ARRAY['type', 'sex'],
ARRAY['1', '2'],
ARRAY[['petowner', '2', 'petowner', 'replace'], ['petseller', '2', 'petseller', 'replace']],
ARRAY['5', '3'])
```

As before, any fields we did not want to display or manage we would simply omit from the call.

API actions

The following *action* parameters are supported in the API call:

- *select* - return the given record, retrieving the attributes and relationships specified.

- *save* - save the given record. If the id passed in is 0 a new record is created otherwise an existing one is updated. Return this record.
- *delete* - physically delete the given record and its relationships. Return a blank record.
- *deletecopy* - physically delete the given record and its relationships. Do the same for any records which are linked through the xref field (see *copysave* below). Return a blank record.
- *withdraw* - mark the status of the record as withdrawn (entity_status bit 0 set to 0). Return this record.
- *close* - mark the status of the record as closed (entity_status bit 0 set to 0). Return this record.
- *copy* - make a complete copy of the given record. Retrieve this record.
- *copysave* - make a complete copy of the given record. Update the new record's xref field with the id of the existing record and clear the entity_status bit 0. Update the existing record's datestamp and updatestamp. Return this record.

Javascript

You can modify the presentation and interaction of many parts of the UI by adding your own Javascript snippets to container and widget properties. The standard Javascript is enhanced by a number of "helper" functions (detailed below) and by the ability to substitute values using the `{}` notation.

Your installation provides a number of global values, referred to using the notation `{globals.<name>}`. As a minimum, you have access to:

- `userid` - the logged in user's unique identifier
- `name` - the logged in user's name
- `email` - the logged in user's email address
- `language` - the short code of the logged in user's preferred language
- `reportorg` - the logged in user's "owning" organisation
- `defpage` - the code of the logged in user's default page
- `docroot` - the root directory in the server file system for the logged in user



Please note that these values are set when the user logs in and are not changed until the next log in.

Helper Functions

brA2H

Proforma: `brA2H('text');`

Alias: `asciiToHex`

Converts the text passed in to a string of hexadecimal numbers.

brAPI

Proforma: `brAPI('widget', 'action', 'next', check, sync, 'parameterlist');`

Alias: `refreshAPI`

This is the principle way of interacting with instances of data groups. This optionally causes an instance of a data group to be created, modified or deleted, or its details to be selected. It is typically used on operational data maintenance pages.

- `Widget` - string: the name of a widget. Can also be **this**.
- `Action` - string: the action to take (one of select, save, new, delete)
- `Next` - string: containing one or more valid Javascript statements to execute once the data repository interaction has occurred. Typically, you might use this to force a report to refresh. This can be null and it can contain references to other widgets.
- `Check` - boolean: `true =>` check the data for completeness according to the mandatory settings and any restrictions imposed.
- `Sync` - boolean: controls the call to the data repository: `true =>` synchronous call - the execution of code does not continue until the call is complete.
- `Parameterlist` - string: a comma-separated list of `name:value` pairs which provide additional parameters for the call.

Note that if you want the *next* action to refer to values returned by the API call, you should make the API call synchronous.

brAppend**Proforma:** brAppend('widget', 'text');**Alias:** appendElement

Appends text to the contents of the identified widget. The types of widget to which this command applies are: plaintext, dbtext, freetext, label, button, text, password, textarea

- Widget - string: the name of a widget.
- Text - string: the text to append.

brClearRadio**Proforma:** brClearRadio('widget');**Alias:** clearRadioButtons

Unchecks all the radio buttons with the same name as the identified widget.

- Widget - string: the name of a widget.

brCommand**Proforma:** brCommand('command', 'arguments');**Alias:** cmd

Executes a Breato registered command on the server with the accompanying arguments.

- Command - string: the Breato alias for a command.
- Arguments - string: a set of arguments to apply to the command.

brDisable**Proforma:** brDisable('widget', state);**Alias:** setDisabled

Set the specified widget's disabled state.

- Widget - string: the name of a widget. Can also be **this**.
- State - boolean: true => disabled.

brEmail**Proforma:** brEmail('to', 'from', 'subject', 'body');**Alias:** sendEmail

Send email. Sending email is subject to specific terms of business.

- To - the address(es) of the recipient(s).
- From - the address of the sender.
- Subject - the subject of the email.
- Body - the body of the email.

brExportHTML**Proforma:** brExportHTML('widget');**Alias:** exportHTML

Export the contents of a particular widget in HTML format to a new window.

- Widget - string: the name of the widget to export.

brGetGlobal**Proforma:** brSetGlobal('name');**Alias:** getGlobal

Retrieve a global variable value.

- Name - string: the name of the variable.

brH2A**Proforma:** brH2A('text');**Alias:** hexToAscii

Attempts to convert the text passed in from a hexadecimal string to its ascii equivalent. If it fails at any point, the text passed in is returned.

brHideDlg**Proforma:** brHideDlg('widget', 'state');**Alias:** disposeDialogue

This causes a dialogue to be hidden from view. It is typically used in conjunction with an action to save or delete data displayed in a dialogue.

- Widget - string: the name of a widget in the dialogue or the code of the dialogue. Can also be **this**.
- State - string: one of *hide*, which causes the dialogue to be hidden as it stands, or *cancel*, which causes the dialogue to be hidden and its contents returned to their default state when the dialogue was last invoked.

brInf**Proforma:** brInf('body', 'header', 'type', append);**Alias:** showInformation

Show information in a separate dialogue. This can be used for showing whatever you wish.

- Body - string: the body of the message.
- Header - string: a header for the message.
- Type - string: one of **information** or **warning**.
- Append - boolean: true => append the message to any existing messages.

brKeyPress**Proforma:** brKeyPress(key, 'action');**Alias:** keyPress

Trap a key event and take the specified action. This should be associated with the Keypress event for a widget.

- Key - integer: the key code.
- Action - string: one or more Javascript statements to execute when the key press is detected.

brListText**Proforma:** brListText('widget');**Alias:** getSelectionText

Get the text of an option selected in a dropdown, listbox or multi-select widget.

- Widget - string: the name of the dropdown, listbox or multi-select widget.

brMSMove**Proforma:** brMSMove('source', 'destination', all, preserve);**Alias:** MultiSelectMove

Move one or more values between a source and destination multi-select box.

- Source - string: the name of a multi-select widget from which values will be moved.
- Destination - string: the name of a multi-select widget to which values will be moved.
- All - boolean: true => move all entries from the source to the destination widget, false => only move selected values.
- Preserve - boolean: true => the source retains all the items moved to the destination

brOpenWindow**Proforma:** brOpenWindow('url', 'winname', 'params', focus);**Alias:** WindowOpen

Open a (new) window pointing to the given url and optionally cause it to receive the focus.

- Url - string: the absolute or relative url for the window.
- Winname - string: a name for the destination window.
- Params - string: parameters to specify how to display the window
- Focus - boolean: true => force the specified window to have the focus

See the User Guide for additional details.

brROI**Proforma:** brROI('Roilist');**Alias:** handleROI

Refresh widgets which have registered an interest in the objects listed in the Roilist.

- Roilist - string: a comma-separated list of objects in which this widget has an interest.

brRead**Proforma:** brRead('widget');**Alias:** getElementValue

Read the value of a widget.

- Widget - string: the name of a widget.

brReadSelectionText**Proforma:** brReadSelectionText('widget');**Alias:** getSelectionText

Reads the text of a dropdown list.

- Widget - string: the name of the widget

brRefreshDD**Proforma:** brRefreshDD('widget', sync);**Alias:** refreshDropDown

Refresh a dropdown, keeping the selection. Dropdowns in this context include combo boxes, list boxes, multi-select boxes.

- Widget - string: the name of a widget.
- Sync - boolean: controls the call to the data repository: true => synchronous call - the execution of code does not continue until the call is complete.

brRefreshDDs

Proforma: brRefreshDDs('widget', sync);

Alias: refreshDropDowns

Refresh the dropdowns in the identified container or the container holding the identified widget, keeping the selection. Dropdowns in this context include combo boxes, list boxes, multi-select boxes.

- Widget - string: the name of a widget in a container or the code of a container.
- Sync - boolean: controls the call to the data repository: true => synchronous call - the execution of code does not continue until the call is complete.

brRefreshOp

Proforma: brRefreshOp('widget', sync);

Alias: refreshOutput

Refresh the output represented by the named widget. Outputs in this context include reports, tables, RSS feeds, maps and charts.

- Widget - string: the name of a widget.
- Sync - boolean: controls the call to the data repository: true => synchronous call - the execution of code does not continue until the call is complete.

brRefreshOps

Proforma: brRefreshOp('widget', sync);

Alias: refreshOutputs

Refresh the outputs in the identified container or the container holding the identified widget. Outputs in this context include reports, tables, RSS feeds, maps and charts.

- Widget - string: the name of a widget in a container or the code of a container.
- Sync - boolean: controls the call to the data repository: true => synchronous call - the execution of code does not continue until the call is complete.

brSetAttribute

Proforma: brSetAttribute('widget', 'attribute', 'value', delay);

Alias: setAttribute

Sets the HTML attribute of a widget to a given value in a number of milliseconds time. This can be used, for example, to change an image displayed for a few seconds and then to change it back. This is used on the operational page when the layout is saved: the icon changes to a green tick for a second.

- Widget - string: the name of the widget
- Attribute - string: the name of the HTML tag attribute (e.g. src)

- Value - string: the value to set the attribute to
- Delay - integer: the time (in milliseconds) before the change should take place.

brSetDate

Proforma: `brSetDate('datatype', 'startdatewidget', 'enddatewidget');`

Alias: `setDate`

Creates a date range based on the datatype and saves the start and end dates in the named widgets. All week-based date ranges start on a Sunday.

- Datatype - string: the type of date required: can be today, tomorrow, yesterday, weektodate, lastweek, monthtodate, lastmonth, yeartodate, lastyear. Any other value is assumed to be a string representation of a date and will be used to generate the output.
- Startdatewidget - string: the name of the widget where the start date should be stored.
- Enddatewidget - string: the name of the widget where the end date should be stored

- **brSetGlobal**

Proforma: `brSetGlobal('name', 'value');`

Alias: `saveGlobal`

Save a global variable value. The value is subsequently accessible by referring to *globals.name*.

- Name - string: the name of the variable.
- Value - string: the value to save. All such vales are held as string.

brSetHidden

Proforma: `brSetHidden('destintation', 'id', 'value');`

Alias: `saveHiddenValue`

Save a value to a hidden element on a particular page. If the element does not already exist it is created.

- Destination - string: the name of a widget in the target container or the code of a container.
- Id - string: the name of a hidden element on the page.
- Value - string: the value to save. All types of value should be passed as strings.

brSetStyle

Proforma: `brSetStyle('widget', 'stylename', 'stylevalue', delay);`

Alias: `setStyle`

Set a particular style for a widget. Optionally, delay the setting of the style by the given number of milliseconds.

- Widget - string: the name of the widget requesting the popup calendar.
- Stylename - string: the style name.
- Stylevalue - string: the style value.
- Delay - integer: an optional number of milliseconds delay before the change is made.

brShowCtxt**Proforma:** brShowCtxt('container', 'widget', refresh, 'parameters', 'next', 'after');**Alias:** showContext

Display a context menu. The context menu will be shown as close to the point on the screen from which it was invoked as is possible.

- ☒ Container - string: the code of the context menu.
- ☒ Widget - string: the widget which is requesting the context menu.
- ☒ Refresh - boolean; true => the contents of the context menu will be refreshed each time the menu is invoked.
- ☒ Parameters - string: a semi-colon-separated list of name:value pairs which provide additional parameters for the call. The value can be a reference to another value - for example id:{id}
- ☒ Next - string: a series of Javascript statements to execute once the context menu has been displayed.
- ☒ After - string: a series of Javascript statements to execute once the context menu has been hidden.

brShowDeDlg**Proforma:** brShowDeDlg('name', null, refresh, 'parameters', 'next', 'after');**Alias:** showDataEntryDialogue

Display the system generated data entry / maintenance dialogue for a particular entity. The dialogue will be shown as close to the point on the screen from which it was invoked as is possible.

- ☒ Name - string: a name for the dialogue.
- ☒ Refresh - boolean; true => the contents of the dialogue will be refreshed each time the menu is invoked.
- ☒ Parameters - string: a semi-colon-separated list of name:value pairs which provide additional parameters for the call. The parameter list should contain **entity:data group name;id:the id of the instance of the data group to be managed.**
- ☒ Next - string: a series of Javascript statements to execute once the context menu has been displayed.
- ☒ After - string: a series of Javascript statements to execute once the context menu has been hidden.

brShowDlg**Proforma:** brShowDlg('container', null, refresh, 'parameters', 'next', 'after');**Alias:** showDialogue

Display a dialogue. The dialogue will be shown as close to the point on the screen from which it was invoked as is possible.

- ☒ Container - string: the code of the dialogue.
- ☒ null
- ☒ Refresh - boolean; true => the contents of the dialogue will be refreshed each time the menu is invoked.
- ☒ Parameters - string: a semi-colon-separated list of name:value pairs which provide additional parameters for the call. The value can be a reference to another value - for example id:{id}
- ☒ Next - string: a series of Javascript statements to execute once the dialogue has been displayed.
- ☒ After - string: a series of Javascript statements to execute once the dialogue has been hidden.

brShowFB

Proforma: brShowFB('Widget', 'Restriction', 'BaseDirectory', 'Action', StripBaseDir);

Alias: openFileBrowserWindow

Show a PDF in a separate window.

- Widget - string: the name of the widget to which a selected filename will be returned.
- Restriction - string: a set of comma-delimited file extensions which the browser will restrict to showing.
- BaseDirectory - string: the relative base directory from the web root.
- Action - string: *return* (default) indicates that the return icon will be shown to allow a selected filename to be returned.
- StripBaseDir - boolean: true => remove the base directory string from the returned value; defaults to false.

brShowPCal

Proforma: brShowPCal('sourcewidget', 'calendarid', 'outputwidget');

Alias: showPopupCalendar

Show a popup calendar.

- Sourcewidget - string: the name of the widget requesting the popup calendar.
- Calendarid - string: a name for the calendar.
- Outputwidget - string: the name of the widget where the selected date is going to be placed.

brShowPDF

Proforma: brShowPDF('Code', 'Title', 'Source', Height, Width);

Alias: showPDF

Show a PDF in a separate window.

- Code - string: the code name for the window.
- Title - string: the title for the window.
- Source - string: an absolute or relative URL to the PDF.
- Height - float: a default height for the window.
- Width - float: a default width for the window.

brShowPTCal

Proforma: brShowPTCal('sourcewidget', 'calendarid', 'outputwidget');

Alias: showPopupTimeCalendar

Show a popup calendar with time entry.

- Sourcewidget - string: the name of the widget requesting the popup calendar.
- Calendarid - string: a name for the calendar.
- Outputwidget - string: the name of the widget where the selected date is going to be placed.

brShowPage

Proforma: brShowPage('container', 'parameters', refresh, 'next', sync, tiled);

Alias: openPage

Display a page. The page will be shown initially in full page mode - subsequent calls to the page will redisplay it in the state it was last in.

- Container - string: the code of the page.
- Parameters - string: a semi-colon-separated list of name:value pairs which provide additional parameters for the call. The value can be a reference to another value - for example id:{id}
- Refresh - boolean; true => the contents of the page will be refreshed each time the menu is invoked.
- Next - string: a series of Javascript statements to execute once the page has been displayed.
- Sync - boolean: controls the call to the data repository: true => synchronous call - the execution of code does not continue until the call is complete.
- Tiled - defines how to show the page when first displayed: true => tiled, false => full screen, undefined => use the configuration parameter open_pages_tiled

brShowPagelet

Proforma: brShowPagelet('container', 'pageid', 'parameters', 'next', refresh);

Alias: openPagelet

Display a pagelet A pagelet is a container for a page embedded in another page or dialogue.

- Container - string: the code of the pagelet container.
- Pageid - string: the code of the pagelet to open.
- Parameters - string: a semi-colon-separated list of name:value pairs which provide additional parameters for the call. The value can be a reference to another value - for example id:{id}
- Next - string: a series of Javascript statements to execute once the context menu has been displayed.
- Refresh - boolean or string:
 - true or 'all' => refresh the contents of the pagelet each time it is opened
 - false or 'none' => do not automatically refresh
 - 'controls' => refresh the controls on the pagelet each time it is opened
 - 'outputs' => refresh the outputs (tables, charts) on the pagelet each time it is opened
 - 'dropdowns' => refresh the dropdowns on the pagelet each time it is opened

brShowWYSIWYG

Proforma: brShowWYSIWYG('Widget');

Alias: showWYSIWYG

Show the WYSIWYG editor.

- Widget - string: the name of the widget to which a selected filename will be returned.

brTextSelected

Proforma: brTextSelected('widget');

Alias: getSelectedText

Get the highlighted text from a textarea. If no text is highlighted, all the text is selected.

- Widget - string: the name of the textarea widget.

brWrite

Proforma: brWrite('widget', 'value');

Alias: updateElement

Save the value of a widget.

- Widget - string: the name of a widget.
- Value - string: the value to save. All such values are held as string.

Supported Widgets

Breato supports a comprehensive set of widgets to help with presentation and interaction. We expand the set supported on a regular basis and are always delighted to consider requests.

The widgets below are shown in the order displayed in the dropdown list.

Form elements...

Label

A simple piece of text commonly used as a label.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container..
- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Label* - The text to be displayed.
- *State* - Whether the widget is enabled or disabled by default.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. *#80c0f0*) or as an RGB function call (e.g. *RGB(180, 140, 37)*).

Double click on the input box to display a palette.

- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget. This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Textbox

A standard text box.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Size* - Define the size of the input box.
- *Type* - The allowable data formats.
- *Default* - The default value. Note that this will be overridden by any default set in the data attribute definition.
- *Mandatory* - Whether the element is mandatory (i.e. must have a non-zero value).
- *Restriction* - The javascript to perform to ensure that the data entered are correct.

Examples:

- `{thiselement}.length > 10`
- `{thiselement} <={page.element}`
- *Warning* - The warning to display should the element fail the mandatory or restriction tests.
- *State* - Whether the widget is enabled or disabled by default.

- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Change* - The javascript to execute on a change event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Resize* - Specify how the object is resized with a container resize
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Textarea

A standard multi-line text input box.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Rows* - Define the number of rows to display.
- *Columns* - Define the number of columns to display.
- *Mandatory* - Whether the element is mandatory (i.e. must have a non-zero value).
- *Restriction* - The javascript to perform to ensure that the data entered are correct.

Examples:

- `{thiselement}.length > 10`
- `{thiselement} <={page.element}`
- *Warning* - The warning to display should the element fail the mandatory or restriction tests.
- *State* - Whether the widget is enabled or disabled by default.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Change* - The javascript to execute on a change event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Resize* - Specify how the object is resized with a container resize
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a

lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Dropdown

A standard dropdown list.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Size* - Select the number of items to display without scrolling.
- *Default* - The default value. Note that this will be overridden by any default set in the data attribute definition.
- *Restriction* - The javascript to perform to ensure that the data entered are correct.

Examples:

- `{thiselement}.length > 10`
- `{thiselement} <={page.element}`
- *Warning* - The warning to display should the element fail the mandatory or restriction tests.
- *State* - Whether the widget is enabled or disabled by default.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Change* - The javascript to execute on a change event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

- See the *Tip* section in the user's guide for more configuration options.
- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Combobox

A combo box allowing a new value to be specified or an existing value to be selected. This is a bespoke widget. Press return to select a value and to close the dropdown.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Default* - The default value. Note that this will be overridden by any default set in the data attribute definition.
- *Restriction* - The javascript to perform to ensure that the data entered are correct.

Examples:

- `{thiselement}.length > 10`
- `{thiselement} <={page.element}`
- *Warning* - The warning to display should the element fail the mandatory or restriction tests.
- *State* - Whether the widget is enabled or disabled by default.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Change* - The javascript to execute on a change event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Multiple select

A bespoke multiple selection box.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Restriction* - The javascript to perform to ensure that the data entered are correct.

Examples:

- `{thiselement}.length > 10`
- `{thiselement} <={page.element}`
- *Warning* - The warning to display should the element fail the mandatory or restriction tests.
- *State* - Whether the widget is enabled or disabled by default.
- *Source* - The multiselect object holding the source data. If used, the SQL definition for this multiselect box should be left empty.
- *Destination* - The multiselect object on which this widget relies, if any.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Highlight background* - Define the colour to use as the background on the current row.

The value can be expressed as a colour name (e.g. *red*) or as an RGB value (e.g. *#80c0f0*) or as an RGB function call (e.g. *RGB(180, 140, 37)*).

- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a

lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Scrolling select

A bespoke scrolling selection box.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Multiple* - Only multiple selections.
- *Restriction* - The javascript to perform to ensure that the data entered are correct.

Examples:

- `{thiselement}.length > 10`
- `{thiselement} <={page.element}`
- *Warning* - The warning to display should the element fail the mandatory or restriction tests.
- *State* - Whether the widget is enabled or disabled by default.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Highlight background* - Define the colour to use as the background on the current row.

The value can be expressed as a colour name (e.g. *red*) or as an RGB value (e.g. `#80c0f0`) or as an RGB function call (e.g. `RGB(180, 140, 37)`).

- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.

- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Listbox

A standard listbox showing more than one selection option at a time.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Size* - Select the number of items to display without scrolling.
- *Default* - The default value. Note that this will be overridden by any default set in the data attribute definition.
- *Restriction* - The javascript to perform to ensure that the data entered are correct.

Examples:

- `{thiselement}.length > 10`
- `{thiselement} <={page.element}`
- *Warning* - The warning to display should the element fail the mandatory or restriction tests.
- *State* - Whether the widget is enabled or disabled by default.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Change* - The javascript to execute on a change event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.

- *Mouseover* - The javascript to execute on a mouseover event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget. This is a shorthand way of displaying a tip and provides no configuration options.
- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Checkbox

A standard checkbox.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Value* - The default value.

- *Checked* - Defines the default *checked* state for the widget.
- *State* - Whether the widget is enabled or disabled by default.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Change* - The javascript to execute on a change event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Radio button

A standard radio button.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Value* - The default value.
- *Checked* - Defines the default *checked* state for the widget.
- *State* - Whether the widget is enabled or disabled by default.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Change* - The javascript to execute on a change event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Password

A standard password entry box.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Size* - Define the size of the input box.
- *Default* - The default value. Note that this will be overridden by any default set in the data attribute definition.
- *Mandatory* - Whether the element is mandatory (i.e. must have a non-zero value).
- *Restriction* - The javascript to perform to ensure that the data entered are correct.

Examples:

- `{thiselement}.length > 10`
- `{thiselement} <={page.element}`
- *Warning* - The warning to display should the element fail the mandatory or restriction tests.
- *State* - Whether the widget is enabled or disabled by default.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Change* - The javascript to execute on a change event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.

- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Text

Text derived from the data repository.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Overflow* - Defines whether the object may expand as required.
 - Visible means that the element will grow as required.
 - Auto means that the element is restricted to the size specified.
- *Text colour* - Define the text colour.

The value can be expressed as a colour name (e.g. *red*) or as an RGB value (e.g. *#80c0f0*) or as an RGB function call (e.g. *RGB(180, 140, 37)*).

- *Text size* - Define the text size.

The value can be expressed as a relative size (e.g. *smaller*) or as an absolute size (e.g. *12pt* - note the inclusion of the dimension).

- *Resize* - Specify how the object is resized with a container resize
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.

- *Keypress* - The javascript to execute on a keypress event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget. This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Free text

Free text defined and formatted by you.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *State* - Whether the widget is enabled or disabled by default.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Contents* - The text to display.
- *Overflow* - Defines whether the object may expand as required.
 - Visible means that the element will grow as required.
 - Auto means that the element is restricted to the size specified.
- *Text colour* - Define the text colour.

The value can be expressed as a colour name (e.g. *red*) or as an RGB value (e.g. #80c0f0) or as an RGB function call (e.g. RGB(180, 140, 37)).

- *Text size* - Define the text size.

The value can be expressed as a relative size (e.g. smaller) or as an absolute size (e.g. 12pt - note the inclusion of the dimension).

- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. #80c0f0) or as an RGB function call (e.g. RGB(180, 140, 37)).

Double click on the input box to display a palette.

- *Resize* - Specify how the object is resized with a container resize
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Hidden text

A hidden form element. this can be used to hold information which the rest of the container relies on without it being visible to the user.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *Value* - The default value.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.

Image

An image to display. The image can either reside on this installation, in which case it may be referred to with a relative URI, or elsewhere, in which case the fully qualified URL must be specified.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *State* - Whether the widget is enabled or disabled by default.
- *Source* - The source of the image.

Enter a filename or url directly or use the browse button to find an image in your local directory on the web server.

- *Alt* - Define the *ALT* text for an image, in case the image itself cannot be displayed.
- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. *#80c0f0*) or as an RGB function call (e.g. *RGB(180, 140, 37)*).

Double click on the input box to display a palette.

- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

- See the *Tip* section in the user's guide for more configuration options.
- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Decorative background

Simple decoration. This provides a simple box which can styled and coloured as required.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container..
- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget. This is a shorthand way of displaying a tip and provides no configuration options.
- *Resize* - Specify how the object is resized with a container resize
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Plain background

A simple background to use behind other widgets.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container..
- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. *#80c0f0*) or as an RGB function call (e.g. *RGB(180, 140, 37)*).

Double click on the input box to display a palette.

- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget. This is a shorthand way of displaying a tip and provides no configuration options.
- *Resize* - Specify how the object is resized with a container resize
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Group box

A grouping box with an optional label to group together widgets.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container..
- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Title* - The title for the groupbox.
- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. *#80c0f0*) or as an RGB function call (e.g. *RGB(180, 140, 37)*).

Double click on the input box to display a palette.

- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.

- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Resize* - Specify how the object is resized with a container resize
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Tab control

A simple tab control.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Resize* - Specify how the object is resized with a container resize
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.

- *Definitions* - A comma separated list of name:container pairs. The name appears on the tab.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.

Reporting...

Interactive Table

An interactive table. This is the mainstay of reporting and is often used as a gateway to a data maintenance dialogue.

This table allows datasets (based on an SQL statement and parameters supplied to it) to be shown and browsed.

Paging options are provided and the data can be further refined and sorted using the built-in filtering and ordering controls.

The layout of the table can be defined including column widths, column names, column and header colours, tooltips, column grouping and footers. The action to be taken on clicking a column can be defined and whether an "add" action is supported.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Overflow* - Defines whether the object may expand as required.
 - Visible means that the element will grow as required.
 - Auto means that the element is restricted to the size specified.
- *Max. records* - Define the maximum number of records to display at one time.
- *No data* - The message to display when no data are available.
- *Hide controls* - Determine whether to hide the table controls.
- *Disable sorting* - Determine whether to prevent column sorting.
- *Print title* - The title to display in the generated PDF.

This may contain embedded parameters and will be displayed in the pdf using H1 tags.

- *Preamble* - The preamble for a report. This can contain embedded parameters.

- *Footnote* - The footnote for a report. This can contain embedded parameters.
- *Resize* - Specify how the object is resized with a container resize
- *Synchronous* - Specify if the widget should be populated synchronously.

You should only set this to *true* when the widget depends on other actions occurring first.

- *Add dialogue* - Define the action to take when the Add button is clicked.

Leaving this blank will cause the add button to be hidden.

- *Columns to display* - Up to the given number of columns will be displayed per row.

A blank or 0 value means display all values.

Any columns not displayed are still available for use in onclick events and tips.

- *Columns to group by* - Group by the given number of columns, starting from the first column specified.
- *Column types* - Define the data type of each column in the table.

Specify the types as a comma-separated list.

The valid types are

- **n** - numeric
- **a** - alphanumeric
- **ai** - alphanumeric (case insensitive)
- **d** - date (and time)
- **t** - time
- **c** - currency
- **b** - boolean
- **i** - image
- *Column alignment* - Define the alignment of each column in the table.

Specify the alignments as a comma-separated list of column number :: alignment pairs.

The valid values are

- **l** - left
- **c** - centre
- **r** - right
- *Column names* - A comma-separated list of names to use at the top of each column displayed. If no name is specified for a column which is displayed, the SQL column name will be used.
- *Column widths* - A comma-separated list of column number:: width pairs. Widths must be defined in a mixture of px or % - a numeric value will be assumed to be px.
- *Header colours* - A comma-separated list of column number :: colour pairs.

Colours are defined by their names or as #codes (e.g. #ff0000 = red).

- *Column colours* - A comma-separated list of column number :: colour pairs.

Colours are defined by their names or as #codes (e.g. #ff0000 = red).

- *Column onclick* - A comma-separated list of column number and onclick event pairs.

Each pair should start with the number of the column displayed (starting from 0) followed by two colons (::) and then one or more valid javascript statements each one terminated with a semi-colon (;). Highlight the Javascript associated with a column to test it.

- *Column tips* - A comma-separated list of column number and tip text pairs.

Each pair should start with the number of the column displayed (starting from 0) followed by two colons (::) and then a string with one or more parameters embedded in it (e.g. Description: {description}).

- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Footer* - A comma-separated list of column number and footer pairs.

Each pair should start with the number of the column displayed (starting from 0) followed by two colons (::) and then a valid javascript statement terminated with a semi-colon (;).

The following built-in functions are provided:

- `table.max('column name')`
- `table.min('column name')`
- `table.sum('column name')`
- `table.avg('column name')`
- `table.value('column name', 'first | last | row index')`
- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. #80c0f0) or as an RGB function call (e.g. RGB(180, 140, 37)).

Double click on the input box to display a palette.

- *Header background* - Define the colour to use as the background on the header row.

The value can be expressed as a colour name (e.g. *red*) or as an RGB value (e.g. #80c0f0) or as an RGB function call (e.g. RGB(180, 140, 37)).

- *Odd rows background* - Define the colour to use as the background on odd numbered rows.

The value can be expressed as a colour name (e.g. *red*) or as an RGB value (e.g. #80c0f0) or as an RGB function call (e.g. RGB(180, 140, 37)).

- *Even rows background* - Define the colour to use as the background on even numbered rows.

The value can be expressed as a colour name (e.g. *red*) or as an RGB value (e.g. #80c0f0) or as an RGB function call (e.g. RGB(180, 140, 37)).

- *Highlight background* - Define the colour to use as the background on the current row.

The value can be expressed as a colour name (e.g. *red*) or as an RGB value (e.g. #80c0f0) or as an RGB function call (e.g. RGB(180, 140, 37)).

- *Auto refresh* - Whether the element can be refreshed automatically.

- *Period* - The period (in seconds) between auto refreshes.

0 disables the refresh.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Plain Table

A reporting table for which you control layout and functionality. You can embed "on" events by creating an "_onevent" (e.g. `_onclick`) attribute in the body of a node. The actual code to be executed should be encoded using the `brA2H` function.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. `report_page.org_dd` refers to the `org_dd` widget on the `report_page` container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: `list:id1:name1;id2:name2` In this case, the dropdown list would be populated with display values of
- *Overflow* - Defines whether the object may expand as required.
 - Visible means that the element will grow as required.
 - Auto means that the element is restricted to the size specified.
- *Synchronous* - Specify if the widget should be populated synchronously.

You should only set this to `true` when the widget depends on other actions occurring first.

- *No data* - The message to display when no data are available.
- *Entries per line* - Define the maximum number of entries to display per line.

A value less than 1 or a non-numeric value will be interpreted as 1.

- *Print title* - The title to display in the generated PDF.

This may contain embedded parameters and will be displayed in the pdf using H1 tags.

- *Resize* - Specify how the object is resized with a container resize
- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. #80c0f0) or as an RGB function call (e.g. RGB(180, 140, 37)).

Double click on the input box to display a palette.

- *Auto refresh* - Whether the element can be refreshed automatically.
- *Period* - The period (in seconds) between auto refreshes.

0 disables the refresh.

- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Chart

A simple chart of data you define. The chart is created server-side and is presented as an image. This has the advantage of the chart being readily printable, but means that there are no opportunities for functionality such as rollovers.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Chart type* - The type of chart to display.

- *No data* - The message to display when no data are available.
- *Synchronous* - Specify if the widget should be populated synchronously.

You should only set this to *true* when the widget depends on other actions occurring first.

- *Title* - The main title for the chart.
- *X-axis title* - The title for the x-axis.
- *Y-axis title* - The title for the y-axis.
- *Legend* - Show a legend
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Interactive chart

Client-side charting. This is provided by the Google Chart Visualization API. Only a subset of the possible visualizations is supported.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *No data* - The message to display when no data are available.
- *Chart type* - The type of chart to display.
- *Title* - The main title for the chart.
- *X-axis title* - The title for the x-axis.
- *Y-axis title* - The title for the y-axis.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Columns to display* - Up to the given number of columns will be displayed per row.

A blank or 0 value means display all values.

Any columns not displayed are still available for use in onclick events and tips.

- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Google App

Objects...

Embedded page

An embedded container within a container. Often used for displaying different reports based on a common set of selection criteria.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Overflow* - Defines whether the object may expand as required.
 - Visible means that the element will grow as required.
 - Auto means that the element is restricted to the size specified.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Resize* - Specify how the object is resized with a container resize
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

RSS

An RSS feed. The feed itself is formatted for you and fed to the browser.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. #80c0f0) or as an RGB function call (e.g. RGB(180, 140, 37)).

Double click on the input box to display a palette.

- *Bullet* - The image to be used as a bullet point.

Enter a filename or url directly or use the browse button to find an image in your local directory on the web server.

- *Source* - The source of the RSS feed.
- *Period* - The period (in seconds) between auto refreshes.

0 disables the refresh.

- *Overflow* - Defines whether the object may expand as required.
 - Visible means that the element will grow as required.
 - Auto means that the element is restricted to the size specified.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

WWW

A draggable and resizable browser window. Please note that some URLs will "hijack" the browser to present themselves at the top level.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Source* - The source URL.
- *Overflow* - Defines whether the object may expand as required.
 - Visible means that the element will grow as required.

- Auto means that the element is restricted to the size specified.
- *Resize* - Specify how the object is resized with a container resize
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Flash

An embedded flash movie player.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Source* - The source URL for the movie.
- *BG Colour* - The background colour for the movie.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Embedded object

Google Ads

A Google Ads widget. See the Google Ads site for more details.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and

underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Source* - The source URL of the element.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Google Maps

A Google Map interface. See the Google Maps site for more details.

If the item being presented has a latitude and longitude specified, that is used, otherwise the full address (including postcode) is used to provide an approximate location.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Interested in* - Define the names of objects in which this widget has an interest.

Specify the objects as a comma-separated list.

It is suggested that you use the names of data groups, although any name can be used.

- *Zoom* - Define the zoom level for the map (1 - 15).
- *Latitude* - Define the starting latitude.
- *Longitude* - Define the starting longitude.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Resize* - Specify how the object is resized with a container resize
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Directory listing

A directory listing of a directory under your local directory

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Directory* - Define the top level directory to be listed. This must reside under your local directory structure.
- *Synchronous* - Specify if the widget should be populated synchronously.

You should only set this to *true* when the widget depends on other actions occurring first.

- *Include* - Define a comma separated list of file extensions to include (e.g. *.xml,.csv* - note the inclusion of the '.').
- *Exclude* - Define a comma separated list of file extensions to exclude (e.g. *.xml,.csv* - note the inclusion of the '.').
- *Show date* - Show the last modified date for the file.
- *Sub directories* - Show sub directories.
- *Hierarchy* - Show the output as a hierarchy.
- *Auto refresh* - Whether the element can be refreshed automatically.
- *Period* - The period (in seconds) between auto refreshes.

0 disables the refresh.

- *Resize* - Specify how the object is resized with a container resize
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Directory tree

Directory file list

Controls...

Generic Button

A generic form button

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Label* - The button label.
- *State* - Whether the widget is enabled or disabled by default.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Save Button

A generic form button

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Label* - The button label.
- *State* - Whether the widget is enabled or disabled by default.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

New Button

A generic form button

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Label* - The button label.
- *State* - Whether the widget is enabled or disabled by default.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Delete Button

A generic form button

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Label* - The button label.
- *State* - Whether the widget is enabled or disabled by default.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Refresh Button

A generic form button

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Label* - The button label.
- *State* - Whether the widget is enabled or disabled by default.
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Inline Calendar

undefined

Popup calendar

An image to display. The image can either reside on this installation, in which case it may be referred to with a relative URI, or elsewhere, in which case the fully qualified URL must be specified.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *State* - Whether the widget is enabled or disabled by default.
- *Source* - The source of the image.

Enter a filename or url directly or use the browse button to find an image in your local directory on the web server.

- *Alt* - Define the *ALT* text for an image, in case the image itself cannot be displayed.
- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. *#80c0f0*) or as an RGB function call (e.g. *RGB(180, 140, 37)*).

Double click on the input box to display a palette.

- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget.
This is a shorthand way of displaying a tip and provides no configuration options.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.

- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

File browser

An image to display. The image can either reside on this installation, in which case it may be referred to with a relative URI, or elsewhere, in which case the fully qualified URL must be specified.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.

A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.

- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *State* - Whether the widget is enabled or disabled by default.
- *Source* - The source of the image.

Enter a filename or url directly or use the browse button to find an image in your local directory on the web server.

- *Alt* - Define the *ALT* text for an image, in case the image itself cannot be displayed.
- *Background colour* - Define the background colour.

The value can be expressed as a colour name (e.g. *red* or *transparent*) or as an RGB value (e.g. *#80c0f0*) or as an RGB function call (e.g. *RGB(180, 140, 37)*).

Double click on the input box to display a palette.

- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

- See the *Tip* section in the user's guide for more configuration options.
- *Context menu* - The javascript to execute on a context event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event. This may include references to elements present in the UI. Double-click on the input box to bring up the built-in javascript editor..
- *Tip* - Text to display as a tip when the mouse passes over the widget. This is a shorthand way of displaying a tip and provides no configuration options.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Diary

A diary display.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores. A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type. In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Default view* - The opening view for a diary.
- *Calendar* - Define the name of the calendar which is associated with this diary (if any).
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.

- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

- *Universe* - The source data repository.

Tree view

A collapsible tree representation of data.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *SQL* - The SQL to execute to populate the element. This should be SQL appropriate to the data repository type.
In the case of dropdown lists and the like, this can also be a static list of the form: *list:id1:name1;id2:name2* In this case, the dropdown list would be populated with display values of
- *Click* - The javascript to execute on a click event. This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Double click* - The javascript to execute on a double click event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Blur* - The javascript to execute on a blur event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Mouseover* - The javascript to execute on a mouseover event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..

Tooltips can be specified here by a call of the form **Tip('your text')**. Note the use of the single quotes.

See the *Tip* section in the user's guide for more configuration options.

- *Context menu* - The javascript to execute on a context event.
This may include references to elements present in the UI.
Double-click on the input box to bring up the built-in javascript editor.
- *Keypress* - The javascript to execute on a keypress event.
This may include references to elements present in the UI
Double-click on the input box to bring up the built-in javascript editor..
- *Resize* - Specify how the object is resized with a container resize
- *Visibility* - Select the default visibility.
- *Top* - Position of the top of the element in px, pt, em or %.

- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.
- *Display order* - Define the display order.

Widgets with a higher display order will be displayed on top of those with a lower order when they overlap. Widgets with the same display order will be displayed with the last added on top.

Horizontal resizer

A horizontal resizer user to resize widgets to the left and right.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Resize* - Specify how the object is resized with a container resize
- *Tip* - Text to display as a tip when the mouse passes over the widget. This is a shorthand way of displaying a tip and provides no configuration options.
- *Source* - A comma separated list of widgets (no container name) which are to be resized and which are on the left of the resizer
- *Destination* - A comma separated list of widgets (no container name) which are to be resized and which are on the right of the resizer
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.

Vertical resizer

A vertical resizer user to resize widgets above and below.

It has the following attributes:

- *Name* - A reference name for the widget. It is useful if the name is unique within the context of the container. For practical purposes the name should be short but meaningful and must consist only of letters, digits and underscores.
A widget can be uniquely referenced by its container name and its own name, e.g. *report_page.org_dd* refers to the *org_dd* widget on the *report_page* container.
- *Comments* - A description of what the widget is used for and what it represents.

This is used in the design documentation.

- *CSS Style* - Select a CSS style to control the appearance of the widget. If you have modified or added to the styles available, you must ensure that the style conforms to the general requirements for widget styles.
- *Resize* - Specify how the object is resized with a container resize
- *Tip* - Text to display as a tip when the mouse passes over the widget. This is a shorthand way of displaying a tip and provides no configuration options.
- *Source* - A comma separated list of widgets (no container name) which are to be resized and which are above the resizer
- *Destination* - A comma separated list of widgets (no container name) which are to be resized and which are below the resizer
- *Top* - Position of the top of the element in px, pt, em or %.
- *Left* - Position of the left of the element in px, pt, em or %.
- *Height* - The height of the element in px, pt, em or %.
- *Width* - The width of the element in px, pt, em or %.
- *Tab index* - Define the tab order.

Acknowledgements

In producing this software use has been made of the following open source, third party software:

Tips	Wz tooltips
Icons	FamFamFam
WYSIWYG editor	ckEditor