



**The Intelligent Management System
for your Facility**

User Manual

**August 2021
Calpendo 9.0.38**

Table of Contents

Foreword	0
Part I Getting Started	9
Part II Version History	11
2.1 Changes in Version 8.4	12
2.1.1 Changes For Regular Users	12
2.1.2 Changes For Administrators	13
2.2 Changes in Version 9.0	21
2.2.1 Changes For Regular Users	21
2.2.2 Changes For Administrators	22
Part III Calpendo User Guide	27
3.1 Getting A User Account	28
3.2 Reset or Change Password	31
3.3 User Settings	34
3.3.1 Calpendo User Settings	34
3.3.2 Appearance	37
3.3.3 Buttons	37
3.3.4 Date & Time	37
3.3.5 Email	38
3.3.6 Menu	38
3.4 Bookings	39
3.4.1 Bookings Calendar	39
3.4.1.1 Exploring The Bookings Calendar	40
3.4.1.2 Display Of The Bookings Calendar	51
3.4.1.3 Creating Bookings	53
3.4.1.4 Editing And Cancelling Bookings	58
3.4.2 Booking Restrictions	62
3.4.2.1 Time Templates	62
3.4.2.2 Booking Rules	65
3.4.2.3 Permissions	65
3.4.3 Searching For Bookings	66
3.4.3.1 Using The Booking Searches	66
3.4.3.2 Booking Cancellations	70
3.4.3.3 My Bookings	71
3.4.3.4 My Projects' Bookings	72
3.4.3.5 Booking Search	73
3.5 Projects	74
3.5.1 Creating Projects	75
3.5.2 Modifying Projects	78
3.5.3 Searching For Projects	78
3.5.3.1 Using The Project Searches	78
3.5.3.2 My Projects	82
3.5.3.3 Project Search	83
3.5.4 Project Membership Request	84
3.6 Resource Usage	85
3.6.1 Resource Usage Session Recorder	85
3.6.2 Resource Usage Calendar	88
3.6.3 Searching For Resource Usage	90
3.6.3.1 Using the Resource Usage Searches	90

3.6.3.2	My Resource Usage.....	93
3.6.3.3	My Projects' Resource Usage.....	94
3.6.3.4	Resource Usage Search.....	95
3.7	Services	96
3.7.1	Available Services	96
3.7.2	Searching For Orders	98
3.7.2.1	Using The Order Searches	98
3.7.2.2	My Orders	101
3.7.2.3	My Projects' Orders.....	103
3.8	Toolbar Button Standard Definitions	104
3.9	Conditions	107
3.10	Searching For Information	118
3.10.1	Search	118
3.10.1.1	Single Item Report.....	119
3.10.1.2	List Report.....	120
3.10.1.3	Summary Report.....	122
3.10.1.4	Group Report.....	127
3.10.1.5	Editing Search Information.....	129
3.10.1.6	Saving And Reusing Searches.....	131
3.10.2	Report Manager	133
3.10.3	History	138
3.10.4	Data Explorer	139
3.11	Frequently Asked Questions	145
3.12	Web Browser Compatibility	145

Part IV Calpendo Administration Guide 147

4.1	Booking Administration	148
4.1.1	Booking Properties	149
4.1.2	The Booking Approval Process	150
4.1.3	The Booking Requests Page	151
4.1.4	Read Only Calendar	155
4.2	Project Administration	155
4.2.1	Project Properties	156
4.2.2	The Project Approval Process	158
4.2.3	The Project Requests Page	159
4.3	User Administration	163
4.3.1	User Properties	163
4.3.2	The User Approval Process	167
4.3.3	The User Requests Page	167
4.3.4	User Search	172
4.3.5	Creating Users	176
4.3.6	Modifying Users	177
4.3.7	Changing A User's Settings	181
4.3.8	Special User	181
4.4	Bookmark Manager	182
4.5	Using Calpendo To Send Emails	186
4.6	Import	190
4.6.1	Example Import Files	190
4.6.2	The Import Page	197
4.7	System Events	204

Part V Calpendo Quick Start Configuration Guide 205

Part VI Calpendo Configuration Guide 209

6.1 Configuration Questions	210
6.2 Initial Configuration	213
6.3 More Calpendo Configuration	215
6.4 Converting From An Existing Booking System	217
6.5 Project Configuration	217
6.5.1 Project Template	218
6.5.2 Configuring Project Properties	218
6.5.3 Configuring The Project Approval Process	220
6.6 Controlling Bookings	221
6.6.1 Choosing Between Time Templates, Booking Rules and Permissions	223
6.6.2 Configuring Time Templates	226
6.6.2.1 How Time Templates Work.....	226
6.6.2.2 The Time Template Editor.....	229
6.6.2.3 The Time Templates Calendar	232
6.6.3 Configuring Booking Rules	235
6.6.3.1 How Booking Rules Work.....	235
6.6.3.2 The Booking Rules Editor.....	236
6.6.3.3 Choosing Which Bookings A Rule Applies To.....	238
6.6.3.4 Types of Booking Rule.....	243
6.6.3.4.1 Simple Booking Rule.....	243
6.6.3.4.2 Double Booking Rule.....	249
6.6.3.4.3 Booking Duration Rule.....	256
6.6.3.4.4 Holiday Booking Rule.....	257
6.6.3.4.5 Booking Interval Rule.....	258
6.6.3.4.6 Number Of Bookings Rule.....	259
6.6.3.4.7 Predefined Slots Booking Rule.....	262
6.6.3.4.8 Search Results Booking Rule.....	263
6.6.3.4.9 Total Time Booked Rule	264
6.6.3.4.10 Advanced Booking Rule.....	269
6.6.3.5 The Booking Rule Validator	277
6.6.4 Tentative Bookings	278
6.7 Types And Groups	279
6.8 Configuring Types And Groups	280
6.9 Configuring Resources And Locations	284
6.10 Configuring Services and Service Providers	291
6.11 Recording Actual Usage	293
6.11.1 pGina	296
6.12 Permissions	314
6.12.1 How Permissions Work	314
6.12.2 The Permissions Editor	322
6.12.3 Example Permissions	326
6.13 Workflows	334
6.13.1 How Workflows Work	334
6.13.2 Workflow Events	340
6.13.2.1 Anonymous HTTP Workflow Event.....	342
6.13.2.2 Booking Rule Workflow Event.....	344
6.13.2.3 Custom Function Workflow Event.....	347
6.13.2.4 Database Workflow Event.....	348
6.13.2.5 Privileged Search Workflow Event.....	350
6.13.2.6 Process Workflow Event.....	351
6.13.2.7 Relative Time Workflow Event.....	353
6.13.2.8 Reminder Workflow Event.....	354

6.13.2.9	Remote Authorisation Request Workflow Event.....	355
6.13.2.10	Remote User Identification Workflow Event.....	356
6.13.2.11	User Workflow Event.....	357
6.13.2.12	User Login Workflow Event.....	363
6.13.2.13	Timed Workflow Event.....	365
6.13.3	Workflow Actions	365
6.13.3.1	Biskit Create Workflow Action.....	370
6.13.3.2	Biskit Delete Workflow Action.....	370
6.13.3.3	Biskit Update Workflow Action.....	371
6.13.3.4	Create Variables Workflow Action.....	371
6.13.3.5	Delay Workflow Action.....	372
6.13.3.6	Diff Workflow Action.....	372
6.13.3.7	Email Workflow Action.....	373
6.13.3.7.1	Example Email Workflows	379
6.13.3.8	Evaluate Expression Workflow Action.....	384
6.13.3.9	Execute System Command Workflow Action.....	385
6.13.3.10	Find Text Workflow Action.....	386
6.13.3.11	For Each Workflow Action.....	387
6.13.3.12	Function Workflow Action.....	388
6.13.3.12.1	Biskit Function Type.....	389
6.13.3.12.2	Calendar Function Type.....	402
6.13.3.12.3	Conversion Function Type.....	405
6.13.3.12.4	Date & Time Function Type.....	416
6.13.3.12.5	File Function Type.....	432
6.13.3.12.6	List/Set Function Type.....	435
6.13.3.12.7	Logical Function Type.....	446
6.13.3.12.8	Mathematical Function Type.....	448
6.13.3.12.9	Miscellaneous Function Type.....	466
6.13.3.12.10	Network Function Type.....	470
6.13.3.12.11	Text Function Type.....	474
6.13.3.13	List Extract Workflow Action.....	476
6.13.3.14	Network Message Workflow Action.....	476
6.13.3.15	Search Workflow Action.....	478
6.13.3.16	Simple Workflow Action.....	479
6.13.3.17	Sort Workflow Action.....	479
6.13.3.18	Templated Text Workflow Action.....	482
6.13.3.19	Type Cast Workflow Action.....	484
6.13.3.20	Veto Workflow Action.....	485
6.13.4	The Workflow Editor	485
6.13.5	Workflow Use Cases	488
6.14	Menu Editor	489
6.15	User Authentication Methods	499
6.15.1	Authentication Methods	499
6.15.2	Authentication Methods Editor	505
6.16	FAQ Editor	506
6.17	Global Preferences	509
6.17.1	Appearance	510
6.17.2	Booking Reminders	515
6.17.3	Bookings	515
6.17.4	Buttons	521
6.17.5	Date And Time	522
6.17.6	Email	523
6.17.7	General	525
6.17.8	Licence	526
6.17.9	Menus	527
6.17.10	Network Metrics	528
6.17.11	Projects	529
6.17.12	Resource Usage	530

6.17.13 Rules	531
6.17.14 Security	532
6.17.15 System Usage Statistics	534
6.17.16 Time Templates	535
6.17.17 Users	536
6.18 Bakery	537
6.18.1 Property Storage Mechanisms	537
6.18.2 Biskit Definitions	539
6.18.3 Property Definitions	544
6.18.3.1 Biskit Property Definitions.....	549
6.18.3.2 Boolean Property Definitions.....	550
6.18.3.3 Double Property Definitions.....	551
6.18.3.4 Int Property Definitions.....	552
6.18.3.4.1 Mapped Integers.....	552
6.18.3.4.1.1 Bit Sets	554
6.18.3.4.1.2 User Roles	554
6.18.3.5 Java Enum Property Definitions.....	555
6.18.3.5.1 Java Enum Definitions	555
6.18.3.6 Set Property Definitions.....	556
6.18.3.7 String Property Definitions.....	558
6.18.3.7.1 Mapped Strings	562
6.18.3.8 String Enum Property Definitions.....	563
6.18.3.8.1 String Enumerations	564
6.18.4 Formulae	564
6.18.5 Tag Properties	567
6.18.6 Bakery Editor	568
6.18.6.1 Bakery Editor Examples.....	574
6.18.6.1.1 Adding Properties.....	574
6.18.6.1.2 Adding Formulaic Properties	577
6.18.6.1.3 Adding Properties For File Attachments.....	580
6.18.6.1.4 Adding Properties For Created/Updated/Version.....	582
6.18.6.1.5 Adding A New Yes-No Mapped Int Property	584
6.18.6.1.6 Creating A New Basic BiskitDef.....	587
6.18.6.1.7 Creating A Set Of Biskits (Attachments).....	588
6.18.6.1.8 Creating A Hierarchy Of BiskitDef.....	592
6.18.6.1.9 Creating A Master-Slave Biskit Relationship.....	597
6.18.6.1.10 Creating An Inheriting BiskitDef (Booking).....	602
6.19 Layout Editor	603
6.20 Processes	622
6.21 Backing Up The Database	628
6.22 Creating A Read Only Copy	630

Part VII API 633

7.1 REST API	634
7.2 Query API	635
7.3 Anonymous HTTP API	639
7.4 General Client HTTP API	640

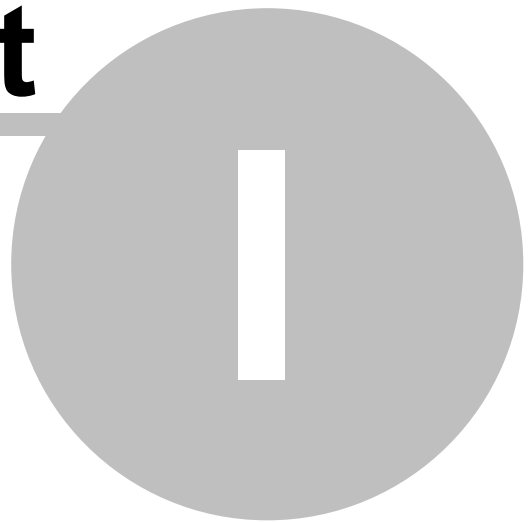
Part VIII Calpendo Installation Guide 641

Part IX Glossary 651

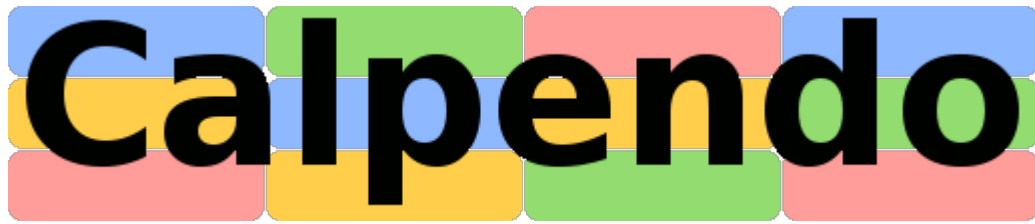
Index

657

Part



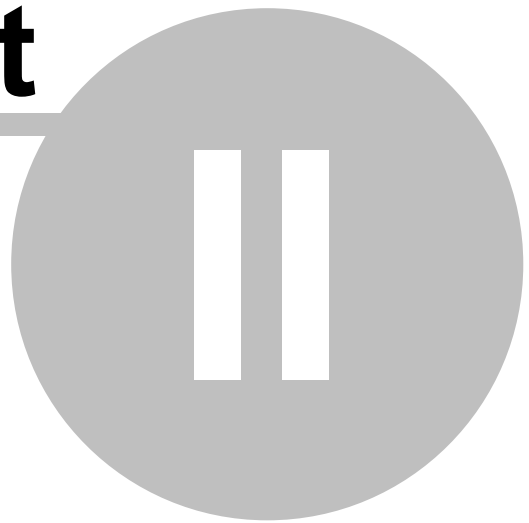
1 Getting Started



Calpendo is an intelligent booking system that applies [Booking Rules](#)⁶⁵² to control who can book what and when. It manages information on [projects](#)⁶⁵⁴ associated with each [booking](#)⁶⁵² and automatically sends emails when required.

This documentation was written for version 9.0.38 of **Calpendo** and was last changed on August 28, 2021.

Part



2 Version History

This section shows the history of changes over the last few versions of **Calpendo** .

2.1 Changes in Version 8.4

Version 8.4 provides a large number of improvements to workflows, in particular with many new functions being added.

To view changes to other releases go to our release notes at the web site at <http://www.exprodo.com/category/release-notes/>

2.1.1 Changes For Regular Users

New Functionality

Changes in Functionality

Calendar

The button bar that appeared in the "Resources" section on the left of the bookings calendar has changed. In 8.3, the functionality of the old "Bookmarks" section moved from the calendar to be in the "Resources" section, and it became a row of icon buttons.

Some people were confused about the buttons and what they did, and it was also difficult to explain to somebody which button to press.

So its been changed again. This time, there's a single button, labelled "Bookmarks", for everything related to bookmarks and choosing the displayed resources, and when clicked, it shows a drop-down menu with the selectable options. This means it now provides both the icons and text that describes what they do.

Add support for users to hide the page banner

- When a system displays a page banner at the top of every page, there's now a small button in the top left corner to hide it.
- A cookie remembers whether its hidden or not, so it will stay hidden after refreshing the browser until the button is clicked again.

Project Owners can create Bookings with their Project

Non-admins can now select projects they own in the booking pop-up, even if they are not in the users for that project.

Summary Report Can Have Multiple Rows

When creating a Summary Report it may now have multiple Rows, allowing a combination of Summary and Group Report.

2.1.2 Changes For Administrators

New Functionality

Workflows

- Add support for workflows to send emails and choose the to/cc/bcc recipients separately. Previously, all recipients were in the "to" section of the email.
- Add support for being able to listen to database events after they have completed.
 - Currently, workflows can only run while the event is in progress, which means the original event can be vetoed.
 - Sometimes this is useful, such as being able to prevent a booking, but it's also sometimes undesirable if an error in your workflow prevents something being saved.
 - With this new option, it is guaranteed that the workflow cannot impact the change being listened to (which is in some cases desirable).
 - It also means changes cannot be vetoed using this mechanism.
- Add support for a new User Login Workflow Event, triggered when a user logs in.
 - This will allow the system to customise aspects of the user's settings at login time.
 - In particular, modifying the menu a user will use, so that it can be ensured that those with the admin role always get the admin menu.
 - It is also possible to run a VetoWorkflowAction to prevent the user from logging in.
 - It is also possible to which layouts the user will see. Do this by calling the new workflow function addUserLayout, passing in the layout to be used by the user, and the layouts given on the UserLoginWorkflowEvent.
- Add support for TemplatedTextWorkflowAction to parse FreeMarker templates
 - The template provided to a TemplatedTextWorkflowAction provides values that are replaced by references to particular values from the workflow's context.
 - That still happens, and is the first step towards converting the template to the output text.
 - There's now an additional step, if there are any FreeMarker directives in the template, then the template is now also run through FreeMarker.
 - There are values provided for the FreeMarker data model, which can all be accessed using standard FreeMarker syntax. An administrator doesn't need to remember how to do this though - the UI's "Insert" button has a new item that appears in the drop-down. One for selecting a value that FreeMarker will parse, and another for creating a FreeMarker loop through a list of biskits.
 - For example, this is a valid FreeMarker template, where event #1 is a database event for a booking being created:
 - Hello there \${METAPROPERTIES.user.userIdentity.loginName}, booking made for \${source1.biskit.resource.name}

- Add support to expression workflow action for logical combinations of booleans. This means the expressions below can be used:

Logical AND:

a && b

a AND b

a and b

Logical OR:

a || b

a OR b

a or b

Logical Exclusive OR:

a ^ b

a XOR b

z xor b

Logical NOR:

a NOR b

a nor b

- addUserLayout can only be used as part of a UserLoginWorkflowEvent, and the function will add a layout to those that the user will see. This is used to specify particular layouts for a user so that the way data is presented can be controlled. For example, it may be required to hide the existence of some properties from some users, rather than only hiding the value of those properties.
- expandRepeats takes a repeating item (template, booking etc), or a list of them, and returns a list of the repeating instances.
 - Specify either the start and end dates, or the start date and a number of days.
- intersect was a function that takes two DateRange values and works out the DateRange representing the period they have in common. There are now two more versions of this method. One takes a DateRange and two date/times, and the other takes four date/times.
- createDateRange creates a new DateRange value. There are two versions of it, one that takes no arguments and gives you a range from now to December 2999 and one that takes two date-time values.
- timeDiffSeconds returns the difference in seconds between two times.
- timeDiffMilliseconds returns the difference in milliseconds between two times.
- Add overloaded version of workflow function getBookingTemplates
 - The new version doesn't take a booking as an argument, but all the details that it would need from a booking.
 - That means resource, booker, project, start and finish

- The name of the function added for a CustomFunctionWorkflowEvent that declares that its function can only be called from within its own workflow will now be limited in scope to that workflow.
 - It was already callable only from that workflow.
 - Now it will not be considered to clash with a function defined outside the workflow that has the same name.
 - A reference to a function will now look for something scoped within the workflow where you're defining the call to the function, and then if it doesn't find it there, it will look in the global scope.
 - This means it is possible to copy-and-paste a workflow that contains a CustomFunctionWorkflowEvent, where previously the save was rejected because it would be creating a new function of the same name as before.
 - This copy-and-paste will only work where the CustomFunctionWorkflowEvent declares its function can only be called from within the workflow. Otherwise, the name of the function will have to be modified before being allowed to save it.
 - Note this means an alternative implementation of a global function can be provided, even one provided with the system, but in order to do that, a CustomFunctionWorkflowEvent must be created with both a matching function name and also matching input and output names as well. In other words, the names of the inputs and outputs are seen as an integral part of the function-matching system.
- ✍️ *Aside: Doing it this way is a design choice made because it allows us to automatically modify any actions that call a function when that function's signature is modified by adding, removing or reordering the parameters. For this to work, the names of the arguments are used to detect the detail of the underlying change and how to modify things accordingly. We take the view that it is better to support automatic modification of user workflows when we make changes to built-in functions than it is to allow custom functions to override functions without a care for input or output names.*
- Add support for expression workflow action to create a system event with details of every partial expression evaluated when in verbose mode
 - Turn on verbose mode for EvaluateExpressionActionHandler and there will be an event as each part of an expression is evaluated.
- Add new "References" tab to the display for each workflow event and action
 - This shows which other events/actions reference this one (the inbound references) and which other events/actions are referenced (the outbound references).
 - When the mouse hovers over each reference, the item is highlighted in the tree of actions and items.
 - Click on a reference to go directly to that item.
- Perform validation of a workflow and trigger every time they are rendered
 - This means that if there are any inconsistencies within a workflow, event or trigger, just clicking on it or saving it will display an error message.

- When action succeeds but won't run its children, mark system event as such
 - This makes it easier when looking at system events to see whether a `TypeCastAction` matched or not.
- Display good properties for `CreateVariablesAction` in system event
- Add collection size to `ForEachWorkflowAction` output
 - When a foreach is run and loops through its contents, the output of the foreach now shows the number of items in that collection as well as the current index.
- Add function name and signature to output of function workflow action
- Display details of output from evaluate expression in system event
- Add workflow function `generateUserWorkflowEventURL`, this allows links to be emailed that will cause a `UserWorkflowEvent` to be run.

Read Only Calendar

It has been possible to display a read-only bookings calendar for quite some time now, but it wasn't easy for people to use this because it needed a suitable workflow to be created.

Version 8.4 now comes with a default workflow that provides anonymous http access to display a read-only calendar. That means it can be seen without logging in to Calpendo.

The workflow is disabled by default. The minimum require to get it working is:

- Go to the Workflow Manager
- Find the workflow, under "Booking" and called "Weekly Calendar"
- Edit it and enable the workflow, and save it
- Point a web browser at:
 - <https://yourcalpendo/anon/bookings.html?title=Your Title&pks=1,2,3,4>

where the arguments in the URL to control the displayed title may be changed and the IDs of the resources whose bookings should be displayed.

It is possible to see an example of this by going here:

<https://demo.calpendo.com/anon/bookings.html?title=Custom%20Calendar&pks=1,2,3,5,6>

The displayed data will reflect the permissions of a particular configured user, and that user is by default "admin". This may not be what is required, so if this is turned on, it should be configured to make sure it's not leaking information.

Grouping of Related Items In the Bakery

- Add support for `MappedInt`, `MappedString` and `StringEnumDef` groups in the Bakery
 - This means that items in the bakery can be grouped so that they can be seen to be related.

Tag Properties

Calpendo now supports "tags". These are either string properties or properties that store a set of strings, and any value may be chosen for the string, but the user will automatically be offered values that have previously been used for the same tag.

An administrator might want to prevent certain values being offered as suggestions, and to do this you would search for "Tag" biskits, and modify the relevant one to mark it as deprecated. Alternatively, they could just delete it.

To set this up, in the Bakery there is now a new top-level item labelled "Tag Def" where the flavour of tags that exist can be defined.

To have a single tag property, add a string property in the Bakery and set the string type to "Tag", and then choose the type of tag it is, which will be one of those created in the "Tag Def" section.

To add a property to store multiple tags, add a "set" property, and specify the set contains items of type "string", and "Tag" will automatically be defined as the type of string. Again, choose the type of tag it is, which will be one of those created in the "Tag Def" section.

Choose Which Roles Can Book For Any Project

- Historically, those with the admin role have been able to select any project when making a booking.
- Global preferences now has an option to choose which roles provide this ability.
- This means you can set up any role as meaning that users with that role can select any project in the booking pop-up.

Display Only Authorised Projects When Admin/Resource Manager Makes Bookings

- When an admin makes a booking for a resource that is marked as requiring authorised projects only, we used to show ALL projects.
- This made it difficult for admins who had to pick one that was authorised.
- It now only shows those that are authorised, when the resource says it requires it.
- It was already working this way for non-admins.

Support for Location Based Cascading Menus

- When there is a cascading selection of locations, possibly with many such locations, it is possible to now add a single menu item that will automatically expand out into a cascading set of menu items for users.
- Each menu item will let user display a bookings calendar with the resources in the location they select from the menu.

New Bakery Tool To Find References to Property Paths

This is found in two places:

- On each PropertyDef there's a new "References" button that finds all references to this PropertyDef or any reference to a string property path that might refer to properties with the same name as this PropertyDef.
- Under the "Tools" option at the top-level of the bakery, there's a tool to help locate references to any property path. Type in the name of a property (or something that used to be a property) or the name of a path that might have been used somewhere, such as in conditions. This could be something like "New.owner" or other things with a period in them as well as a simple property name.

Add Auto-Refresh to Calenders

All calendars can now automatically refresh every so often

- The time between refreshes is defined in a **Global Preference** (in the tab labelled "**Security**") and a user setting (in a new tab labelled "**Refresh**").
- By default, user settings will indicate it's the global preferences setting that should be used for the time between automatic calendar refreshes.
- The automatic refresh only happens while a calendar page is displayed.
- If the user visits a calendar page, then goes to a different page for a long time and goes back to the calendar, it will automatically refresh when the user goes back to the calendar.
- If you leave a calendar page displayed and do not interact with it for long enough so that the session expires, then it will automatically switch to the login screen.

Add Support For Pictures of Resources

- In the resource editor, set the picture in the "Picture" tab
- On the bookings calendar, the head of each column that shows the name of the resource now shows a picture when you mouse hovers over it.

Add New Suspended Status For Users

- The new **Suspended** status allows users to be able to login, but not count towards your user limit. These users will not receive emails but on login will have their status changed to **Normal**.

Add Support For Logging SMTP Transactions

This provides an option for logging the details of the transaction with an SMTP server, and can be useful in identifying problems with sending emails.

If this is enabled, then every time Calpendo tries to send an email, the system event generated will record the log of the communication with the SMTP server.

This may be helpful for debugging, but is a security risk because the entire content of all emails sent will be recorded in the system events.

So you should:

- not turn this on unless absolutely required
- not leave it turned on after its been finished with
- manually delete any system events containing sensitive email logs if it is turned on

Changes in Functionality

Authentication

Additional authentication options are available that can delegate the authentication to a separate server. This is supported in two new ways:

- an internal authentication method in which a user gives their login name and password to Calpendo, and this is validated by a separate Calpendo server.
- an external authentication method in which the password is never seen, but authentication is delegated to a proxy server. In this configuration, a central Exprodo Software ID server is used for the authentication, and that central ID server is configured for Shibboleth.

The end result of these new options is that we can delegate authentication to proxy servers, and that we can add support for Shibboleth authentication much more easily than we could previously.

Add support for attributes published by a single-sign-on provider to automatically populate user information that appears in the user registration form.

- When installing Calpendo on an external server, this requires the Apache configuration to capture Apache environment variables and make them available as http headers.
- The shibboleth External Proxy authentication run by the Exprodo Software service at <https://sp.exprodo.com/> exposes many standard attributes as http headers, although each identity provider that is dealt with may choose to expose each of those attributes to us or not.
- The external authentication in Calpendo can now be set up to provide a mapping from an http header to any string property of a user.
- Multiple headers may be mapped to the same user property, in which case the first one which is actually populate with a value is the one that will be used.

Add support for ExternalProxy authentication to choose a fixed identity provider

- External Proxy Authentication method can now be told which identity provider to use. This avoids the need for users to select their institution when they register and when they log in.

- Therefore it is possible to configure one authentication method specific to the institution and another that doesn't specify the identity provider.

Cancellation Comments

When canceling bookings, there is now an option to provide free-text comments to explain why it is being canceled, as well as the existing facility for choosing from a customisable drop-down of reasons.

iCal Feed

- Add support for anonymised iCal feeds
 - All existing iCal feeds will no longer work
 - Each user wanting a feed must ask Calpendo for a URL
 - Each user is given a different URL for the same underlying data
 - The URLs used now include a long random string so that it is hard to guess an active feed URL.
- Add special user called ical_viewer
 - ical_viewer is a new user that will exist in all Calpendos, and you can't change its name, roles or status.
 - This user cannot log in and so does not count towards the number of licensed users.
 - When generating an iCal feed, it is this new user whose permissions are checked. That means that if you want to hide something from the iCal feed, you would set permissions so that user ical_viewer cannot read the relevant information.
 - All existing and future Calpendos are now given permissions to prevent ical_viewer from seeing who made or owns bookings and also the names of resources.
 - If this information is required to be available in the iCal feed, then permissions will need to be changed to open it up.
- Disable ical feeds
 - This is a precaution following recent security issues and corrects the problem that ical feeds have historically been enabled by default.
 - This reverses that and requires everybody to turn the feed back on if it's required.
- Add support for choice of permissions user when download single booking
 - From the bookings calendar, you can click on a booking and select the option to download it in ical format so you can add it to some other calendar.
 - There's now a global preferences option to choose whether to control the content of the downloaded data according to permissions for the user doing the downloading or the special ical_viewer user.
 - Normally, the ical_viewer user would be the most restrictive, and this is the default.

2.2 Changes in Version 9.0

Version 9.0 provides a large number of improvements to speed and security.

To view changes to other releases go to our release notes at the web site at <http://www.exprodo.com/category/release-notes/>

2.2.1 Changes For Regular Users

New Functionality

Skins/Themes

Allows the user in User Settings choose what skin/theme they want to use as well as adding their own CSS.

Changes in Functionality

Display as much of the menu as possible

- Previously, if the menu was wider than the page, then the whole menu would be replaced with a "Hamburger" style icon that let the user see the whole menu when you clicked it.
- Now, it only hides as much of the menu as is required for it all to fit in, with the hamburger there to provide access to those menu items that had to be removed to make it fit.

2.2.2 Changes For Administrators

New Functionality

Communication

Version 9.0 has completely replaced the mechanism used for communicating between the browser and the server. This has been done for several reasons:

- Future-proofing due to the third party software we are using being likely to remove support for the communication method we were using.
- Making development easier by making the messages between browser and server human-readable.
- Allowing us to remove some server-side code that was used to make the old mechanism work, but which had large negative performance consequences. That is, it slowed things down and caused the server to use more memory than it would otherwise have required.
- Version 10 (Enterprise) requires inter-database communication so that one Calpendo can be set up as a slave to another. The new communication protocol between browser and server will be used for the server-server communication required by Enterprise.

Speed

Some systems have become much slower over time. Version 9.0 completes the major work that was begun with version 8.4 to address these speed issues. Those suffering with poor performance should find that 9.0 has a dramatic effect on the feel of the system.

While there will continue to be individual operations that can be tuned to perform better, it is already much faster than before in many places.

Workflows

- Add a workflow function to convert a number from one unit to another called `convertUnits`.
- Add a "Run Now" button to `UserWorkflowEvent` in the workflow manager
 - This makes it a little easier to test a user workflow event by providing a button in the workflow manager to run it.
- Add support for converting XSL-FO formatted strings to PDF
 - There's a new workflow function called `"toPDF"`
 - Give it an XSL-FO formatted string, and it returns the PDF file
 - There are some "Hello World" type examples in the workflow manager. Open Workflow Biskit/Versioned Workflow Biskit/Workflow/Example PDF generation with XSL FO
 - XSL-FO tutorials and information:

 <http://www.herongyang.com/XSL-FO/>

22

- ✎ <http://w3schools.sinsixx.com/xslfo/default.asp.htm>²²
- ✎ <https://www.webucator.com/tutorial/learn-xsl-fo/index.cfm>²²
- ✎ <https://www.antennahouse.com/comprehensive-xsl-fo-tutorials-and-samples-collection/>²²
- ✎ <https://www.qctutorials.com/learning/xslfo/index.html>²²
- ✎ <https://www.ibm.com/developerworks/library/x-xslfo2app/?ca=dnt-46h>²²
- ✎ <http://www.cafeconleche.org/books/bible3/chapters/ch16.html>²²
- ✎ <http://www.renderx.com/tutorial.html>²²
- The standard is described here:
 - ✎ <https://www.w3.org/TR/xsl/>²²
- Add support for barcodes
 - PDFs can include barcodes
 - ✎ See the XSL FO workflow example mentioned above for a "Hello World" type example.
 - ✎ For examples of all the types of barcode available, see <http://barcode4j.sourceforge.net/examples.html>²²
 - ✎ The format of the XML required inside the XSL FO file is described at <http://barcode4j.sourceforge.net/2.1/barcode-xml.html>²²
 - A string-valued property can be configured to display its content as a barcode in the browser.
 - ✎ In the bakery, choose a String Property Type of "Barcode"
 - ✎ Then select one of the barcode types
 - ✎ These properties are inherently read-only which means they would generally either be formulaic properties (that is, generated from other properties) or be created by a workflow.
 - ✎ For details on the different types of barcode available, see <https://github.com/lindell/JsBarcode/wiki>²²
 - Note that the PDF barcodes (generated by Barcode4j) have a different set of barcode types from those available in the browser (generated by JsBarcode). In particular, the PDF barcodes can include 2D barcodes in QR format whereas those in the browser cannot.

Reports

- Add a "Run Now" button to Report Manager so that when there's a scheduled report, it can be made to run now and send its result via an email.
- Add a preamble at the start of a scheduled report to control the text that appears in the email.

Add Support for Skins and Themes

- Global preferences now has a new tab under Appearance
- Allows the selection of skins and/or themes to load.
- This provides an easy way to customise how everything looks.
- Select the skins and/or themes to include, save and refresh your browser.
- Each user also has personal settings where they can select skins and/or themes and provide their own personal CSS. This allows each user to customise the way their own system looks without affecting other users.
- Create a new biskit of type "Theme", and once saved, the custom theme can be selected in global preferences or by individual users.
- Change the default look of all tables to be much cleaner
- This is done using the new support for skins.
- If prefer the old look, add the "uncleanTables" skin in global preferences.

Add Support for Stepped-Editing

- Add support for stepped-editing of biskits with one-to-many properties
 - This is required to properly support sample tracking.

Changes in Functionality

Menu Editor

The menu editor has had a big makeover. It is now consistent with the look and feel of the rest of Calpendo rather than doing things in completely unexpected ways. It used to have a button bar at the top and the bottom, with the one at the bottom affecting menu items and the one at the top affecting whole menus.

It now has a list of the existing menus on the left, and you click on a menu to view or edit, like our other editors.

Holiday Date

Now has a description property.

Services

Has a property to disable a service therefore hidden from available services.

Properties not in Biskit Detail, Allow check editing

- Allow properties hidden in biskit detail in checked edit (if use layout)
 - Normally, a property that is marked as hidden in biskit detail is not shown when looking at checked editing.
 - Now, a layout is provided which includes such a property, it will appear in checked editing.
 - If a layout not provided, it is still hidden as before.

Password Checking

- Add support for checking whether passwords have been exposed in a security breach
 - When a user sets a new or initial password, we can now check whether that password has been exposed in a security breach.
 - There's a new global preferences setting on the Security tab for this. The administrator can choose a threshold which is the number of times a password has been exposed, beyond which the password should be rejected.
 - By default, this is turned on for everybody and the threshold set to one. This means a password will be rejected as long as it has been exposed at least once in a security breach.
 - Note that this method rejects passwords that have been exposed in a security breach and does not directly check password complexity. That means it is possible to reject a complex password and accept a simple one.

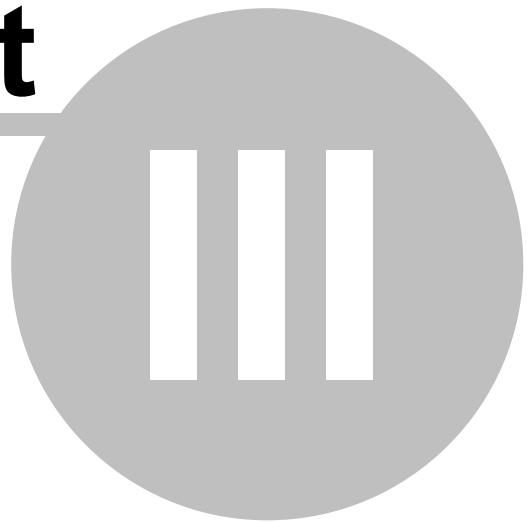
External iCal Feed Support

- Add support for importing bookings from an external iCal feed
- Resource now has two extra properties - one to record the URL and another to specify the update frequency.
- Booking has a hidden property that records the iCal ID that generated the booking.
- If an iCal feed contains a repeat booking, then it will be imported as multiple non-repeating bookings, all having the same iCal ID.
- Users cannot create, update or delete bookings that come from an iCal feed. The iCal importer takes complete control of the bookings.
- If a booking that used to exist in an iCal feed is later removed from the feed, then Calpendo will delete the booking for it.

Support For Linking Bookings And Service Orders

- This provides support for a convenient way to specify bookings for instruments that will be used in fulfillment of a service order.
- To configure this, start by adding one or more properties to a service order in the bakery to store a reference to a booking.
- Then, when configuring a service, after the type of biskit has been selected to use for its service orders, it will show a table with a row for all the Booking properties on the service order so that the resource to be used for that Booking can be chosen.
- When entering the service order, there will be a button labelled "Select Booking" which allows a booking to be chosen for the appropriate resource.
- If the service order has a project, then the user will only be offered bookings that are set up to use that project.
- When an order is updated so that it references a booking, there is a check to see if the booking has a reference to a Service Order. If so, then the booking will be updated to reference the order it relates to.

Part



3 Calpendo User Guide

This guide explains how to use **Calpendo** to see or make [bookings](#)⁶⁵², create or edit [projects](#)⁶⁵⁴ and how to search for information such as how much time a user has booked altogether.

3.1 Getting A User Account

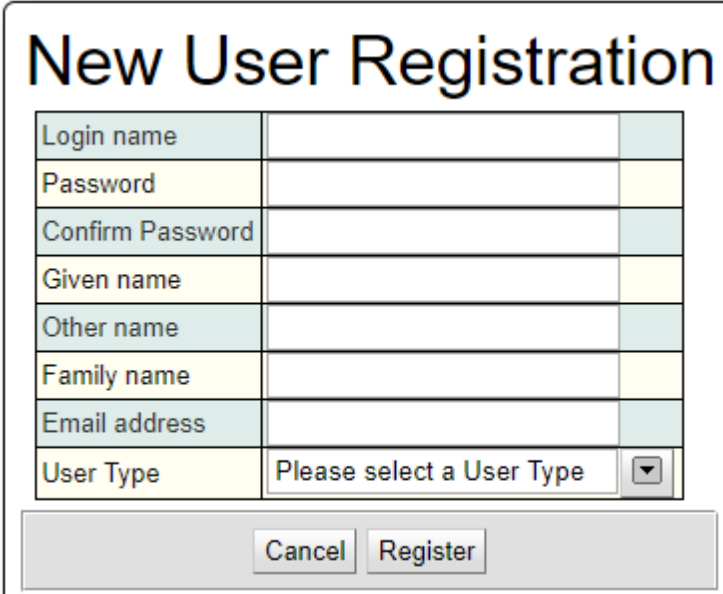
Local Calpendo Account

To get a local user account, first open a web browser. **Calpendo** supports the most popular web browsers - see [Web Browser Compatibility](#)¹⁴⁵ for the full list. The administrator should have told you the address of the **Calpendo**. When **Calpendo** is started, the opening screen will look something like this:



The image shows the Calpendo login screen. At the top, the word "Calpendo" is displayed in a large, bold, black font, with "Version 7.0.0" centered below it in a smaller font. Below the title, there are two input fields: "Username:" followed by a text box, and "Password:" followed by a text box. At the bottom of the form, there are two buttons: "Login" and "Register new user".

Press the **Register new user** button to see the **New User Registration** form shown here:



The image shows the "New User Registration" form. The title "New User Registration" is at the top in a large, bold, black font. Below the title is a table with the following fields:

Login name	<input type="text"/>	
Password	<input type="password"/>	
Confirm Password	<input type="password"/>	
Given name	<input type="text"/>	
Other name	<input type="text"/>	
Family name	<input type="text"/>	
Email address	<input type="text"/>	
User Type	Please select a User Type	<input type="button" value="v"/>

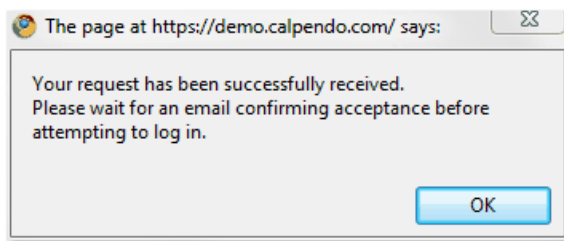
At the bottom of the form, there are two buttons: "Cancel" and "Register".

Enter the desired login identifier, which can consist of letters, numbers, and other characters such as @!:/(). Enter the password that is required by the user with conformation, then enter the given name (usually the first name) and family name (usually the last name). Finally enter the required email address. It is important that the email address is entered correctly because **Calpendo** will send an email to confirm it has received the registration request and again to inform when or if the request has been approved and the user can log in. If User Types have been configured you will need to choose one.

Depending on how **Calpendo** has been configured, other information may need to be selected.

When all values are entered, press the **Register** button.

If the requested login name, password and email address are acceptable, then a message like this will appear:



This means that the registration request has been received, but the user cannot log in yet. They will receive an email to tell them when they can log in using the login name and password selected. Note that the administrator will not know any passwords, nor have any way to find out what they are. However, the administrator is able to reset a password if required.

When a user first logs in to **Calpendo**, they will see the [Bookings Calendar](#)³⁹ unless configured differently.

Single Sign On Calpendo Account

Some customer environments use a single sign on. To access **Calpendo** using a single sign system a user would need to get authenticated by the single sign on system. To log in click on the button for your sign in system or the **Register new user** button to register using a specific authentication method. The names on the button(s) will have been chosen by the administrator when single sign on authentication is set up so may be different. If an unregistered user attempts to login using an external authentication method they are taken to the new user registration page for their authentication method.



Username:

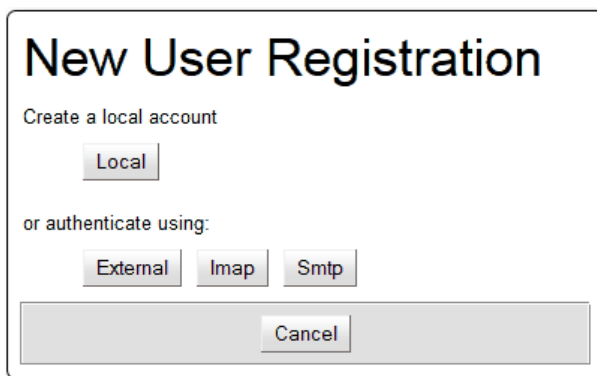
Password:

Sign in using one of the following:

Alternatively, sign in using:

If you do not already have an account:

If registering a new user choose the system to be registered by. If using a non-local method the user will need to be authenticated by that system before filling in the new registration form.



New User Registration

Create a local account

or authenticate using:

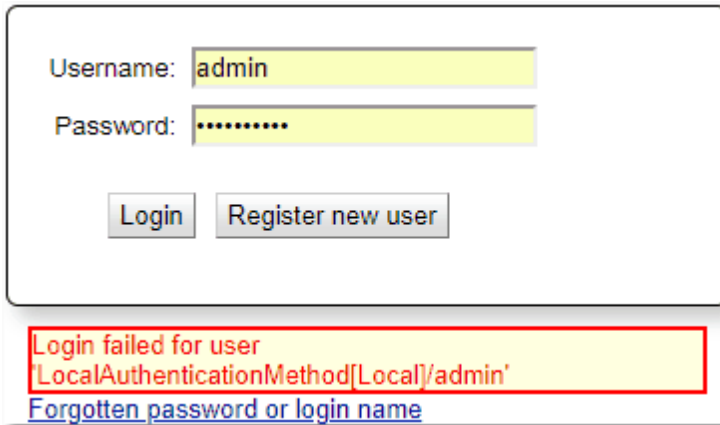
And then complete the **New User Registration** form, to create an account within **Calpendo**.

Once this account has been accepted by the administrator of **Calpendo** then the user will be able to login. This enables administrators to control whom has access to **Calpendo**.

3.2 Reset or Change Password

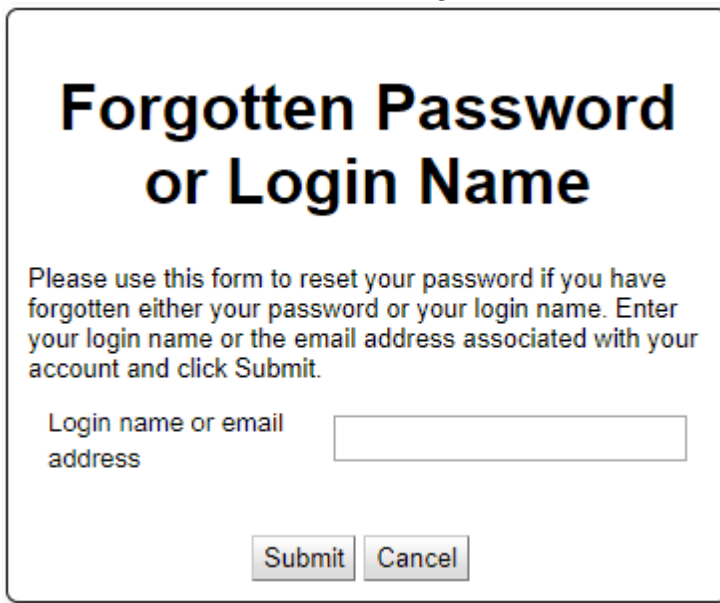
Reset Password

If a user has forgotten their password and failed to login, they will have the option of resetting their password by clicking the option below the login screen.



The screenshot shows a login form with two input fields: 'Username:' containing 'admin' and 'Password:' containing a masked password. Below the fields are two buttons: 'Login' and 'Register new user'. A red-bordered box highlights an error message: 'Login failed for user [LocalAuthenticationMethod[Local]/admin]'. Below the error message is a blue underlined link: 'Forgotten password or login name'.

Then enter their email address or login name.



The screenshot shows a form titled 'Forgotten Password or Login Name'. Below the title is a paragraph: 'Please use this form to reset your password if you have forgotten either your password or your login name. Enter your login name or the email address associated with your account and click Submit.' Below this text is a label 'Login name or email address' next to an empty input field. At the bottom are two buttons: 'Submit' and 'Cancel'.

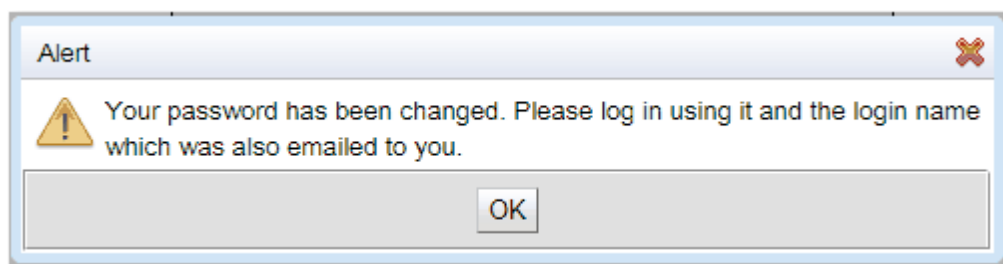
They will receive an email with a reset code, using this they can reset their password.

Forgotten Password or Login Name

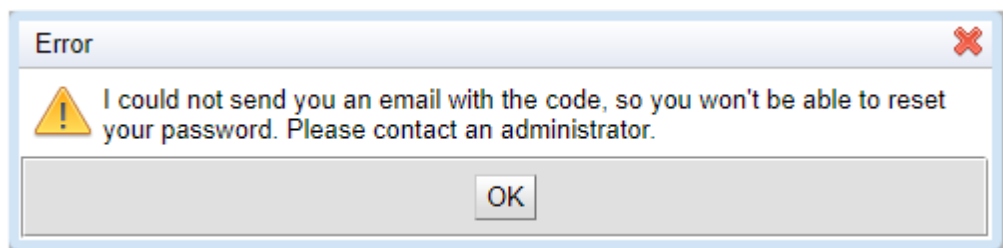
Please check your email for the reset code, and enter it below along with a new password.

Code	<input type="text" value="H4QM3CF7M8E8GWEX"/>
Password	<input type="password" value="....."/>
Repeat Password	<input type="password" value="....."/>

A message will let the user know the reset succeeded. There will also be an email sent to the user to confirm this.

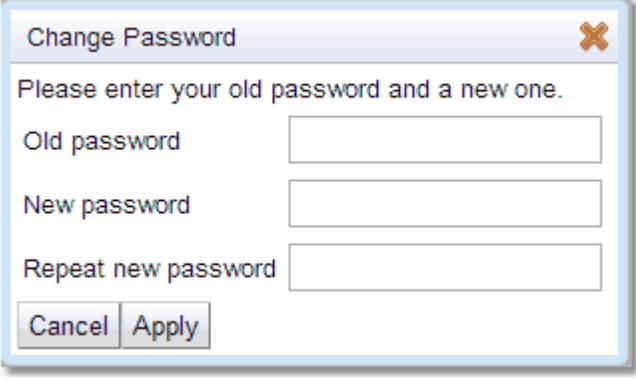


If it fails there will be an appropriate error message such as:



Change Password

If logged in using a local authentication method, in the top-right corner of every page of **Calpendo**, there is a **Change Password** link. Click on this if the user would like to change their password. A pop-up will appear that requests the old password and two copies of the new password. Press the **Apply** button to change the password.



A screenshot of a 'Change Password' dialog box. The dialog has a title bar with the text 'Change Password' and a close button (X) in the top right corner. Below the title bar, the text 'Please enter your old password and a new one.' is displayed. There are three input fields: 'Old password', 'New password', and 'Repeat new password'. At the bottom of the dialog, there are two buttons: 'Cancel' and 'Apply'.

Change Password

Please enter your old password and a new one.

Old password

New password

Repeat new password

Cancel Apply

3.3 User Settings

The [User Settings](#)⁶⁵⁶ page configures various options that control how **Calpendo** looks. Access it by going to the **Settings** link that is shown in the top-right corner of every page in **Calpendo**.

This will show a *users settings* with several tabs, as described below. Once the required settings have been changed then press the **Save** button to save all the changes.

3.3.1 Calpendo User Settings








Booking Reminder

When a [booking](#)⁶⁵² is created, the user can choose whether a reminder email should be sent ahead of the time that has been booked. This is described in [Creating Bookings](#)⁵⁶. Each of the settings on the reminder email section are given default values, and this **Booking Reminders** tab of the [User Settings](#)⁶⁵⁶ page sets up what those defaults are.

If it is preferred that *booking* reminders are to be off by default when creating a new *booking*, then just untick **Enable Automatic Booking Reminders**. Set up the default for who should receive an email, and the notice period ahead of the *booking* when the email should be sent.

Calendar View

This tab customises some aspects of the way the [Bookings Calendar](#)⁶⁵² is displayed.

Appearance Booking Reminders Buttons Calendar View Date & Time Email Menu Refresh	Booking tooltip delay (milliseconds, zero to disable)	1000
	Number of day/week vertical pixels per 30 minutes	Use system default ▼ 30
	Time displayed in vertical timeline	Use custom setting ▼ 3 ▼ Days ▼
	Default Calendar View Type	Use system default ▼
	Default Project	Please select a Project ▼
	Time of Day markers	Use system default ▼
	Bookings Calendar Background Colour	
	Colour when bookings would be automatically denied	Use system default ▼ 
	Colour when booking warnings would be given	Use system default ▼ 
	Colour when booking requests can be made	Use system default ▼ 
	Colour when bookings would be automatically approved	Use system default ▼ 
	Colour when templates are indeterminate	Use system default ▼ 
	Colour when no templates apply	Use system default ▼ 
	Start and Finish	
	Day starting hour	Use system default ▼
Day finish hour	Use system default ▼	
First day of week	Use system default ▼	
Number of days per week	Use system default ▼	
Resource Columns		
Resource Column Display	Use system default ▼	
Show header for resource columns	Use system default ▼	
Resource Usage Calendar Background Colour		
Background colour in the future	Use system default ▼ 	

The top section deals with some general viewing options.

When the mouse hovers over a *booking*, a tool tip appears that gives information about the *bookings* content. These tool tips can be disabled or the period of time the mouse must be stationary before the tool tip shows can be set, by changing the **Booking tooltip delay**. Set this to zero to disable the *booking* tool tips, otherwise set this to a number of milliseconds.

Choose the amount of space allocated to each hour of the day in the calendar by changing the **Number of vertical pixels per 30 minutes**. The default is 30. If a larger number is selected, then the *calendar* will take up more vertical space. Or it can be shrunk by reducing the number. Note that this does **not** change the font size used to display on the calendar.

Choose approximately how many days worth of *bookings* will be viewed in the **Vertical Timeline**. This is also used to decide how much space to use for bookings in the **Horizontal Timeline**, where usually more days can be viewed. This defaults to the system default.

Choose whether to use the system default view for the *calender* when **Calpendo** first starts or whether it is to start in Day, Week or Month view.

Choose which project will be auto selected by default in booking popup.

Choose whether to use the system default for the **Time of Day Markers** or for the user to specify themselves whether the **Time of Day Markers** can be seen.

The **Bookings Calendar Background Colour** section allows specification of the colours used to display [Time Templates](#)⁶⁵⁶ in the background, if requested on the *Bookings Calendar*. Choose the colours depending on whether *bookings* would be [automatically denied](#)⁶⁵², [automatically approved](#)⁶⁵², given a warning, [requested](#)⁶⁵⁵, indeterminate or when no *Time Template* applies.

The Start and Finish section allow the user to define which days of the week to be displayed as well as the hours on their calendar. So for example, the *calender* will only show from 8am to 6pm and from Monday to Friday if the building was always closed outside those hours.

The **Resource Columns** section:

Lets you select how to allocate columns for each [resource](#)⁶⁵⁵.

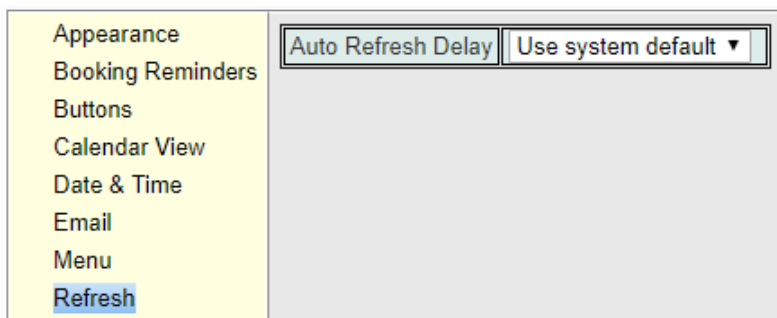
- Share When Possible** means bookings for different *resources* will be put into the same column as long as they do not occur at the same time.
- Separate** means *bookings* for different *resources* always appear in separate columns. If there are no *bookings* for a particular *resource*, then no column will be shown for that *resource*.
- Always Present** means *bookings* for different *resources* always appear in separate columns, and the column will display even if there happens to be no *bookings*

This also allows the definition of whether the *Bookings* and [Time Templates Calendars](#)⁶⁵⁶ will have a heading at the top of each column for each *resource* displayed on the *calendar*.

Finally set the colour to appear on the [Resource Usage Calendar](#)⁶⁵⁵ for the future when creating [resource usage](#)⁶⁵⁵ reports would be denied.

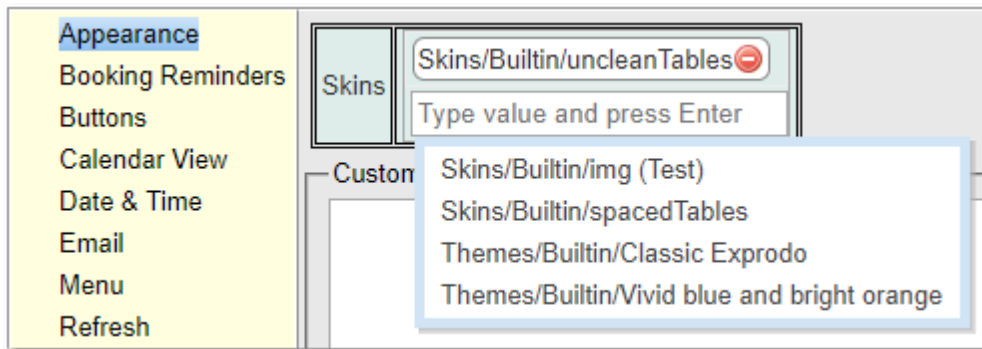
Refresh

This tab customises the refresh time of the calendar. It defaults to the **System Default** but can be changed for an individual user.

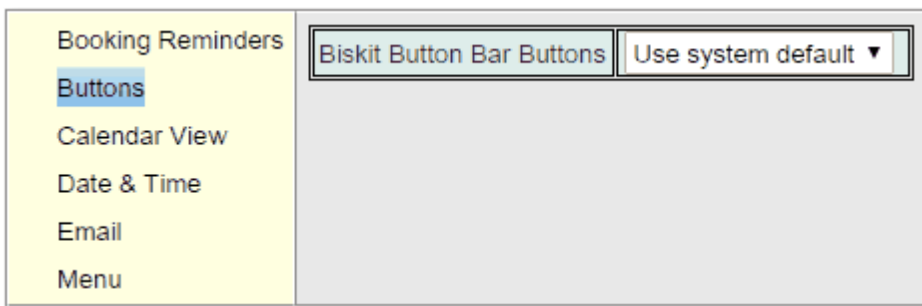


3.3.2 Appearance

This tab allows a user to set up their own CSS to build a theme. Or select from the current list of themes/skins available.



3.3.3 Buttons

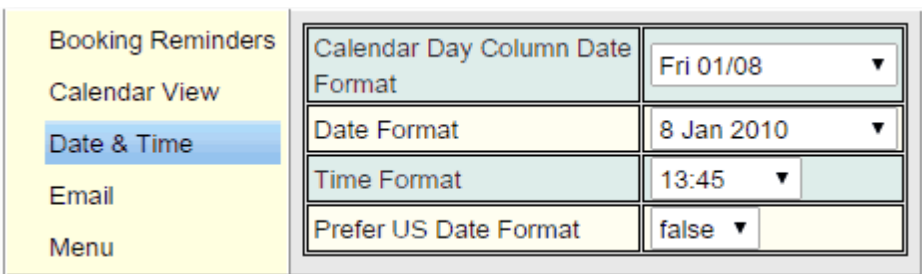


This tab controls what buttons appear when viewing a single [biskit](#)⁶⁵² or a list of *biskits*.

This initially only controls whether the relatively new **"Send Email"** button is displayed, but other buttons are likely to appear here in the future.

3.3.4 Date & Time

This tab allows selection of the format to use for displaying the day and date at the top of each day in the [Bookings Calendar](#)⁶⁵², and also the date and time format used everywhere else. Also specify whether US date format is preferred.



3.3.5 Email

This tab allows the user to choose whether to opt out of automatic emails and reminders. The user can also update their email address stored under their user details.

The screenshot shows the 'Email' settings tab. On the left is a sidebar with a yellow background containing the following menu items: 'Booking Reminders', 'Buttons', 'Calendar View', 'Date & Time', 'Email' (which is highlighted in blue), and 'Menu'. The main content area on the right has a light gray background and contains two rows of settings. The first row has a label 'Opt Out Of Automatic Emails' and a dropdown menu currently set to 'Allow automatic emails'. The second row has a label 'Email address' and a text input field containing 'ben@exprodo.com'.

3.3.6 Menu

This tab allows the user to choose how the menu operates. Either configure it so that each sub menu opens up just by placing the mouse over a menu, or that the user must click on a menu for its sub menu to open.

Also, **Calpendo** can show different menus to different user roles. If the user has sufficient [permission](#)⁶⁵⁴ to change their menu, then a drop-down that lets you select which menu you should see will be shown.

The screenshot shows the 'Menu' settings tab. On the left is a sidebar with a yellow background containing the following menu items: 'Date & Time', 'Menu' (which is highlighted in blue), and 'Menu'. The main content area on the right has a light gray background and contains a single setting: a label 'Automatically Open Menu' followed by a dropdown menu currently set to 'false'.

3.4 Bookings

A [booking](#)⁶⁵² in **Calpendo** is for a period of time and always for one [resource](#)⁶⁵⁵. A *resource* is a room, piece of equipment or person that is bookable. If two *resources* need to be booked simultaneously, then that requires two **Calpendo bookings**. *Bookings* may represent an [exclusive use of a resource](#)²⁴⁹, or there might be non-exclusive use of some *resources*, depending on how **Calpendo** has been configured.

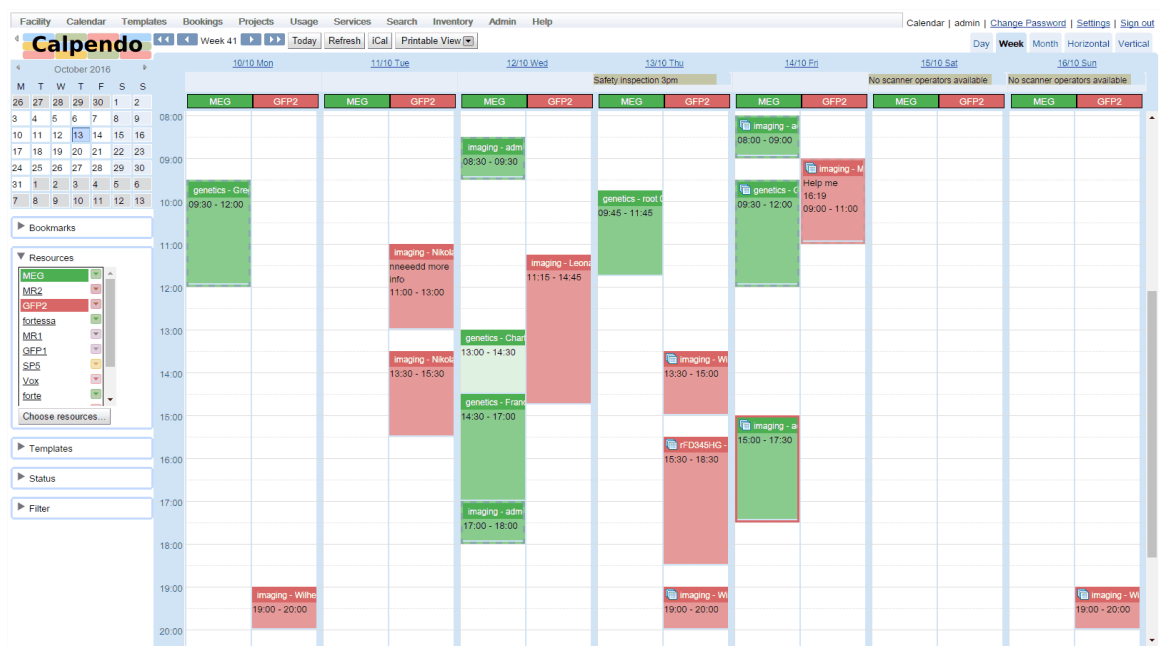
Some *resources* may require that there is a [project](#)⁷⁴ associated with each *booking*.

Bookings may be repeating (for example, every day or every week) and it's possible that a user may be able to create [repeat](#)⁶⁵⁵ *bookings* only on some *resources* or not at all, depending on the [Booking Rules](#)²³⁵ and [Permissions](#)³¹⁴ that have been set up.

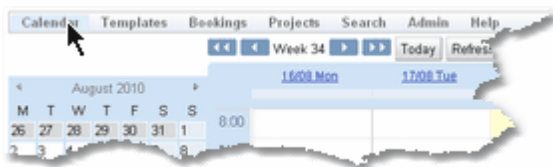
Bookings may have to use predefined slots where both the time and the length of the *booking* has been defined by the administrator.

3.4.1 Bookings Calendar

When a user first goes to the , the user will see the [Bookings Calendar](#)⁶⁵² showing a week of [bookings](#)⁶⁵².



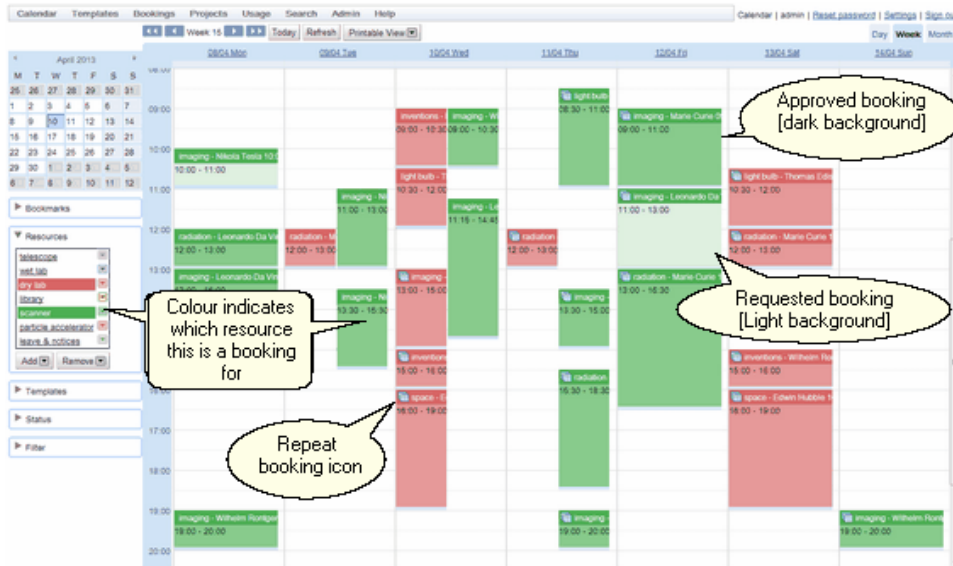
By default, the *Bookings Calendar* appears on the **Calpendo** menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

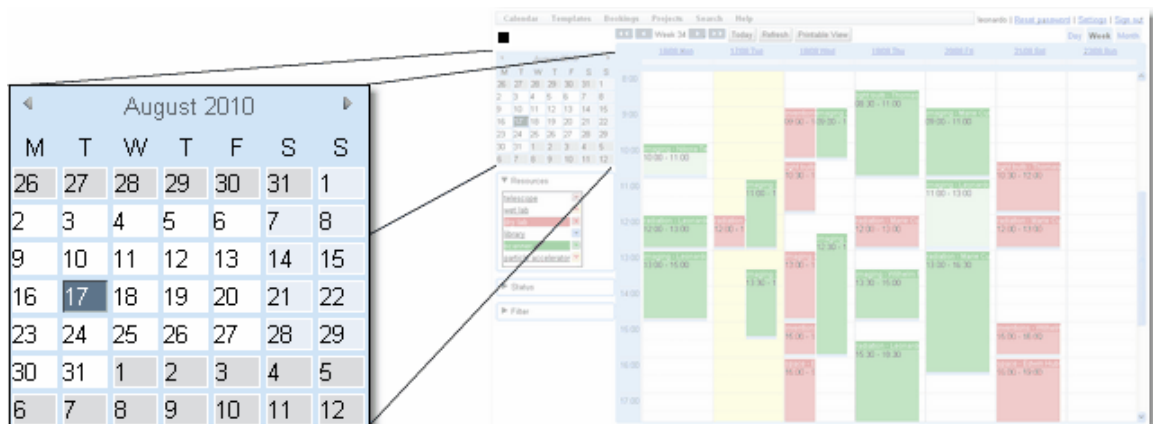
3.4.1.1 Exploring The Bookings Calendar

The colours on the [Bookings Calendar](#)⁶⁵² are used to indicate the [resource](#)⁶⁵⁵ being booked, and the shade tells you whether a [booking](#)⁶⁵² is a [request](#)⁶⁵⁵ or if it has been [approved](#)⁶⁵². A dark *booking* is *approved*, while a light *booking* is a *request* (although if **tentative**, **cancelled** or **denied** bookings are being displayed, they are displayed with stripes, Tentative: green stripes, Canceled: dark stripes, Denied: red stripes). This screen shot shows a week's *bookings* for two *resources*, the *bookings* for one *resource* in green and the other in red.

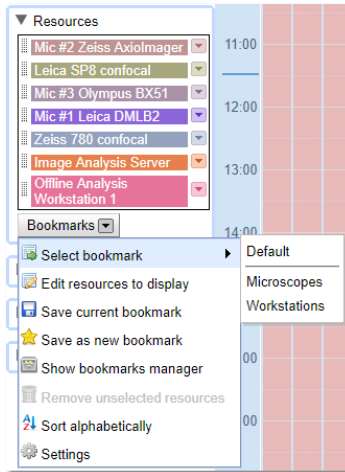


The navigator and filters/selectors can be removed/returned by pressing the arrow button found in the top left hand corner of the screen. This can be found on all calendars.

The month navigator quickly changes the date being displayed. Click on any date to display *bookings* for that date. This will work the same regardless of whether the view is for a day, week or month of *bookings*. Today's date is shown with a dark background.



The **Resource Selector** shows the available *resources*, which ones are currently selected, and what their colours are. Click on a resources name to make it the only visible resource. Hold the shift or control key down and click on a resource, to also select that resource for viewing. If the background colour is visible, then that *resource* will be on the *calendar*.



Use the **Bookmarks** dropdown to choose which bookmark is currently being viewed. *Bookmarks* can be system wide or on a per user basis. Click on any *bookmark* to see the *resources* that *bookmark* shows. For more information on how to set up and use *bookmarks* see the [Bookmark Manager](#)¹⁸² chapter.

If all the *resources* are not required in view click the **Edit Resources to Display** icon and use the **Resource Selector** to pick the *resources* to be shown on the list. For a description of how the **Resource Selector** works read the [Resource Selection](#)⁴⁸ section at the end of this chapter.

The **Save Current Bookmark** icon will save the current selected *resources* as the current bookmark.

The **Save As New Bookmark** icon will save the current selected *resources* as a new bookmark.

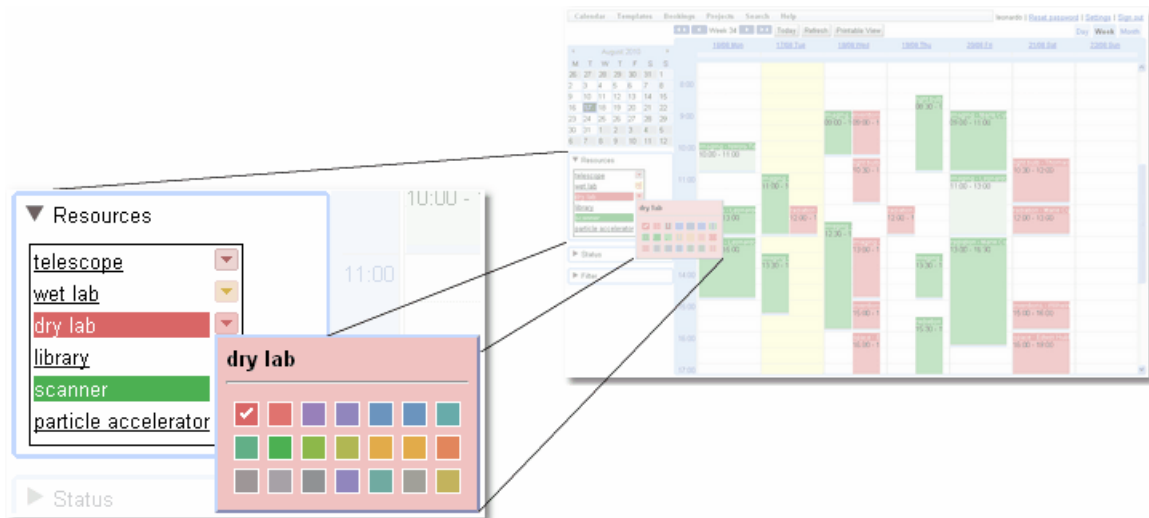
The **Show Bookmarks Manager** will take the user to the **Bookmarks Editor**.

The **Sort Alphabetically** icon will sort the resources alphabetically in the bookmark and on the calendar.

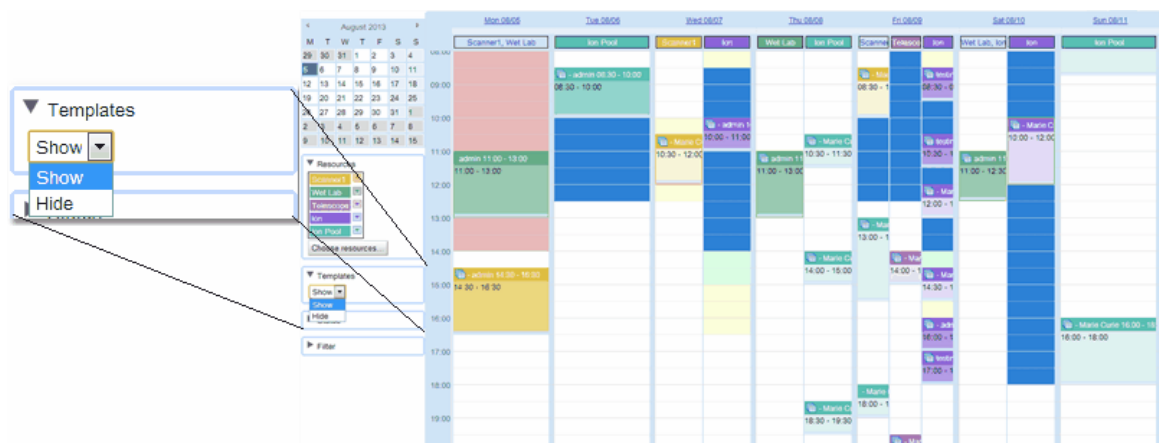
The **Settings** icon allows the following changes

Setting	Effect
Allow Reordering by dragging Resources	Adds the drag and drop icon to the left of the <i>resource</i> name so they can be reordered by dragging and dropping.
Toggle Resource on click.	This changes the behavior so that clicking a resource hides it rather than hides everything else.

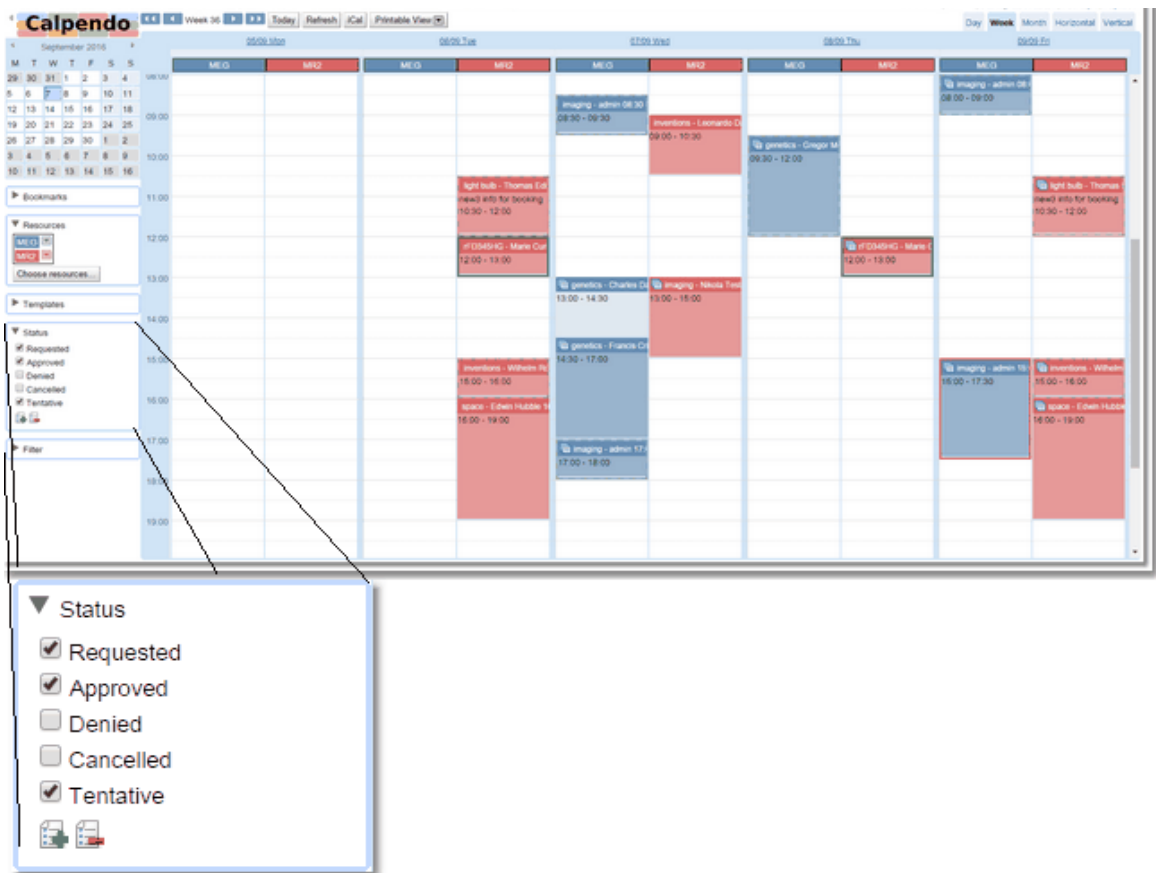
To change the colour used for any *resource*, click the button to the right of the *resource* name. This shows a pop-up where you can select from the available colours.



The *Time Templates* option allows you to decide whether to show or hide basic *Time Template* information for all *resources*. This allows the user to quickly see for a *resource* which *Time Templates* are being used. Areas in red are either in the past or *bookings* are not allowed, in yellow, *bookings* may be requested awaiting acceptance although a warning may be issued, in green when *bookings* will be automatically accepted or white where no *Time Template* is present. (The colours are the defaults and may be changed in [User Settings](#) ³⁴). In the example below the *Requested* colour has been changed to blue to make those areas stand out from the warning areas.). If the *calendar* shows *Time Templates* for multiple *resources* in the same column, then it will show the background colour to represent the most lenient restriction for all the *resources* it's showing *Time Templates* for (no *Time Template* being the least restrictive). Which means if the background is red, then none of the *resources* displayed in that column can not be booked. If it's white, yellow (or blue in the example below) or green, then the user will be able to book for at least one of the *resources* displayed in that column. Moving your cursor over an area where *Time Templates* are active, for the current user, will provide a tool tip with any message(s) the administrator has defined for the *Time Template(s)*.



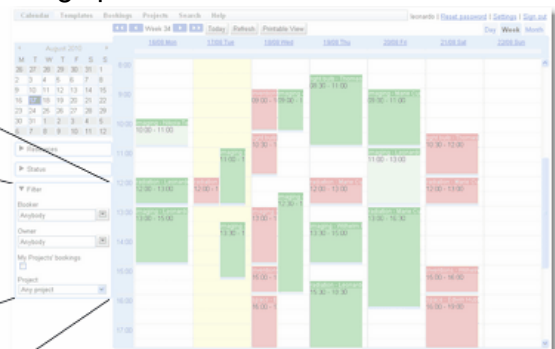
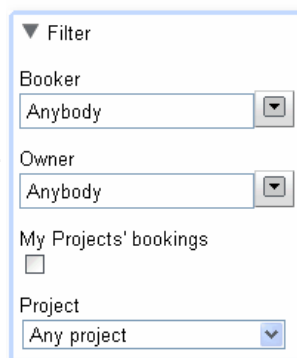
Normally, the *bookings calendar* displays *bookings* that are *approved*, *tentative* or *requested*, but ignores **denied** or **cancelled** bookings. Click on the **Status** label, to open up the **Status Selector**. Here choose the statuses to be displayed.



Click on the **Filter** label to provide more detailed filtering options.

Using a filter can restrict which *bookings* are displayed by selecting who booked or who owns each *booking*.

Click **My Project's Bookings** to show only *bookings* for *projects* that the user is associated with.



Select a particular *project* to show only *bookings* for that *project*.

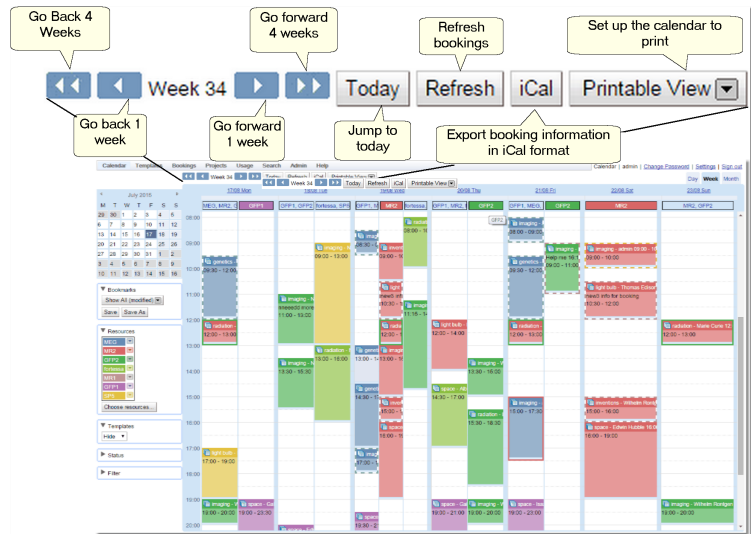
The button bar at the top of the *calendar* allows movement through time, refreshes the currently displayed *bookings*, exports *iCal* information or switches to a view suitable for sending to a printer.

The forwards and backwards buttons change how far forwards and backwards they go depending on which view is currently shown, day, week or month view.

In a day view, the buttons go forwards or backwards one day or one week.

In a week view, they go forwards and backwards one week or one month.

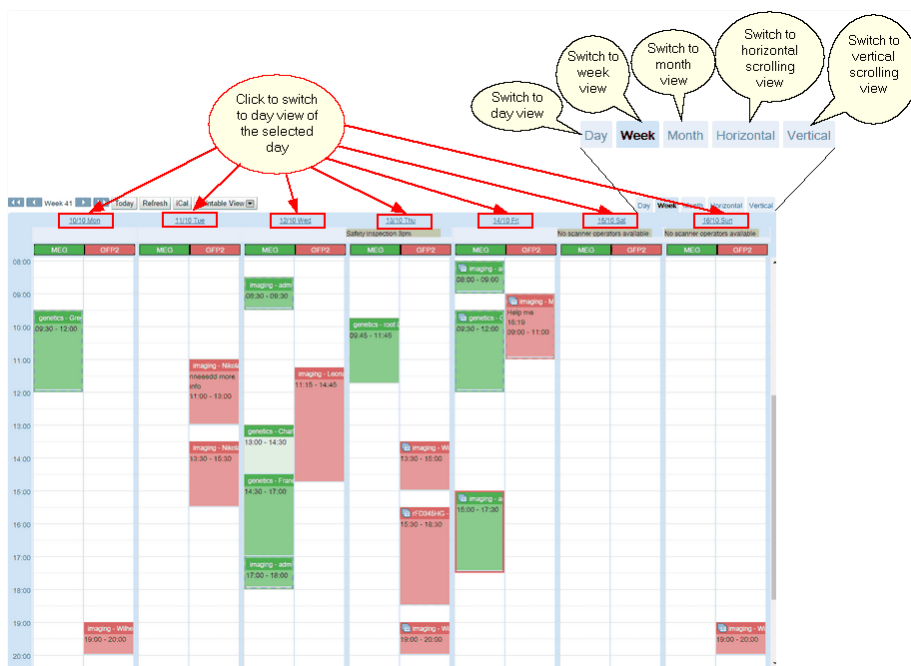
In a month view, they go forwards and backwards one month or one year.



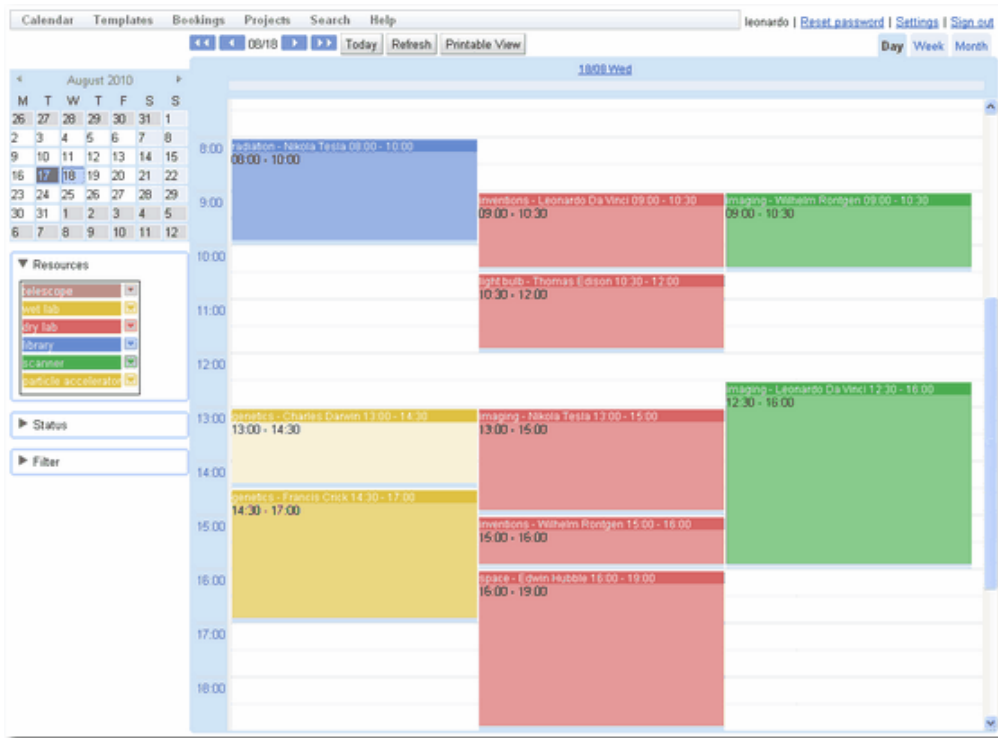
The **Today** button will always jump to today.

The **Refresh** button will refresh the display and find any changes that have been made since *bookings* were last fetched.

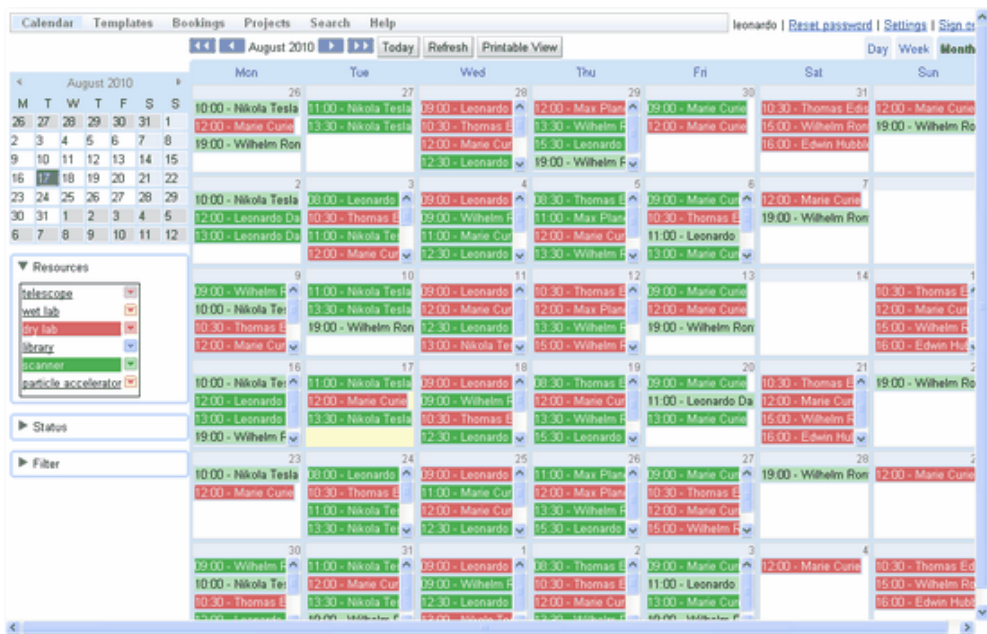
At the top of each day in the week view is a label showing the date. Click on that date to go to a day view of that date. There's also a tab selector on the far right to choose the day, week or month view.



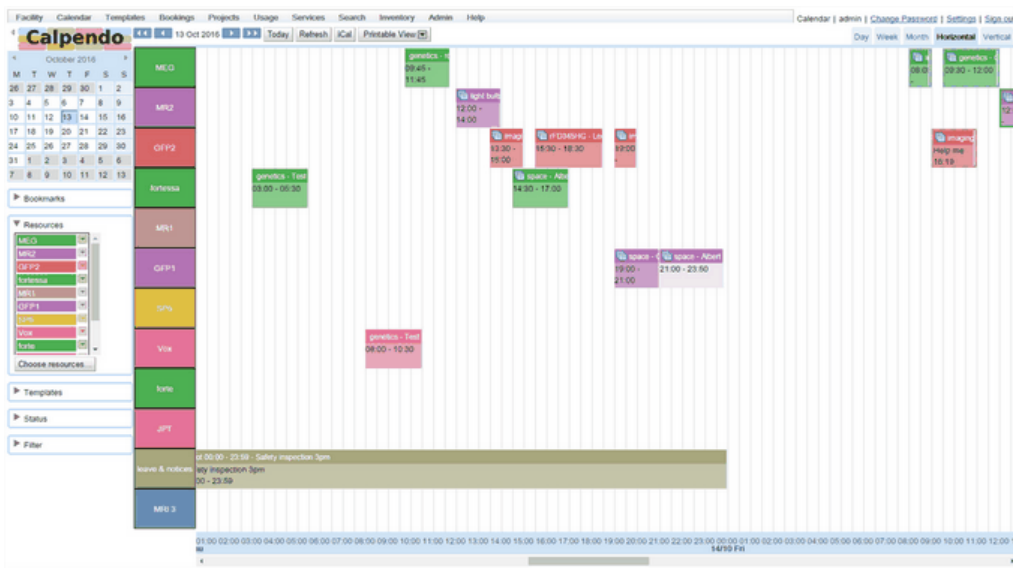
This is a day view. *Bookings* for different resources may display in the same column, as long as they don't conflict with each other. This is shown in this screen shot by the yellow and blue bookings.



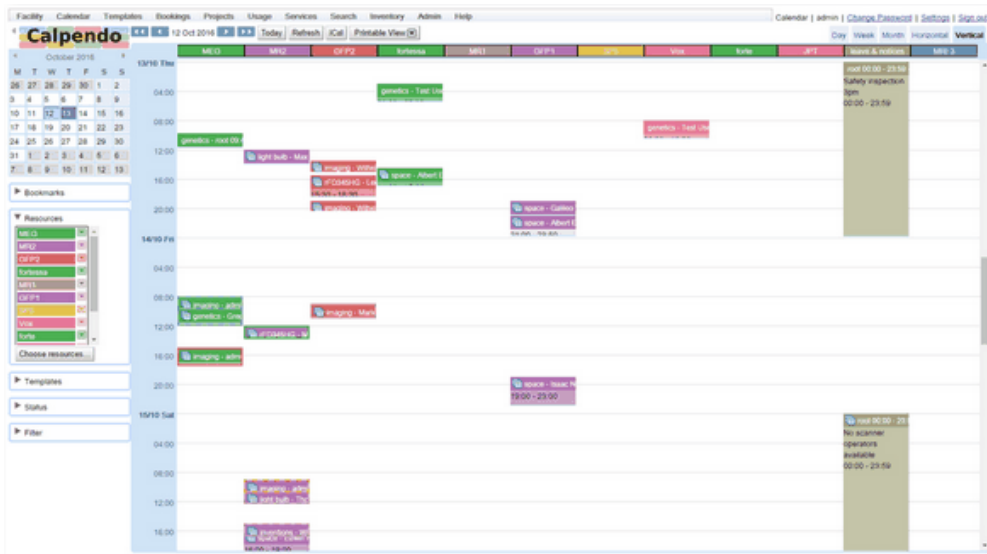
This is a month view. It can get very cluttered if *bookings* are not filtered.



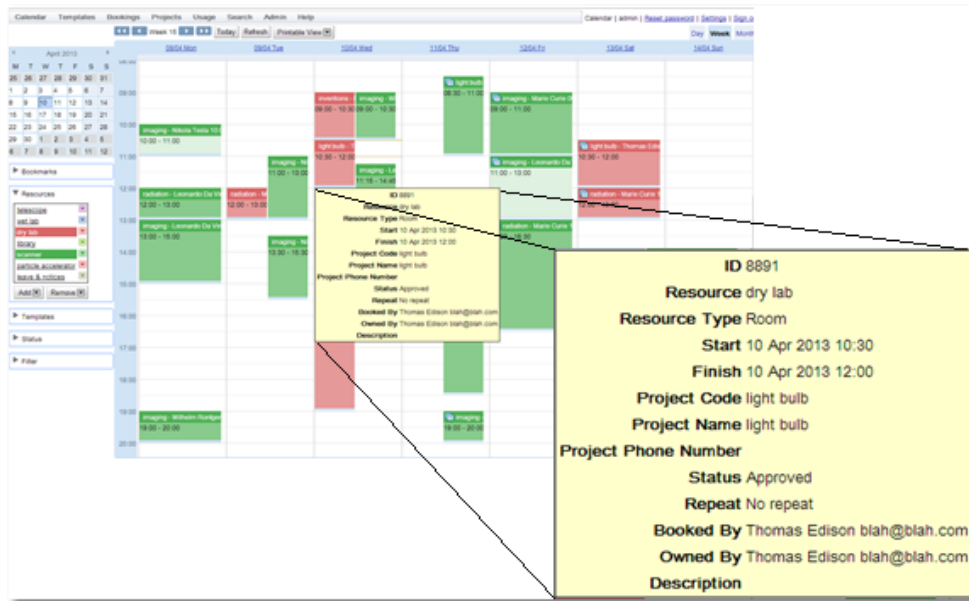
This is the **Horizontal Scrolling View**. This allows you to see a large number of *resources* at the same time and scroll through time. The number of days shown at any one time is defined in [User Settings](#) ³⁴.



This is the **Vertical Scrolling View**. This allows you to see a large number of *resources* at the same time and scroll through time. The number of days shown at any one time is defined in [User Settings](#) ³⁴.

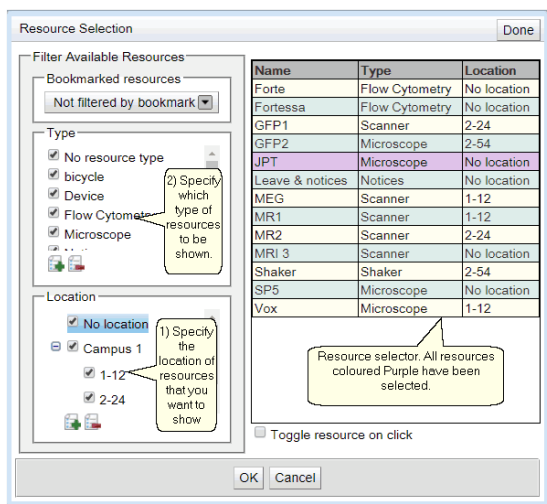


When the mouse hovers over a *booking*, a tool tip will appear which shows some of the details for the *booking*. More information can be found by clicking on the *booking* to get the *booking* pop-up, but the tool tip shows the most important information. This tool tip information may be configured by the administrator in the [Resource Editor](#)²⁸⁴.



Resource Selection

When dealing with a number of *resources* there will be times when a user will only need to see some of the available *resources*. The **Resource Selection** pop-up allows users to determine which resources will be shown.



First of all select the location the *resources* come from. Selecting a top level location will automatically select all the lower level locations associated with it, the reverse does not happen. Lower level locations can be selected without their upper level selected.

Then select the types of *resources* required to be viewed.

All these *resources* will appear in the **Available Resources** box. Select the resources in this can be done individually, or use the ctrl key to select multiple non adjacent *resources* or the shift key to select multiple adjacent *resources*. (i.e use the mouse to select the first resource, hold the ctrl key down while using the mouse to select other *resources*).

Toggle resource on click. By default when you click on an individual resource all other resources will be unselected, if this choice is ticked then the opposite is the case, each resource clicked individually will be added to the current selection list.

When happy with the list of **Available Resources** click the **OK** or **Done** button.

Note: If a *resource* is selected in the **Available Resources** area and it is removed from that box because either its type box, or its location box are unticked, then the *resource* will be unselected.

Resource Picture

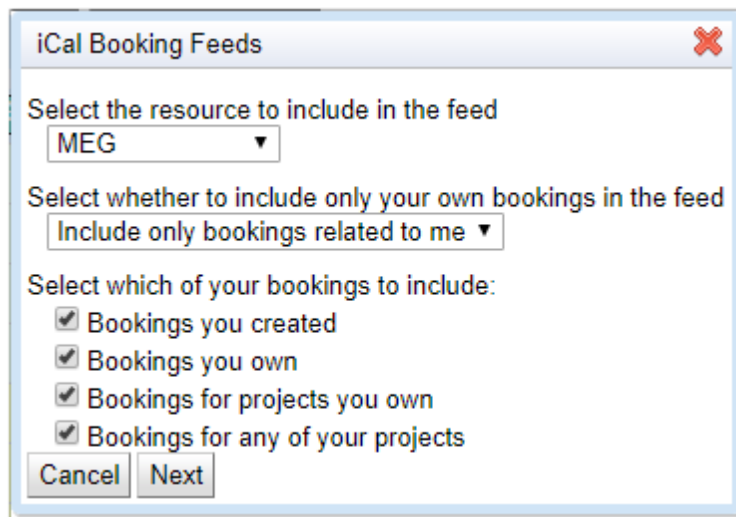
If the administrator has set it up a user can see a picture of the resource by moving the cursor over the column header or the resource in the list of resources in the filter section.



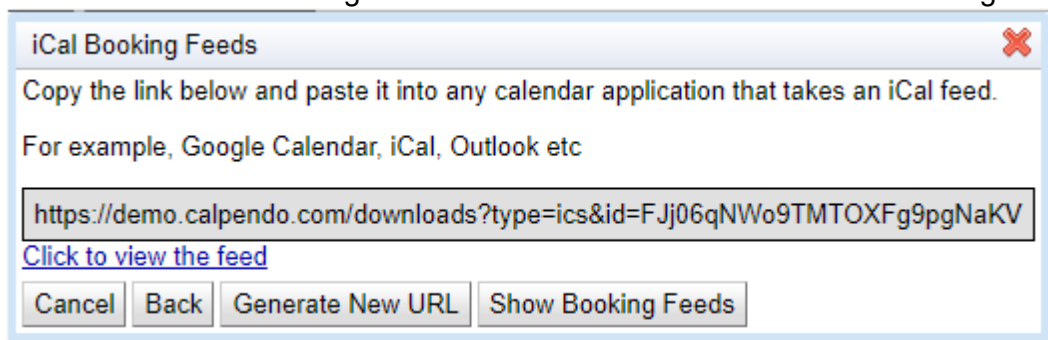
Exporting iCal Information

When exporting **iCal** information the user needs to decide if they want to export information on a single resource or all *resources*, and whether they want *bookings* related to themselves or other peoples *bookings* as well. If they want *bookings* related to themselves they can also decide which of the relationships they want to choose. Once these options have been chosen a link will appear at the bottom of the *booking* feed box, copy and paste this link to the **calendar** application. The frequency of updates to the **calendar** may be determined by the **calendar** application itself, if not, it is every hour.

If there are any **Permissions** which stop users from seeing parts of a **Booking** these may effect the **iCal** feed. For instance a **Permission** which stops **ical_User** from seeing the name of the **Booker** on *bookings* they have not booked will mean that all *bookings* will not be seen by the feed as when the request is made it is impossible to know who is making the request and therefore it fails.



Once the resource and booking combinations are selected then choose next to get the URL

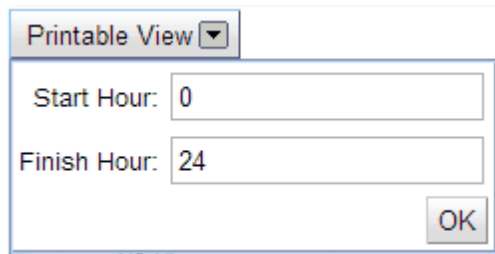


Generate URL will create a new **URL** to replace the existing one. **Show Booking Feeds** will take the user to a page which will display all the current feeds from this **Calpendo**.

The link to be copied is made up of two parts, the first part is extracted from **Admin->General Preferences->Email->Emailed Base URL** and then the "downloads?..." section is appended. Please make sure that the base URL is set-up correctly.

Printable View

In the *bookings calendar* you can bring up a printable view. Use the pull down to determine the hours you wish to view, and then click on the **OK** button.



Printable View ▼

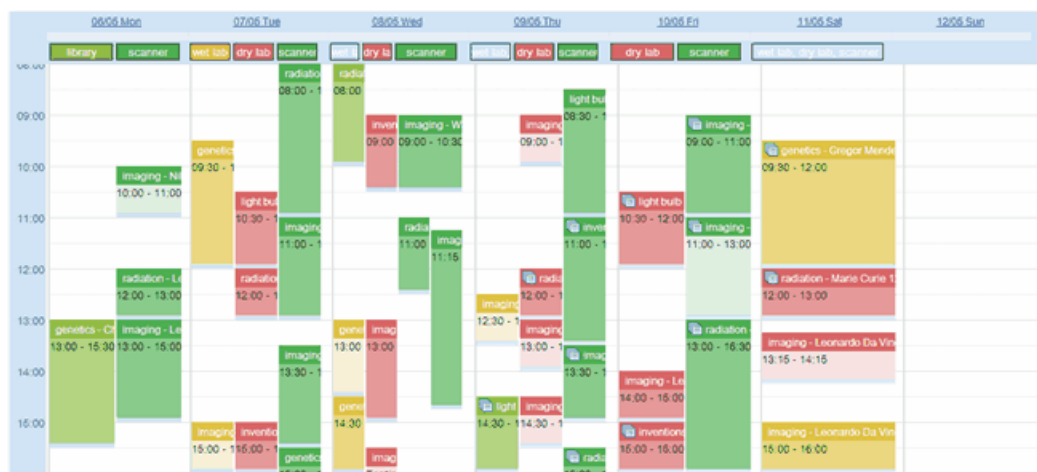
Start Hour: 0

Finish Hour: 24

OK

The **Printable View** you get will be the same type of view as you were originally in, day, week or month, but with only the hours requested being shown. To exit the view either press the browser back button, or left click the mouse anywhere in the window.

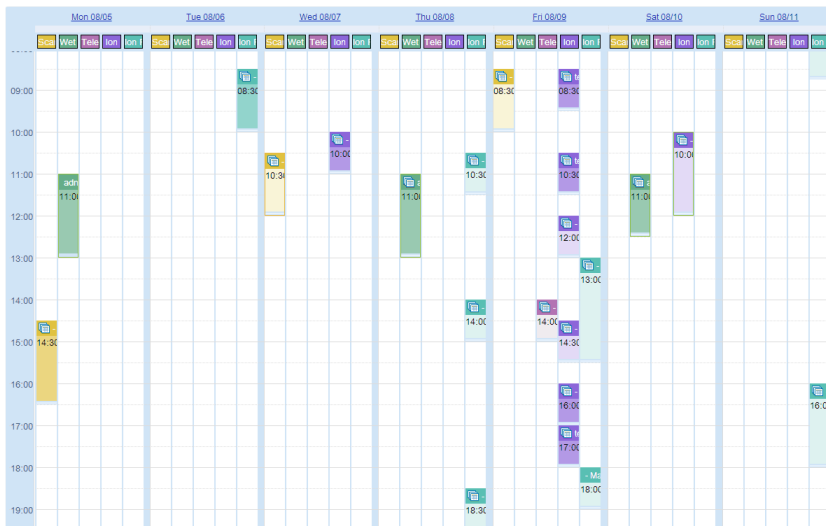
Here is an example of a weekly view showing from 08:00-16:00.



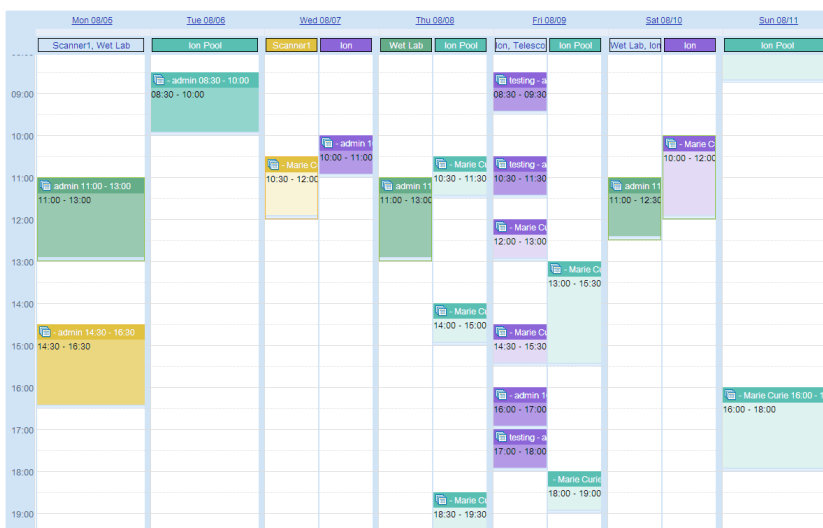
3.4.1.2 Display Of The Bookings Calendar

There are three methodologies for displaying [resources](#)⁶⁵⁵ in the [calendar](#)⁶⁵² (changeable in a users [Settings](#)³⁴ or globally in [Global Preferences->Bookings](#)⁵¹⁵).

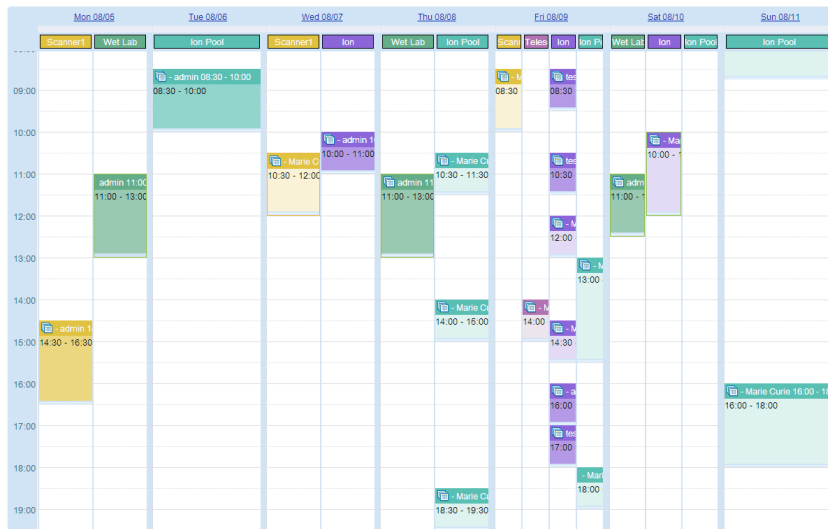
1) **Always present:** with this option all *resources* will be shown in separate columns at all times. The optimal methodology for using with **Calpendo** is **Always Present**. This is because both predefined slots, and viewing [Time Templates](#)⁶⁵⁶ work best. Predefined slots work best when a *resource* has its own column as the booking creation pop up can be filled in with the details of the predefined slot available at the time clicked on the calendar. *Time Templates* are viewed for all *resources* easily and accurately.



2) **Share when possible:** with this option *resources* will be displayed in the same column unless they overlap. The examples below also have their column headings switched on (also changeable in a users [Settings](#)³⁴ or globally in [Global Preferences->Bookings](#)⁵¹⁵). Columns displaying multiple *resources* will have their headings in a neutral light blue rather than the resources colour. With the display of *Time Templates* read the section dealing with Time Templates to understand how they are displayed in shared *resource* columns. For predefined slots if there is one column or shared columns then the pop up cannot have the slots time information automatically filled in, as there is no way of knowing whether the *resource* with the predefined slots is to be used.



3) **Separate**: with this option all *resources* are shown in separate columns, columns are only present if the *resource* has a booking on that day. This shows *Time Templates* well, but as with **Shared when possible**, those *resources* that do not have columns will not show *Time Templates* and predefined slot information will not be filled in the create booking pop up.

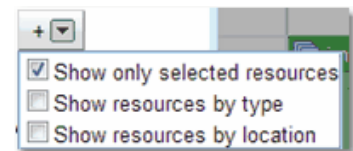


3.4.1.3 Creating Bookings

To create a new [booking](#)⁶⁵², in the [Bookings Calendar](#)⁶⁵², navigate to the date that the *booking* needs to appear on. Then either single-click in the *calendar* at the time when the *booking* is to start, or click-and-drag to select both the start and end times of the *booking*. Both methods will cause the *booking* pop-up to be displayed:

This pop-up is where the details of *bookings* are viewed and can be edited. Depending on the configuration of **Calpendo**, some of the fields displayed in this pop-up may not be editable, such as the [Status](#)⁶⁵³ and [Owner](#)⁶⁵⁴. If the user is not allowed to make [repeat](#)⁶⁵⁵ *bookings*, then there won't be an entry for the repeating information shown in the pop-up. Also additional fields may be present if added by the administrator.

Every *booking* must have a [resource](#)⁶⁵⁵, so select which *resource* to make a *booking* for. If the *bookings calendar* had only one *resource* displayed, or the *resources* are displayed in separate columns, then the *resource* will have been automatically selected, but it can still be changed to allow *booking* for some other *resource*. The default is to display the list of *resources* currently selected for display on the calendar. If this needs to be changed, click on the pop-up next to the *resource* box and choose what and how the *resources* will be displayed.



Deselect the **Show only Selected Resources** to show all *resources*.

Select **Show resources by type** to show the *resources* in a drop down by type. This option will only be available if the administrator has [Configured Types and Groups](#)²⁸⁰ for the *resources*.

Select **Show resources by location** to show the *resources* in a drop down by location.

Then click back on the **Resource** box to get the new list of *resources* to be chosen from.

Note that a *booking* is made for a *resource* that's not currently being shown in the calendar, then the calendar will automatically display all *bookings* for that new *resource*.

The administrator may have set up [Booking Types](#)⁶⁵³ in which case choose the type of *booking*, different *booking types* may be displayed in different ways on the *calendar*.

Some *resources* require a [project](#)⁶⁵⁴ for all *bookings*, and other *resources* do not. If a *resource* is selected that doesn't require a *project*, then the *project* selection will be hidden. The list of *projects* shown, will only be the *projects* the user is associated with. If the user is not associated with any *projects*, then a *booking* cannot be made for a *resource* that requires *projects*. The administrator can specify that a *project* can only book a *resource* if that *project* has an entry in its *resource settings* for the relevant *resource*. this is specified in the **Resource Editor**.

The time and date of the original selection are shown, but these can be changed before saving the *booking*. A *booking* that is set up to be **All day** will display at the top of the day column making it ideal for holiday notifications etc. If predefined *booking slots* are being used then the time in the pop up will match the slot available where the *calendar* was clicked. The times can be changed, but may not be allowed to be saved due to the **Predefined Booking Rule**. *Bookings* can go over multiple days.

Now set whether to get an [email reminder](#)⁵⁶ and whether the *booking* is [repeatable](#)⁵⁷. See below for more information on how to set these up.

The *status* will default to [Best Possible](#). This means that the best possible outcome will be attempted from [Approved](#)⁶⁵², [Requested](#)⁶⁵⁵ or **Denied**. If there is no *Time Template* available and the user is not denied by a *Booking Rule* or *Permission*, the [Default Booking Status](#)⁵¹⁵ set up in [Global Preferences](#)⁵⁰⁹ by the administrator will be used. This choice can be overridden by using the drop down menu, although this choice may be overridden in turn by *Time Templates*, [Booking Rules](#)⁶⁵² or [Permissions](#)⁶⁵⁴. If there is a *Time Template* available and it issues a **Warning** or the booking is deemed [Acceptable](#)⁶⁵² then *Best Possible* becomes *Requested*. For more information on how *Time Templates* restrict bookings read the section on [Time Templates](#)⁶² in the [Booking Restrictions](#)⁶² chapter.

The table below will show for each type of possible *Booking Status* and *Time Template* combination for a standard user, what final *Booking Status* will be applied to the *booking*. These results may be altered by any *Booking Rules* and *Permissions* that affect *bookings*.

Initial Chosen Booking Status	Time Template Available	Final Booking Status
Best Possible	No Time Template	Default Booking Status ⁵¹⁵
	Warning Template	Requested
	Acceptable	Requested
	Auto-Approval	Approved
Requested	No Time Template	Requested
	Warning Template	Requested
	Acceptable	Requested
	Auto-Approval	Requested
Tentative	No Time Template	Tentative
	Warning Template	Requested
	Acceptable	Requested
	Auto-Approval	Approved
Approved	No Time Template	Permission denied
	Warning Template	Permission denied
	Acceptable	Permission denied
	Auto-Approval	Approved

The *description* field is a place for you to enter any text that describes your *booking* or is required as a comment.

Notice that the **Create Booking** button in the previous diagram is greyed out. This is because every *booking* must have a *resource*, and there is no *resource* selected yet. Similarly, if a *resource* is selected that requires a *project*, but a *project* is not selected, then the **Create Booking** button will be greyed out. **Calpendo** may also detect that the selected values are prohibited by the *Permissions* that have been configured. If this happens, then again, the **Create Booking** button will be greyed out.

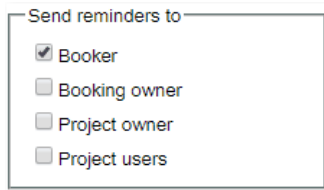
Once all the details have been entered and **Create Booking** has been pressed, then the new *booking* will be sent to the **Calpendo** server. Once there, it will undergo more tests to make sure that the *booking* is valid. An error message will appear if it is not allowed, a warning message if it is allowed, or the *booking* will be quietly accepted. Use **Cancel** to cancel the booking process.

Bookings that are created may be made as requests, or they may be auto-approved. This depends upon the configuration of the **Calpendo** and is described further in [Booking Restrictions](#)⁶².

Bookings can be copied and pasted, for fast entry of similar *bookings*. Just click on a *booking* in the *calendar* and select **Copy** from the pop up menu. Then, when the *calendar* is clicked somewhere to create a new *booking*, if there is a *booking* in the clipboard, its content will be used to populate the *booking* pop-up. It can then be edited as required.

Booking Reminders

Calpendo can send an email to remind about *bookings* that have been made. Tick the **Send reminder email** box in the *booking* pop-up to see the reminder options. The default reminder options are taken from the personal [booking reminder](#)³⁴ settings.



Send reminders to

- ☒ Booker
- ☐ Booking owner
- ☐ Project owner
- ☐ Project users

The notice period indicates how far in advance of the booking the email is to be sent.

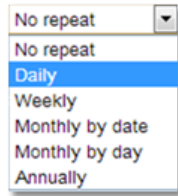
Then choose who gets a reminder email, with the options being:

- The person that made the *booking*.
- The person that owns the *booking*.
- The person that owns the *booking's project*. If there is no *project* on the *booking*, then this setting is not shown.
- The people associated with the *booking's project*. If there is no *project* on the *booking*, then this setting is not shown.

If a person falls into more than one category they will only receive one reminder email, for example the *project* owner is also the person that made the *booking*. Reminders are not sent to users who can no longer login unless their status is set to **Lurker**.

Repeat Bookings

If repeat *bookings* are allowed, then the *Repeat* option will be viewable. Select the type of *repeat* and then set the configuration options for that *repeat* type.



When creating a *booking* it can be set up to automatically *repeat*. There are a number of *repeat* choices:

Daily: Repeats at the same time each day.

Weekly: Repeats each week on this day at the same time. Specify which days the *repeat* will occur on by selected the appropriate tick boxes.

Monthly by date: Repeats each month on a particular date.

Monthly by day: Repeats each month on a particular day, this could be defined for example as: "Repeat on the 4th Monday of every month"

Annually: Repeats on the same date each year.

Once the type of *repeat* required has been selected, then choose how often the *repeat* occurs, every month, every other month etc.

Also define an end date for the repeated *bookings*.

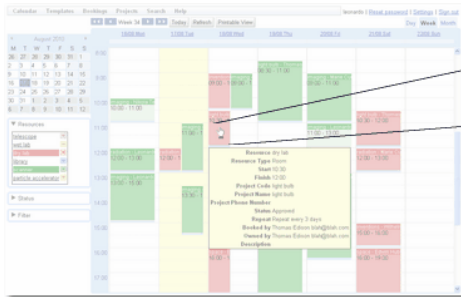
If the repeat is **Monthly by day** and a day towards the end of the month has been chosen, the **Last week** enables swapping between the two possible options.

e.g. "Repeat on the 4th Monday of each month" and "Repeat on the last Monday of each month".

The description box at the bottom gives a written definition of the *repeat* that has been defined allowing the choices to be checked to make sure they are correct.

3.4.1.4 Editing And Cancelling Bookings

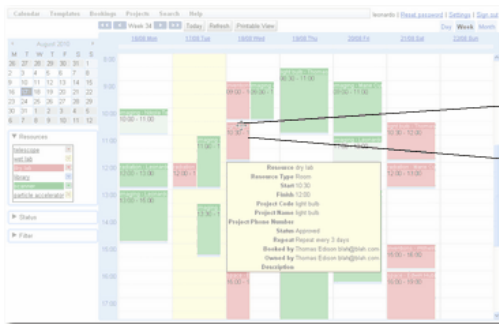
As you move the cursor over [bookings](#)⁶⁵² in the [calendar](#)⁶⁵², notice that the cursor change shape. The cursor will give a hint about whether or not a *booking* can be edited; if a *booking* cannot be edited then the cursor does not change as it moves over the *booking*.



This screen shot shows what the cursor will look like when the mouse moves over a *booking* that can be edited. Click with the mouse while over a *booking*, then the *booking* pop-up will appear.

If the *booking* is editable, then the fields in the pop-up will allow their values to be changed, just like creating a *booking*.

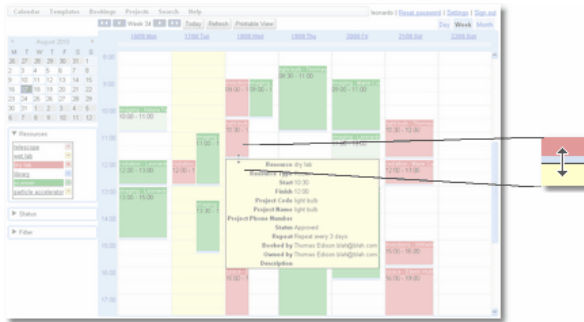
If there is no [permission](#)⁶⁵⁴ to edit a *booking*, then clicking will still get the *booking* pop-up, but nothing will be able to be changed.



If the cursor is placed over the title bar of a *booking* that can be modified, then the cursor changes to a new shape. This indicates not only that the user can edit the *booking* by clicking to display the *booking* pop-up, but that the user can also use drag-and-drop to move the *booking*.

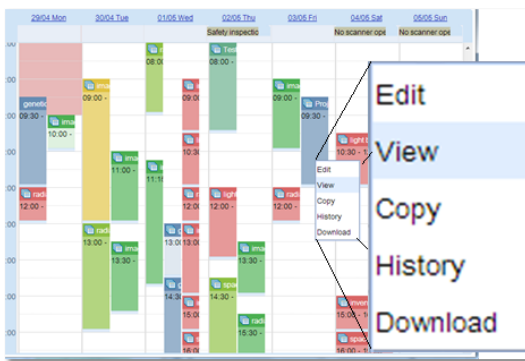
There may be limitations that have been set up to control where a *booking* can be dragged to, but if there is *permission*, then a *booking* can be dragged to a different start time, [resource](#)⁶⁵⁵ or different day.

Dragging weekly [repeatable](#)⁶⁵⁵ *bookings* to different days from their original *booking* day will cause the *booking* to move unpredictably. To change this type of *booking*, edit the *booking* and select different days of the week for the *booking* to be on.



Finally, by placing the cursor over the very bottom of a *booking*, the cursor changes to indicate that the *booking* duration can be changed by drag-and-drop. Click on the re-size bar at the bottom of a *booking* and drag it to the required time for the *booking* to end.

When a *booking* is dragged and dropped, or its duration changed by dragging its re-size bar at the bottom, then the change is made without showing the *booking* pop-up. However, by clicking on a *booking*, a pop-up menu is shown:



Edit : Will bring up the **Edit Booking** pop up as seen below, this is almost identical to the one seen when creating a new *booking*.

View: Will bring up a similar pop up but everything will be read only.

Copy: Will put the *booking* in the clipboard, when the user clicks somewhere to create a new *booking*, if there is a *booking* in the clipboard, its content will be used to populate the *booking* pop-up.

History: This will display the history of the *booking* showing when it was created and any updates to the *booking*. A *repeatable booking* will have one update for each repeat.

Download: Downloads the *booking* to your Outlook calendar.

The title of the pop-up now says **Edit Booking** instead of **New Booking**.

Some of the fields may not be changeable. For example, there's a **History** field that shows when the *booking* was created or last modified and a user will probably not have *permission* to change those values. Once changes have been completed use the **Update Booking** to implement the changes. Use the **History** from the pop up described above to get a full history of changes to this *booking*.

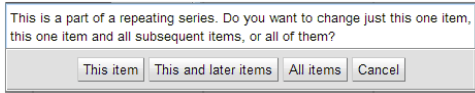
There is also a **Cancel Booking** button. *Bookings* are never deleted, they are only cancelled (unless they are *repeatable bookings*, see below). This means that the status on a *booking* is changed to **Cancelled**. **Cancelled bookings** are not normally displayed on the *Bookings Calendar*, so a **Cancelled booking** would disappear from the calendar. However, you may choose to display **Cancelled bookings**.

Calpendo may be configured so that cancelling a *booking* for a particular *resource* requires a reason. If a reason is required a text box will also appear for any additional explanation.

It is possible to uncanceled a *booking* by displaying **Cancelled bookings** and then changing its status to [Requested](#)⁶⁵⁵ or [Approved](#)⁶⁵², but suitable *Permissions* would be needed to be able to do that.

Use the **Cancel Changes** button to remove any edits and to exit the pop up.

If a *repeatable booking* is being edited, any occurrence of this *booking* in the past will not be affected as they are now separate unique instances, only occurrences of the *booking* in the future will be affected. This means a *repeatable booking* can be changed without affecting any information for reports of those *bookings* in the past. When dealing with *repeatable bookings* these are deleted rather than cancelled.

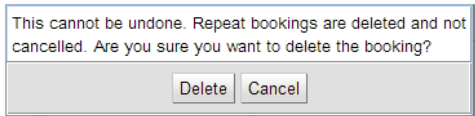


With *repeatable bookings* an option appears asking which items to be deleted.

This Item: Just the selected *booking*.

This and later items: The selected *booking* and all *bookings* after it in the repeatable sequence.

All Items: The selected *booking* and all *bookings* before and after it excluding any *bookings* in the past. ie. before the current time and date. If the first *booking* of a *repeat* has been chosen then **All Items** and **This and later items** are identical in function.



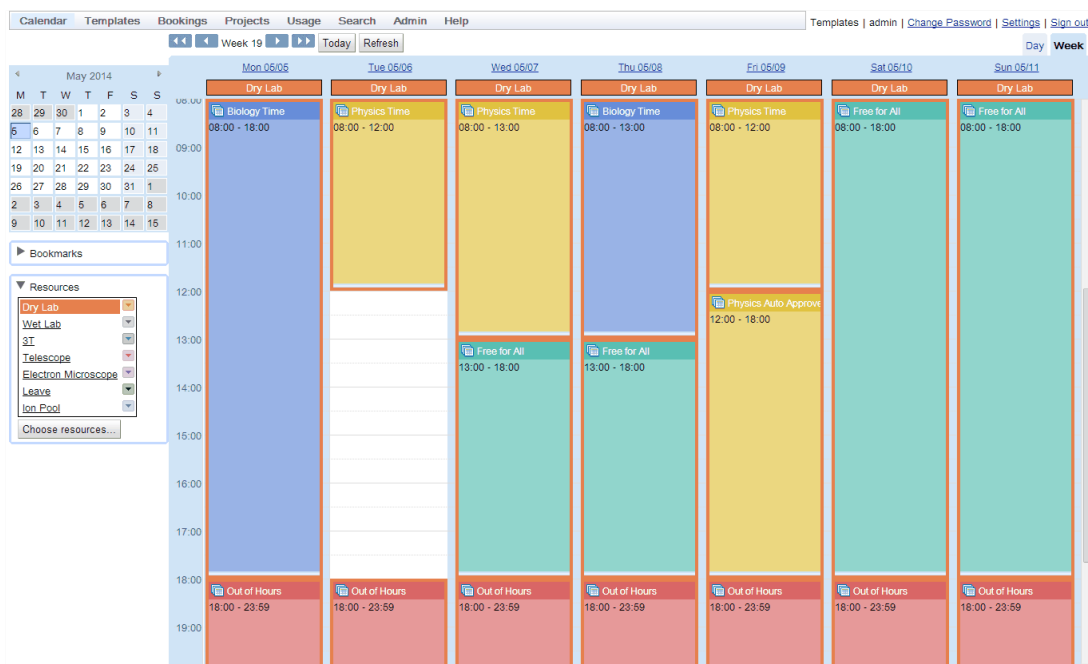
Then a reminder that *repeat bookings* are deleted and therefore cannot be recovered.

3.4.2 Booking Restrictions

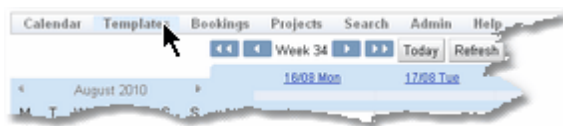
Calpendo allows users to make [bookings](#)⁶⁵² for themselves, while the system makes sure that *bookings* can only be created that adhere to the policies of the facility. This means that **Calpendo** will restrict what can be booked, when and for how long. These restrictions come in three types: [Time Templates](#)⁶⁵⁶, booking [Booking Rules](#)⁶⁵² and [Permissions](#)⁶⁵⁴. If a *booking* does not pass any one of these three restrictions it will be rejected.

3.4.2.1 Time Templates

[Time Templates](#)⁶⁵⁶ are a means of specifying periods of time during which different groups of people may book each [resource](#)⁶⁵⁵. The [Time Templates Calendar](#)⁶⁵⁶ looks very much like the [Bookings Calendar](#)³⁹. The main difference is that colour no longer pertains to the *resource*. Instead, a *Time Template* specifies who can book, and that *Time Template* spans various periods of time. Each *Time Template* is given a colour, and so the colour is an indication of who can book. *Time Templates* cannot span multiple days.



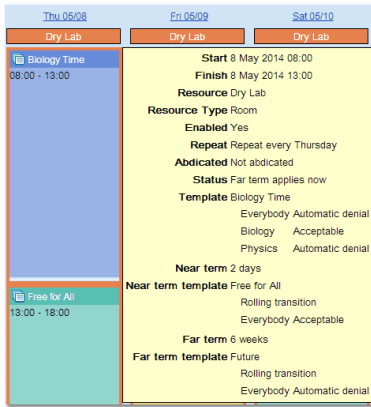
By default, the *Time Templates Calendar* appears on the **Calpendo** menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

A *Time Template* can choose one of four possible states for a [booking](#)⁶⁵² request:

1. [Automatic Approval](#)⁶⁵²
2. [Acceptable](#)⁶⁵², bookings will be given the status [Requested](#)⁶⁵⁵.
3. *Warning*, bookings will be given the status *Requested* but a warning message will be issued.
4. [Automatic Denial](#)⁶⁵²



Hovering over the blue *Time Template* for Thursday morning, which is named **Biology Time**, shows a tooltip with the details indicating it is acceptable for **Biology** people to make a *booking* for the dry lab. This means that *booking* requests can be made.

It can also be seen that for **Physics** people and everybody else, *bookings* are *automatically denied*.

All blue items shown in this *Time Templates calendar* are for the same *Time Template*, and so the same restrictions apply.

Abdicated⁶⁵²: This facility allows users to give up their saved time slots if they know no one will be using the *resource*. This means that the standard *Time Template* will now use the *Near Term Time Template*⁶⁵⁴ definition. Only users with the appropriate *permission*⁶⁵⁴ can abdicate a time slot and these will be set up by the administrator.

Status: This indicates which of the three possible *Time Templates* are currently being used for this specific time slot. They are:

Template: The standard *Time Template* with information on the restrictions enforced by the template.

Near term: A different *Time Template* used due to the proximity of the time slot to the current data and time, defined is when the *Near Term* starts.



Near term template: These define the *Time Templates* to be used, what time period defines near or far and the *booking* acceptability for different groups and how the transition is made.

Far term⁶⁵³: A different *Time Template* used due to the need to restrict *bookings* too far in the future.






Far term template: These define the *Time Templates* to be used, what time period defines near or far and the *booking* acceptability for different groups and how the transition is made.

For more on *Near* and *Far Term Templates* see the appropriate section later in the chapter.

Note that the *Time Templates Calendar* does not say exactly what is meant by **Biology** and **Physics**. The names of these *Time Templates* should be chosen so that they are meaningful enough that everyone understands what they mean. The details of precisely what they do mean can be found by looking at *Configuring Time Templates*²²⁶, but hopefully when a user looks at the *Time Templates* in **Calpendo**, they will understand what they mean from their name.

Fri 05/09	Sat 05/10
Dry Lab	Dry Lab
 Physics Time 08:00 - 12:00	 Free for All 08:00 - 18:00
<p> Start 9 May 2014 08:00 Finish 9 May 2014 12:00 Resource Dry Lab Resource Type Room Enabled Yes Repeat Repeat every Tuesday and Friday Abdicated Not abdicated Status Far term applies now Template Physics Time Physics Acceptable Everybody Automatic denial Biology Automatic denial Near term 2 days Near term template Free for All Rolling transition Everybody Acceptable Far term 6 weeks Far term template Future Instant transition Everybody Automatic denial </p>	

The Friday morning yellow *Time Template*, called **Physics Time**, indicates that **Biology** people are automatically denied *booking* the dry lab whereas **Physics** people are allowed to make *booking* requests.

Fri 05/09	Sat 05/10	Sun 05/11
Dry Lab	Dry Lab	Dry Lab
 Physics Time 08:00 - 12:00	 Free for All 08:00 - 18:00	 Free for All 08:00 - 18:00
 Physics Auto Approve 12:00 - 18:00	<p> Start 9 May 2014 12:00 Finish 9 May 2014 18:00 Resource Dry Lab Resource Type Room Enabled Yes Repeat Repeat every Friday Abdicated Not abdicated Status Far term applies now Template Physics Auto Approve Everybody Automatic denial Physics Automatic approval Biology Automatic denial Near term 2 days Near term template Free for All Rolling transition Everybody Acceptable Far term 6 weeks Far term template Future Instant transition Everybody Automatic denial </p>	
 Out of Hours 18:00 - 23:59		

The Friday afternoon *Time Template* is also yellow, the same colour as **Physics Time** and yet it is a different *Time Template* and called **Physics Auto Approve**. This is because whoever configures the *Time Templates* is free to assign whatever colours they want to each *Time Template*. Ideally, *Time Templates* for the same *resource* would be given different colours, but it's not an absolute requirement.

Physics Auto Approve specifies that **Physics** people will receive *automatic approval* for any *bookings* made for the dry lab during this time.

Near And Far Term Time Templates

Time Templates allow an administrator to specify who can use what when.

However, sometimes, it's more important to make sure that a *resource* is used by somebody rather than go unused because it couldn't be used by the "right" people. This is why *Time Templates* can be configured to mutate in the near term. For example, if by Thursday, the dry lab has no *bookings* for the following Friday morning, then perhaps the administrator would like to allow anybody from **Physics** or **Biology** to be allowed to book it. This is achieved by allowing the **Physics Time** *Time Template* to mutate to a more liberal *Time Template* within two days of the desired *booking* time. The tool tips above show not only what the *Time Templates* mean, but also what they mutate to in the near term.

It may also be important to make sure *resources* are not being booked too far in the future, in the hope they may be used. This is why *Time Templates* can be configured to mutate in the far term. For example, it may be policy that no one may book a *resource* more than six weeks in advance. This is achieved by allowing the **Physics Time** *Time Template* to mutate to a more restrictive *Time Template* if we are at least 6 weeks from the current time.

Transition allows the administrator to define how the change will take place from the Medium Term to the *Near* or *Far Term*. With **Instant** transition, at 9am, the whole of a *Time Template* from 9am to 5pm would become available for *booking*. With Rolling transition at 10am the portion of the *Time Template* from 9am to 10am would be available for booking.

3.4.2.2 Booking Rules

Whenever a [booking](#)⁶⁵² is created or modified, the **Calpendo** server checks the *booking* against its [Booking Rules](#)⁶⁵². There are many different sorts of *Booking Rules*, these allow **Calpendo** to apply pretty much any check the administrator might want to run.

Normally, the only time *Booking Rules* will be noticed is if a *booking* is created or modified that triggers a particular *Booking Rule*. *Booking Rules* may do one of the following things:

- Accept the change without showing you any message.
- Accept the change, but give a warning message.
- Refuse the change and show an error message.

The administrator should give indicate which *Booking Rules* are being used by the facility.

The configuration of *Booking Rules* is beyond the scope of the **Calpendo User Guide**, but read the [Configuring Booking Rules](#)²³⁵ section for more details.

3.4.2.3 Permissions

[Permissions](#)⁶⁵⁴ are not discussed much in the **Calpendo User Guide** because the *Permissions* that exist are completely configurable for the facility by the administrator and the user should not need to know much about them. A user will be subject to whatever *Permissions* have been set up for **Calpendo**, and the only way a user will know when something is not allowed is if the system prevents them from attempting to do something (such as greying out buttons that lead to actions that cannot be performed) or by giving an error pop-up that will say **Permission Denied**.

To know more about *Permissions*, then see the [Permissions](#)³¹⁴ chapter.

3.4.3 Searching For Bookings

Calpendo has multiple ways of finding out information about the [bookings](#)⁶⁵² apart from using the [Bookings Calendar](#)³⁹.

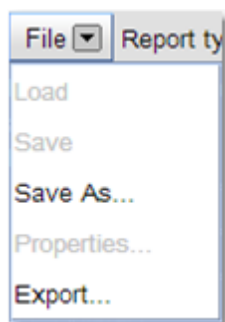
3.4.3.1 Using The Booking Searches

All the [booking](#)⁶⁵² search pages work in a similar way. This chapter explains the basic mechanisms for all the *booking* searches. To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

On entering any of the *booking* search pages a search will automatically start, of the appropriate type, with a date range of today. This is what a search will look like when it is completed.

File	Search for: Booking	Report type: List Report	17 Feb 2014	00:00	- 20 Feb 2014	23:59	Resource...	Status...	Project...	Conditions...	Columns...
Bookings where (dateRange between 17 Feb 2014 and 20 Feb 2014) and (status equals Requested or Approved)											
Found 14 bookings											
<input type="checkbox"/>	Resource	Booker	Owner	Project	Type	Status	Date Range	Created	Cancelled	Modified	All day Repeat
<input type="checkbox"/>	3T	admin (admin)	curie (Marie Curie)	Rad-1 (Radiation)	Animal	Approved	17 Feb 2014 14:30 - 18:30	17 Feb 2014 15:04		17 Feb 2014 15:04	false
<input type="checkbox"/>	Wet Lab	admin (admin)	darwin (Charles Darwin)	Gene-1 (Genes)		Approved	18 Feb 2014 13:00 - 15:00	18 Feb 2014 13:50		18 Feb 2014 13:50	false
<input type="checkbox"/>	Dry Lab	admin (admin)	curie (Marie Curie)	Rad-1 (Radiation)		Approved	19 Feb 2014 11:00 - 12:30	19 Feb 2014 11:11		19 Feb 2014 11:11	false
<input type="checkbox"/>	Telescope	curie (Marie Curie)	curie (Marie Curie)	Rad-1 (Radiation)	Mineral	Requested	19 Feb 2014 12:00 - 14:00	19 Feb 2014 12:11		19 Feb 2014 12:11	false
<input type="checkbox"/>	Electron Microscope	curie (Marie Curie)	einstein (Albert Einstein)	Space-1 (Space)		Approved	19 Feb 2014 12:30 - 14:30	19 Feb 2014 09:54		19 Feb 2014 09:54	false
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	einstein (Albert Einstein)	Space-1 (Space)		Requested	19 Feb 2014 12:30 - 15:00	19 Feb 2014 10:00		19 Feb 2014 10:00	false
<input type="checkbox"/>	Wet Lab	admin (admin)	darwin (Charles Darwin)	Gene-1 (Genes)		Approved	19 Feb 2014 13:30 - 15:00	19 Feb 2014 13:54		19 Feb 2014 13:54	false
<input type="checkbox"/>	Dry Lab	admin2 (admin2)	admin2 (admin2)	Rad-1 (Radiation)		Approved	19 Feb 2014 15:00 - 18:00	18 Feb 2014 13:46		18 Feb 2014 13:48	false
<input type="checkbox"/>	Electron Microscope	curie (Marie Curie)	admin2 (admin2)	Space-1 (Space)		Requested	19 Feb 2014 15:00 - 17:30	19 Feb 2014 09:58		19 Feb 2014 10:06	false
<input type="checkbox"/>	Leave	curie (Marie Curie)	curie (Marie Curie)			Approved	20 Feb 2014 00:00 - 23:59	20 Feb 2014 00:22		20 Feb 2014 00:22	true
<input type="checkbox"/>	Electron Microscope	admin (admin)	curie (Marie Curie)	Rad-1 (Radiation)	Animal	Approved	20 Feb 2014 11:00 - 13:00	20 Feb 2014 11:22		20 Feb 2014 11:22	false
<input type="checkbox"/>	3T	curie (Marie Curie)	curie (Marie Curie)	Rad-1 (Radiation)	Animal	Requested	20 Feb 2014 12:00 - 14:30	26 Sep 2013 16:54		26 Sep 2013 16:55	false Weekly
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	curie (Marie Curie)	Space-1 (Space)		Approved	20 Feb 2014 12:00 - 15:00	18 Feb 2014 14:17		18 Feb 2014 14:18	false
<input type="checkbox"/>	Wet Lab	admin (admin)	darwin (Charles Darwin)	Gene-1 (Genes)		Approved	20 Feb 2014 14:00 - 16:30	26 Sep 2013 16:08		26 Sep 2013 16:57	false Weekly

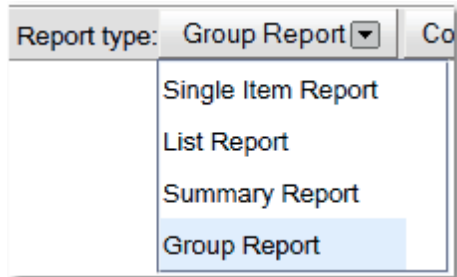
The following section goes through the options available after the search has completed, enabling the user to change the search, save it or export the data, as well as change the information viewed by the search. Not all options will be available for every *booking* search type, but that will be outlined in the specific search types chapter. For more information on any of the features shown read the [Search](#)¹¹⁸ chapter.



Save As: Saves the report for later use.

Export: Export the information found in the search to a file.

This button determines what type of report will be seen:



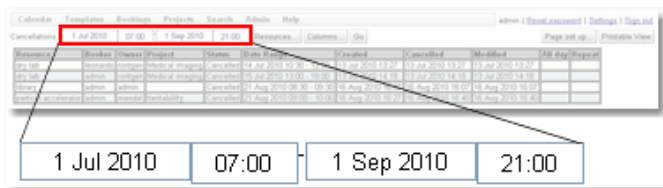
[Single Item Report](#)⁶⁵⁵ Displays a single record that matches the search. Does not appear for innumerable objects such as bookings.

[List Report](#)⁶⁵³: Lists each record found that matches the search one under the other.

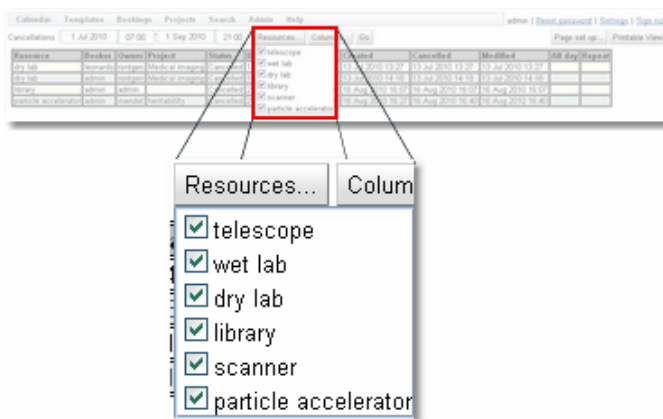
[Summary Report](#)⁶⁵⁶: Allows a summary of the information found.

[Group Report](#)⁶⁵³: Allows the report to be shown by groupings with a count of the number of records that fit each group.

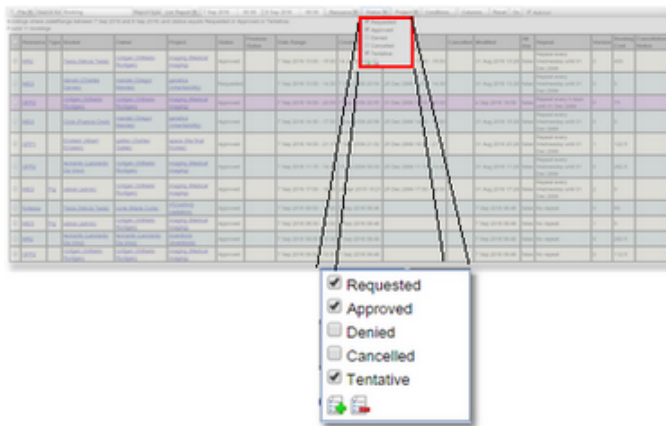
To change the search parameters



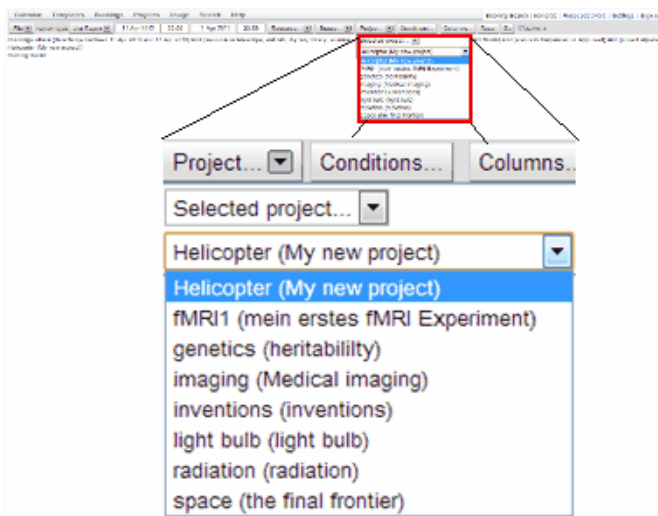
First select an appropriate range of dates for the new search.



Then select the [resources](#)⁶⁵⁵ required.

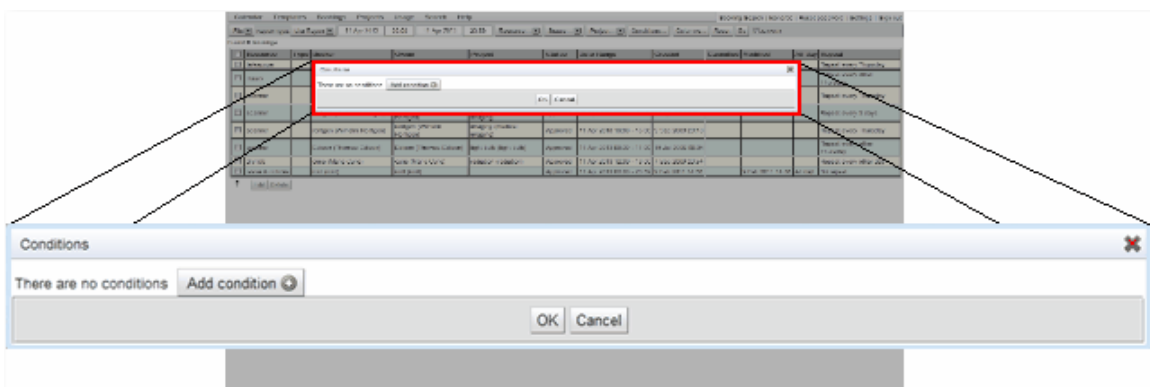


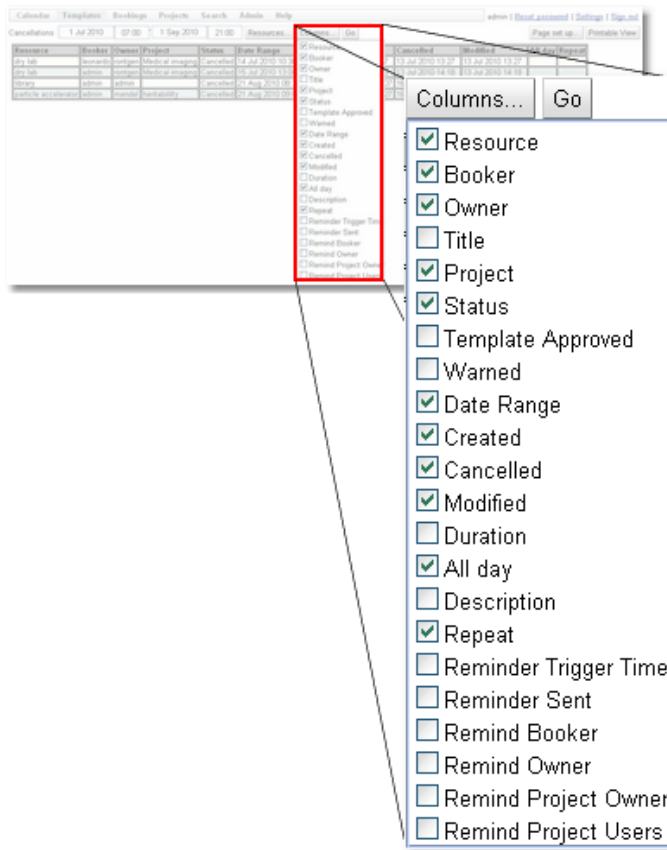
And select the required [booking statuses](#)⁶⁵³.



Then select the required [project](#)⁶⁵⁴.

Additional [conditions](#)⁶⁵³ for the search can also be set up. For more information on how to set up [conditions](#) read the information in the Setting Search Conditions page.





And then change the columns displayed in the output.

Reset: Resets the search *conditions* to the defaults set up on starting the search.

Go: Runs the search with the current *conditions*.



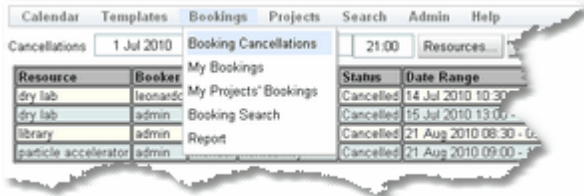
Autorun: If this button is ticked then searches will run as soon as any *conditions* change. If a number of the *conditions* of the search will be changed then it is more efficient if this is not ticked as each change will start a search and the appropriate transfer of data from the server.

For a more complete explanation on how to edit records in a booking search view read the [How to Edit Multiple Items At Once](#) ¹⁴⁰ section of the [Data Explorer](#) ¹³⁹ chapter. The only difference is in the booking search views as well as **Edit** and **Delete** there are also options to **Approve**, **Deny** or **Cancel** multiple *bookings*.

3.4.3.2 Booking Cancellations

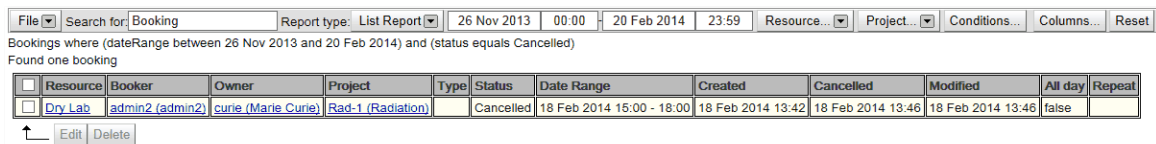
The **Booking Cancellations** search page allows searches for [bookings](#)⁶⁵² that have been cancelled.

By default, the **Booking Cancellations** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **Booking Cancellations** page looks like once a search has been done:



The **Booking Cancellation** search is run with the following default parameters:

Bookings where (**dateRange** between **start_date** and **end_date**) and ([status](#)⁶⁵³ equals **Cancelled**)

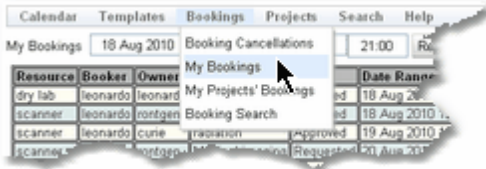
Because the *status* condition is set specifically to **Cancelled** there is no **Status** button available but the other buttons can be used to change the parameters of the search.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.4.3.3 My Bookings

The **My Bookings** search page does a search for [bookings](#)⁶⁵² that the user has created.

By default, the **My Bookings** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **My Bookings** page looks like once a search has been done:

File Search for Booking Report type: List Report 11 Feb 2014 00:00 20 Feb 2014 23:59 Resource... Status... Project... Conditions... Columns... Reset

Bookings where (dateRange between 11 Feb 2014 and 20 Feb 2014) and (status equals Requested or Approved) and (booker equals curie (Marie Curie))

Found 10 bookings

<input type="checkbox"/>	Resource	Booker	Owner	Project	Type	Status	Date Range	Created	Cancelled	Modified	All day	Repeat
<input type="checkbox"/>	3T	curie (Marie Curie)	curie (Marie Curie)	Rad-1 (Radiation)	Animal	Requested	20 Feb 2014 12:00 - 14:30	26 Sep 2013 16:54		26 Sep 2013 16:55	false	Weekly
<input type="checkbox"/>	Telescope	curie (Marie Curie)	curie (Marie Curie)	Rad-1 (Radiation)	Mineral	Requested	12 Feb 2014 12:00 - 14:00	26 Sep 2013 16:54		26 Sep 2013 16:54	false	
<input type="checkbox"/>	Leave	curie (Marie Curie)	curie (Marie Curie)			Approved	13 Feb 2014 00:00 - 23:59	14 Oct 2013 11:39		6 Feb 2014 12:46	true	
<input type="checkbox"/>	3T	curie (Marie Curie)	curie (Marie Curie)	Rad-1 (Radiation)	Animal	Requested	13 Feb 2014 12:00 - 14:30	26 Sep 2013 16:54		26 Sep 2013 16:55	false	
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	curie (Marie Curie)	Space-1 (Space)		Approved	20 Feb 2014 12:00 - 15:00	18 Feb 2014 14:17		18 Feb 2014 14:18	false	
<input type="checkbox"/>	Electron Microscope	curie (Marie Curie)	einstein (Albert Einstein)	Space-1 (Space)		Approved	19 Feb 2014 12:30 - 14:30	19 Feb 2014 09:54		19 Feb 2014 09:54	false	
<input type="checkbox"/>	Electron Microscope	curie (Marie Curie)	admin2 (admin2)	Space-1 (Space)		Requested	19 Feb 2014 15:00 - 17:30	19 Feb 2014 09:58		19 Feb 2014 10:06	false	
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	einstein (Albert Einstein)	Space-1 (Space)		Requested	19 Feb 2014 12:30 - 15:00	19 Feb 2014 10:00		19 Feb 2014 10:00	false	
<input type="checkbox"/>	Telescope	curie (Marie Curie)	curie (Marie Curie)	Rad-1 (Radiation)	Mineral	Requested	19 Feb 2014 12:00 - 14:00	19 Feb 2014 12:11		19 Feb 2014 12:11	false	
<input type="checkbox"/>	Leave	curie (Marie Curie)	curie (Marie Curie)			Approved	20 Feb 2014 00:00 - 23:59	20 Feb 2014 00:22		20 Feb 2014 00:22	true	

↑ Edit Delete Approve Deny Cancel

The **My Bookings** search is run with the following default parameters:

Bookings where (**dateRange** between **start_date** and **end_date**) and (**status**⁶⁵³ equals **Requested** or **Approved** or **Tentative**) and (**booker**⁶⁵² equals **current_user**)

All the possible buttons are available to change parameters of the search.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.4.3.4 My Projects' Bookings

The **My Projects' Bookings** page allows searches for [bookings](#)⁶⁵² that use [projects](#)⁶⁵⁴ that have been created by the user.

By default, the **My Projects' Bookings** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **My Projects' Bookings** page looks like once a search has been done:

File	Search for: Booking	Report type: List Report	21 Feb 2014	00:00	21 Feb 2014	23:59	Resource...	Status...	Conditions...	Columns...	Reset
Bookings where (dateRange between 21 Feb 2014 and 21 Feb 2014) and (status equals Requested or Approved) and (project is one of my projects)											
Found 4 bookings											
<input type="checkbox"/>	Resource	Booker	Owner	Project	Type	Status	Date Range	Created	Cancelled	Modified	All day Repeat
<input type="checkbox"/>	Telescope	admin (admin)	newton (Isaac Newton)	Space-1 (Space)	Human	Approved	21 Feb 2014 09:00 - 12:00	26 Sep 2013 15:04		14 Feb 2014 09:04	false Weekly
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	einstein (Albert Einstein)	Space-1 (Space)		Approved	21 Feb 2014 12:00 - 15:00	18 Feb 2014 14:19		18 Feb 2014 14:24	false
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	curie (Marie Curie)	Space-1 (Space)		Approved	21 Feb 2014 09:00 - 12:00	18 Feb 2014 14:25		18 Feb 2014 14:25	false
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	einstein (Albert Einstein)	Space-1 (Space)		Approved	21 Feb 2014 15:00 - 18:00	18 Feb 2014 14:26		18 Feb 2014 14:26	false
<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Approve"/> <input type="button" value="Deny"/> <input type="button" value="Cancel"/>											

The **My Projects' Booking** search is run with the following default parameters:

Bookings where (**dateRange** between **start_date** and **end_date**) and (**status**⁶⁵³ equals **Requested** or **Approved** or **Tentative**) and (**project** is **one of my projects**)

Because the *project* parameter is set specifically to **one of my projects** there is no **Project** button available but the other buttons can be used to change the parameters of the search.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.4.3.5 Booking Search

The **Booking Search** page sets up a search for all [bookings](#)⁶⁵² for a certain date range, it defaults to a date range which specifies today.

By default, the **Booking Search** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **Booking Search** page looks like once a search has been done:

 A screenshot of the Calpendo 'Booking Search' page. The page shows a search filter bar at the top with various dropdowns for 'Resource', 'Status', 'Project', 'Conditions', and 'Columns'. Below the filter bar, a table displays the search results. The table has columns: 'Resource', 'Type', 'Booker', 'Owner', 'Project', 'Status', 'Date Range', 'Created', 'Cancelled', 'Modified', 'All day', and 'Repeat'. There are 8 rows of data.

Resource	Type	Booker	Owner	Project	Status	Date Range	Created	Cancelled	Modified	All day	Repeat
telescope		Einstein (Albert Einstein)	galileo (Galileo Galilei)	space (the final frontier)	Requested	11 Apr 2013 21:00 - 23:50	14 Jul 2009 22:20				Repeat every Thursday
library		Planck (Max Planck)	Edison (Thomas Edison)	light bulb (light bulb)	Approved	11 Apr 2013 14:30 - 17:00	14 Jul 2009 22:29				Repeat every other Thursday
scanner		leonardo (Leonardo Da Vinci)	curie (Marie Curie)	radiation (radiation)	Approved	11 Apr 2013 15:30 - 18:30	15 Jul 2009 00:05				Repeat every Thursday
scanner		roentgen (Wilhelm Rontgen)	roentgen (Wilhelm Rontgen)	imaging (Medical imaging)	Approved	11 Apr 2013 19:00 - 20:00	7 Sep 2009 20:55		7 Feb 2012 13:42		Repeat every 3 days
scanner		roentgen (Wilhelm Rontgen)	roentgen (Wilhelm Rontgen)	imaging (Medical imaging)	Approved	11 Apr 2013 13:30 - 15:00	9 Sep 2009 20:10				Repeat every Thursday
dry lab		curie (Marie Curie)	curie (Marie Curie)	radiation (radiation)	Approved	11 Apr 2013 12:00 - 13:00	7 Sep 2009 20:54				Repeat every other day
leave & notices		root (root)	root (root)		Approved	11 Apr 2013 00:00 - 23:59	9 Feb 2011 14:56		9 Feb 2011 14:56	All day	No repeat
scanner		Edison (Thomas Edison)	Edison (Thomas Edison)	light bulb (light bulb)	Approved	11 Apr 2013 08:30 - 11:00	15 Jul 2009 00:04				No repeat

The **Booking** search is run with the following default parameters:

Bookings where (**dateRange** between **start_date** and **end_date**) and (**status**⁶⁵³ is **Requested** or **Approved** or **Tentative**)

All the possible buttons are available to change parameters of the search.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.5 Projects

Calpendo allows management of [projects](#)⁶⁵⁴ that can be associated with [bookings](#)⁶⁵². This helps keep track of what the [resources](#)⁶⁵⁵ are being used for, but can also be used as a means of controlling who can book which *resources*. A *project* can have multiple users associated with it, and each user may only book for those *projects* that they are associated with.

The information stored within a *project* can be customised for the facility, as described in [Project Configuration](#)²¹⁷. The screen shots that are shown in this section are likely to be different from those that the user in your facility would see, depending on how the *projects* have been set up.

3.5.1 Creating Projects

To create a new [project](#)⁶⁵⁴, go to the menu option **Projects-->Create Project**.

The content viewed when creating a new *project* will vary depending on how the systems administrator has configured **Calpendo**. By default each *project* contains the following:

Property	Description
Project Code	This is an identifier that will be assigned by an administrator. These should be unique, but <i>projects</i> may not have a <i>project code</i> assigned to them until the project approval process ⁶⁵⁴ has completed.
Type	The facility may, or may not, classify <i>projects</i> by type ⁶⁵⁵ . If types are configured for use, then the user will be able to select the <i>type</i> that should be used for your <i>project</i> .
Status	Newly created <i>projects</i> will have a <i>status</i> of Requested . The user will probably not be able to modify the <i>status</i> of their own <i>projects</i> , but this depends on how the Permissions ⁶⁵⁴ are configured to allow access.
Owner	The owner of a <i>project</i> is normally the person that created the <i>project</i> request. The user may be able to select a different owner, but the administrator may also reassign the ownership of a <i>project</i> . Owners can also use the project for booking without being in the user list.
Name	This should be a <i>short</i> text description of the <i>project</i> . The maximum length that this can be will have been configured by the administrator.
Description	This is a longer description of your <i>project</i> .
Project Resource Settings	This specifies the resources ⁶⁵⁵ that the <i>project</i> can book, and provides a place to store some information relating to what will be used. For example, this may record how much the <i>project</i> will be charged for using each <i>resource</i> or how much time in total is allowed to be booked for the <i>project</i> .
Project Service Settings	This specifies the services that the <i>project</i> can order, and provides a place to store some information relating to what will be used. For example, this may record how much the <i>project</i> will be charged for ordering each service.
Users	This specifies all the users that are allowed to make bookings ⁶⁵² on behalf of the <i>project</i> .

In the default system the following properties are also on Project, but may be removed or used as required.

Property	Description
Phone Number	Phone number to be associated with the <i>project</i> .
Start	Start date of the <i>project</i> .
Finish	Finish date of the <i>project</i> .
Principle Investigator	The main investigator of the <i>project</i> .
Ethics Approval Number	The Ethics Approval Number for the <i>Project</i> .
Funding Agency	Name of the funding agency.
Account Number	Account number for the funding.

Once the relevant information has been filled in, press the **Submit project request** button.

The administrator may have added a number of other pieces of information that need to be completed for a *project* to be created. If there are problems filling out the *project* creation form please talk to the **Calpendo** administrator.

The process of approving *projects* can vary a great deal between different **Calpendo** installations. The administrator should be able to tell the user about the process that's in place at the facility. However, in all cases, a *project* becomes available for booking once its status is changed to **Approved**.

See [The Project Approval Process](#)¹⁵⁸ for information on the options for approving *projects*.

As soon as a *project* request has been created, it will appear on the [My Projects](#)⁸² page.

Here is an example of creating a *project*, click on **Create Project**,

Project creation starts with the **General** tab. Fill in the form and then move on to the other tabs.

Click on **Choose Resource** to get a pop up which will allow the choice of *resources* to be used for this *project*. View the section on Resource Selection to see how to select *resources* using this pop up.

Click on **Project Service Settings** to get a pop up which will allow the project manager to choose the services available to be ordered by this project.

The current user name will appear automatically in the user list. Use the drop down to select other users to be added to the list.

To remove a user, tick the option box and click **Remove**.

Now submit the *project* request using the **Save** button.

The **Project Groups** tab cannot be used until the *project* is initially saved.

3.5.2 Modifying Projects

Modification of [projects](#)⁶⁵⁴ is done using the [My Projects](#)⁸² and [Project Search](#)⁷⁸ pages. Each of those pages shows a list of *projects*, and allows the selection of a particular *project* so that its full details can be seen, and also to be able to edit it. See the documentation for [My Projects](#)⁸² and [Project Search](#)⁷⁸ for full details. Note that the ability to edit a *project* is always subject to the user having [permission](#)⁶⁵⁴ to do so.

3.5.3 Searching For Projects

Calpendo has multiple ways finding out information about the [projects](#)⁶⁵⁴ that are being used.

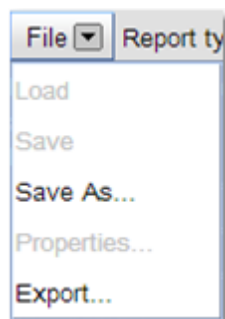
3.5.3.1 Using The Project Searches

All the [project](#)⁶⁵⁴ search pages work in a similar way. This chapter explains the basic mechanisms for all the *project* searches. To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

On entering any of the *project* search pages **Calpendo** will automatically start a search of the appropriate type with a date range of today. This is what a search will look like when it is completed.

File	Report type: List Report	Type...	Owner...	Status...	Conditions...	Columns...	Reset	Go	<input checked="" type="checkbox"/> Autorun
Projects where (status equals Requested or Approved)									
Found 3 projects									
<input type="checkbox"/>	Project Code	Type	Status	Owner	Name	Professor Name	Sign Off	Sign Off Date	Profesor Role
<input type="checkbox"/>	Space-1	Physics	Approved	einstein (Albert Einstein)	Space		false		
<input type="checkbox"/>	Gene-1	Biology	Approved	darwin (Charles Darwin)	Genes		false		
<input type="checkbox"/>	Rad-1	Physics	Approved	curie (Marie Curie)	Radiation		false		
<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Approve"/> <input type="button" value="Deny"/> <input type="button" value="Terminate"/>									

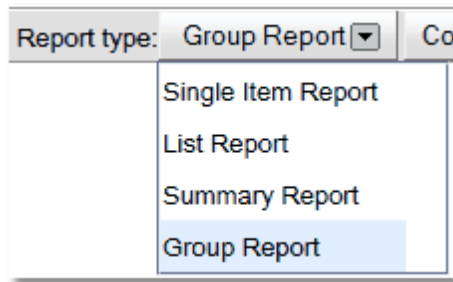
The following section goes through the options available after the search has completed, enabling the user to change the search, save it or export the data, as well as change the information viewed by the search. Not all options will be available for every *project* search type, but that will be outlined in the specific search types chapter. For more information on any of the features shown you will need to read the [Search](#)¹¹⁸ chapter.



Save As: Saves the created report for later use.

Export: Exports the information found in the search to a file.

This button determines which type of report will be seen:



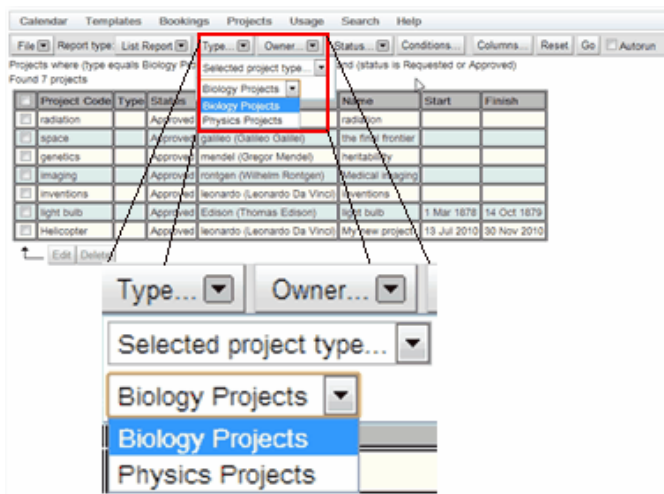
Single Item Report⁶⁵⁵: Displays a single record that matches the search. Does not appear for innumerable objects such as bookings.

List Report⁶⁵³: Lists each record found that matches the search one under the other.

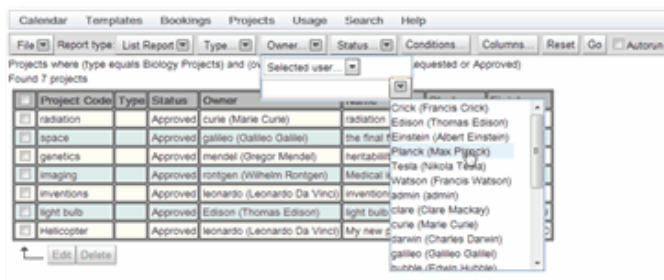
Summary Report⁶⁵⁶: Summarises the information found.

Group Report⁶⁵³: Shows the report by groupings with a count of the number of records that fit each group.

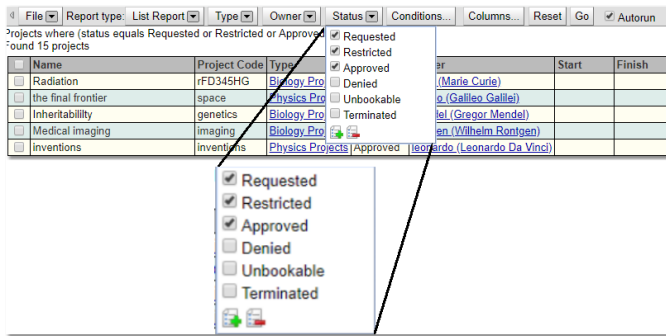
To change the search parameters



First select the type of *project* required



Then select the owner needed.



And select the required [project statuses](#) ⁶⁵⁴.

Requested: Requested *projects* are those that are still going through the [project approval process](#) ⁶⁵⁴.

Restricted: Only Admins and Resource Managers can use these projects for making [bookings](#) ⁶⁵².

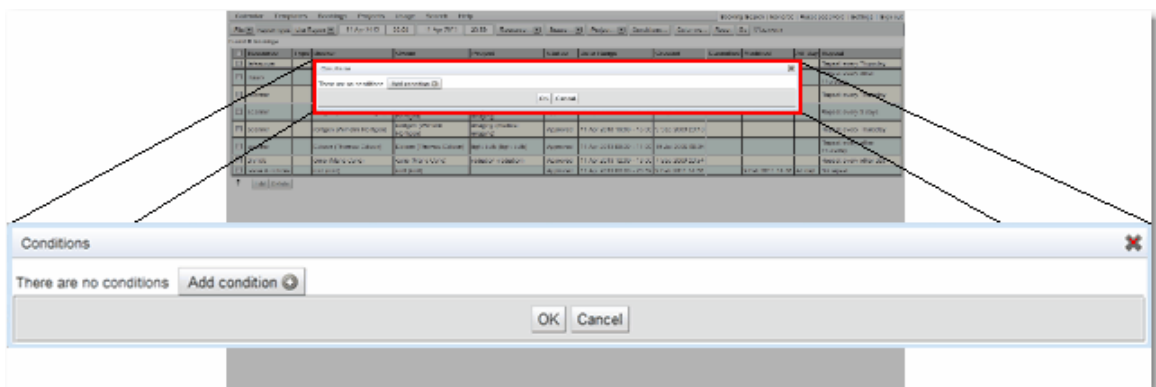
Approved: This is the normal *status* used for *projects* that are approved for making bookings.

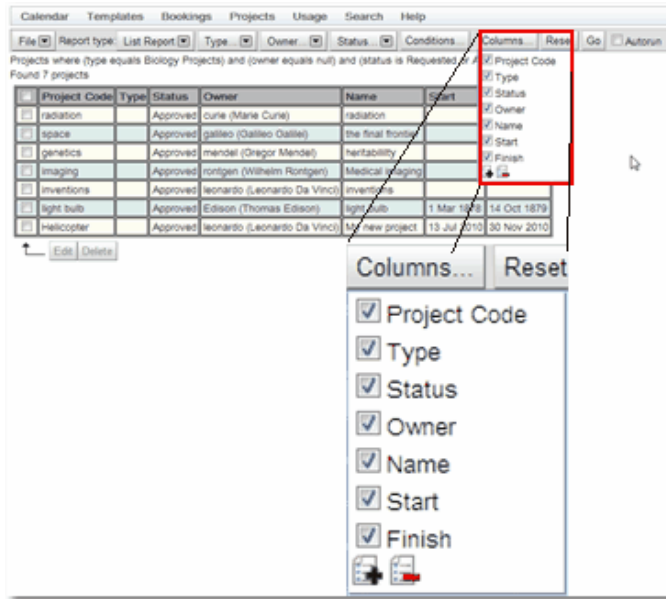
Denied: a *project* request is turned down, then its *status* may be set to **Denied**.

Unbookable: Some special *projects* may have a status of **Unbookable**. As the name suggests, these projects are not available for *bookings*. Typically, this is used only for the *project* that's used as a template for newly created *projects*.

Terminated: This is the *status* used for *projects* that are no longer active

Additional [conditions](#) ⁶⁵³ for your search can also be set up. For more information on how to set up *conditions* read the information in the [Search](#) ¹¹⁸ page.





And finally change the columns displayed in the output.

Reset: Resets the search *conditions* to the defaults set up on starting the search.

Go: Runs the search with the current *conditions*



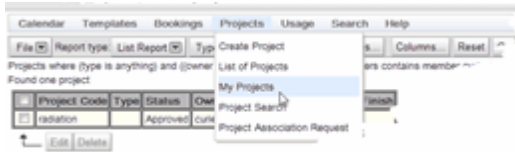
Autorun: If this button is ticked then searches will run as soon as any *conditions* change. If a number of the *conditions* of the search will be changed then it is more efficient if this is not ticked as each change will start a search and the appropriate transfer of data from the server.

For a more complete explanation on how to edit records in a *project* search view read the [How to Edit Multiple Items At Once](#)¹⁴⁰ section of the [Data Explorer](#)¹³⁹ chapter. The only difference is in the booking search views as well as **Edit** and **Delete** there are also options to **Approve**, **Deny** or **Terminate** multiple *projects*.

3.5.3.2 My Projects

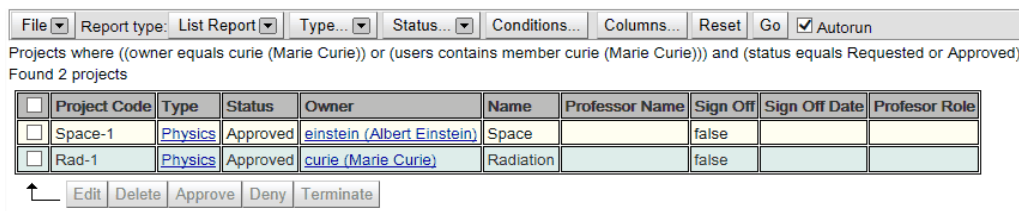
The **My Projects** search page does a search for [projects](#)⁶⁵⁴ that the user have created or is a member of.

By default, the **My Projects** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **My Projects** page looks like once a search has been done:



The **My Projects** search is run with the following default parameters:

*Projects where ((owner equals **current_user**) or (users contains member **current_user**))) and ([status](#)⁶⁵⁴ equals **Requested** or **Approved**)*

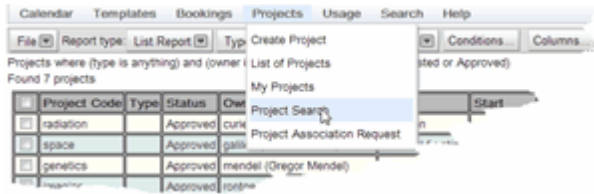
Because the **owner** parameter is set specifically to **current_user** there is no **Owner** button available but the other buttons can be used to change the parameters of the search.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.5.3.3 Project Search

The **Project Search** page provides a convenient way to search for [projects](#)⁶⁵⁴. By default it returns all the **Requested** or **Approved** projects.

By default, the **Project Search** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **Project Search** page looks like once a search has been done:

File Report type: List Report Type... Owner... Status... Conditions... Columns... Reset Go ☒ Autorun

Projects where (status equals Requested or Approved)
Found 3 projects

<input type="checkbox"/>	Project Code	Type	Status	Owner	Name	Professor Name	Sign Off	Sign Off Date	Professor Role
<input type="checkbox"/>	Space-1	Physics	Approved	einstein (Albert Einstein)	Space		false		
<input type="checkbox"/>	Gene-1	Biology	Approved	darwin (Charles Darwin)	Genes		false		
<input type="checkbox"/>	Rad-1	Physics	Approved	curie (Marie Curie)	Radiation		false		

↑ Edit Delete Approve Deny Terminate

The **Projects Search** is run with the following default parameters:

*Projects where ([status](#)⁶⁵⁴ equals **Requested** or **Approved**)*

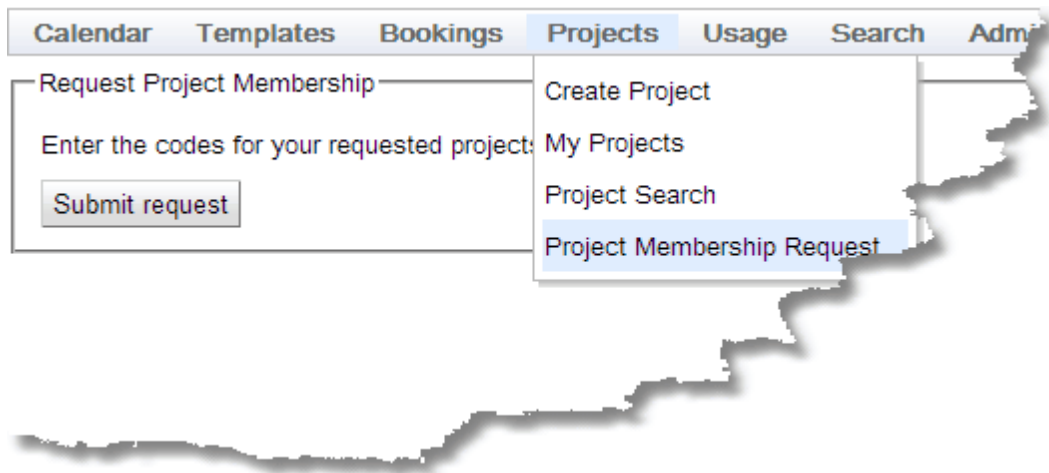
All the possible buttons are available to change parameters of the search.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

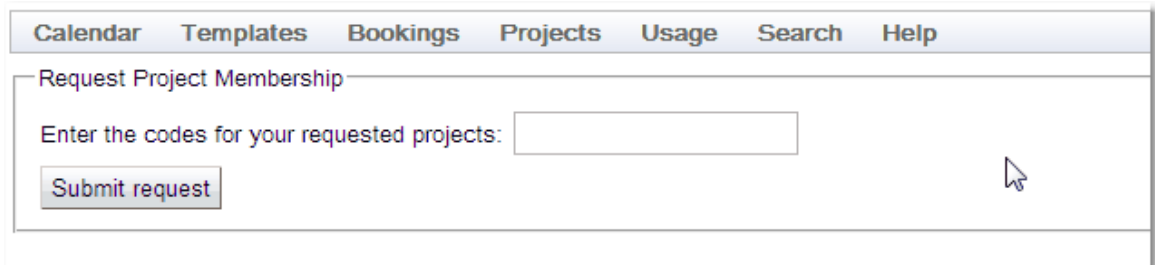
3.5.4 Project Membership Request

If a user would like to be associated with a [project](#)⁶⁵⁴ but they don't have sufficient [permission](#)⁶⁵⁴ to edit the *project* and add themselves, then they can ask the administrator or they can make a request using the **Project Membership Request** page.

The default place on the menu for this page is at **Projects-->Project Membership Request**. However, **Calpendo** may have been configured so that this page is called something else, or is not shown at all.



Enter the [code](#)⁶⁵⁴(s) for the *project(s)* for which a joining request is needed, and press the **Submit Request** button. This will notify the administrator who will see whatever text has been entered. There is no automated response to this. Once the request has been submitted, wait for your administrator to respond to it.



3.6 Resource Usage

While **Calpendo** is primarily about [bookings](#)⁶⁵², it can also record the actual [usage](#)⁶⁵⁵ of a [resource](#)⁶⁵⁵. This is done with a page that essentially offers a start and a stop button, and captures information about what the *resource usage* was for and whether it went well or not. Typically, this would be done with a computer that is next to the equipment whose *resource usage* is being recorded.

The administrator may have configured *resource usage* recording only for some *resources*, or for none of them. If *resource usage* recording is disabled for all *resources*, then the menu options relating to *resource usage* are not shown.

3.6.1 Resource Usage Session Recorder

The [Resource Usage Session Recorder](#)⁶⁵⁵ provides the means for recording [resource usage](#)⁶⁵⁵, and is split into three sub-pages:

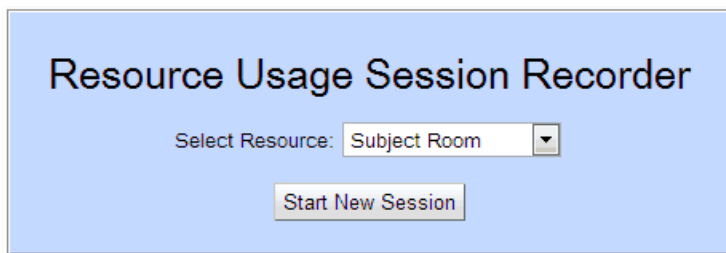
- The **Resource Selector** page that allows selection of the [resource](#)⁶⁵⁵ to enter *usage* for. Once a *resource* is selected, either the pre-usage page or the during-usage page are open, depending on whether there is currently ongoing *resource usage* activity.
- The **Pre-Usage** page, that gathers information about what the *resource* is going to be used for and lets the user indicate that *resource usage* has actually started.
- The **During Usage** page that shows limited information about what the *resource* is being used for, when it started and how long it's been going for. It also provides a **Stop Session** button to indicate that *resource usage* has stopped.

A normal user can record *usage* for *resources* that do not need [projects](#)⁶⁵⁴ associated with them or for *resources* where they are a member of the *project* that is using the *resource*. An administrator can record *usage* for any *resource* and *project* combination. This means that if the *resource usage* information is being recorded by operators they either need to be a member of all *projects* that use the *resource* they are operating or be logged in as an administrator.

Resource Selector Page

This page allows selection of which *resource* to enter *usage* for, or to view or terminate the current *resource usage* of.

If there are no current *resource usages* being collected then the following form will come up to choose the *resource* and start a new session and then move to the **Pre-Usage** page.



If there is a current *resource usage* being collected then the **During Usage** page will be opened.

Pre-Usage Page

This page lets you enter the following:

Setting	Description
Booking ⁶⁵²	Select a <i>booking</i> if a <i>booking</i> is to be associated with this <i>resource usage</i> . There may be no <i>booking</i> though, so this can be left unselected.
<i>Project</i>	This is only offered as an option if the <i>resource</i> has been configured to require a <i>project</i> for its <i>bookings</i> .
User	Specifies the user that is using the <i>resource</i> . A standard user may only select themselves. Administrators may select anyone.
Other	<p>More information can be collected at this point, if required. This requires that the Resource Usage definition be modified by the administrator to add whatever additional information is required. For example, a user may want to capture information such as patient name, DOB and hospital number, whether this is a patient or healthy volunteer and any other required data.</p> <p>For each configured item, a user can specify whether Calpendo will store it in its database or not. (In a future version, it will be able to send data to another system via http/https, and this will include those things Calpendo doesn't store in its own database)</p>

First select the *booking* to record *resource usage* for. When selecting a *booking*, the pop-up **Booking Selector** will display the *bookings* the user has [permission](#)⁶⁵⁴ to access for the current *resource* around the current date and time. This should make it easy to select the right *booking*.

Session Recorder for scanner

Booking: Select Booking

Project: Please select a Project

User: Please select a User

Start Session

Select Booking

Date	Start	Finish	Project	Booker	Booking ID
11 Apr 2013	19:00	20:00	imaging (Medical imaging)	Wilhelm Rontgen	8,905
12 Apr 2013	09:00	11:00	imaging (Medical imaging)	Marie Curie	8,908
12 Apr 2013	11:00	13:00	imaging (Medical imaging)	Leonardo Da Vinci	8,909
12 Apr 2013	13:00	16:30	radiation (radiation)	Marie Curie	8,907
14 Apr 2013	19:00	20:00	imaging (Medical imaging)	Wilhelm Rontgen	78

Earlier Now Later

OK Clear Selection Cancel

If a *project* is requested, then one must be selected. Note that if a *booking* is selected, then the *project* will be automatically selected. Also select the user that is using the *resource*, here again *Permissions* may apply. Once the user, and possibly the *project*, is selected, then press the **Start Session** button when the *resource usage* begins. At this point, **Calpendo** records the time and other information entered, and displays the **During Usage** page.

During Usage Page

This page shows the following information in a read-only format:

- The *resource*, together with a button to allow the *resource* to be changed, by going back to the **Resource Selector** page.
- The *project* (if applicable)
- The user
- The session ID that has been allocated. This displays as a hyper link that will display detailed information about the *resource usage* that has been recorded so far.
- The time that the *resource usage* started
- How long *resource usage* has been ongoing

There is also a **Stop Session** button, which should be pressed when *resource usage* ceases.

Session Recorder for scanner	
Booking	ID 8907, 12 Apr 2013 13:00 - 16:30
Project	radiation (radiation)
User	curie (Marie Curie)
Session ID	8

Started 10:58, current duration 0 minutes

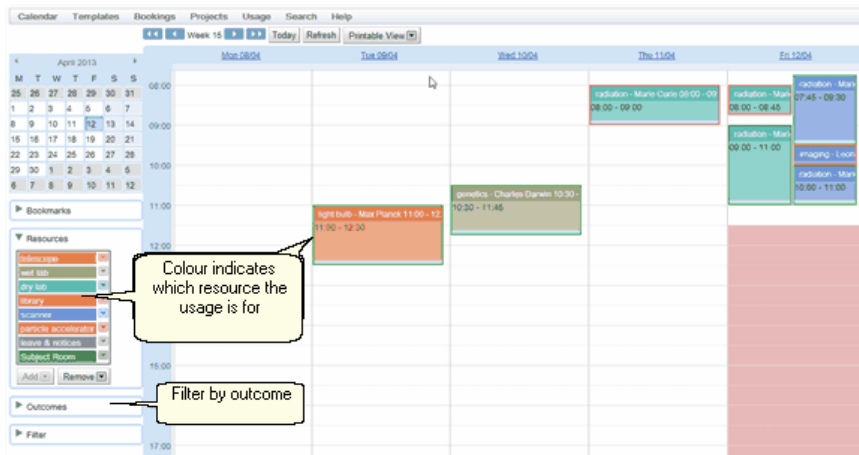
Stop Session

At this point, a pop-up will appear asking whether the session was successful. If the answer is no, then it asks for the outcome to be selected from a list of options that the administrator will have configured. The user may also provide a comment.

After completing the pop-up, the view will return to the **Pre-Usage** page.

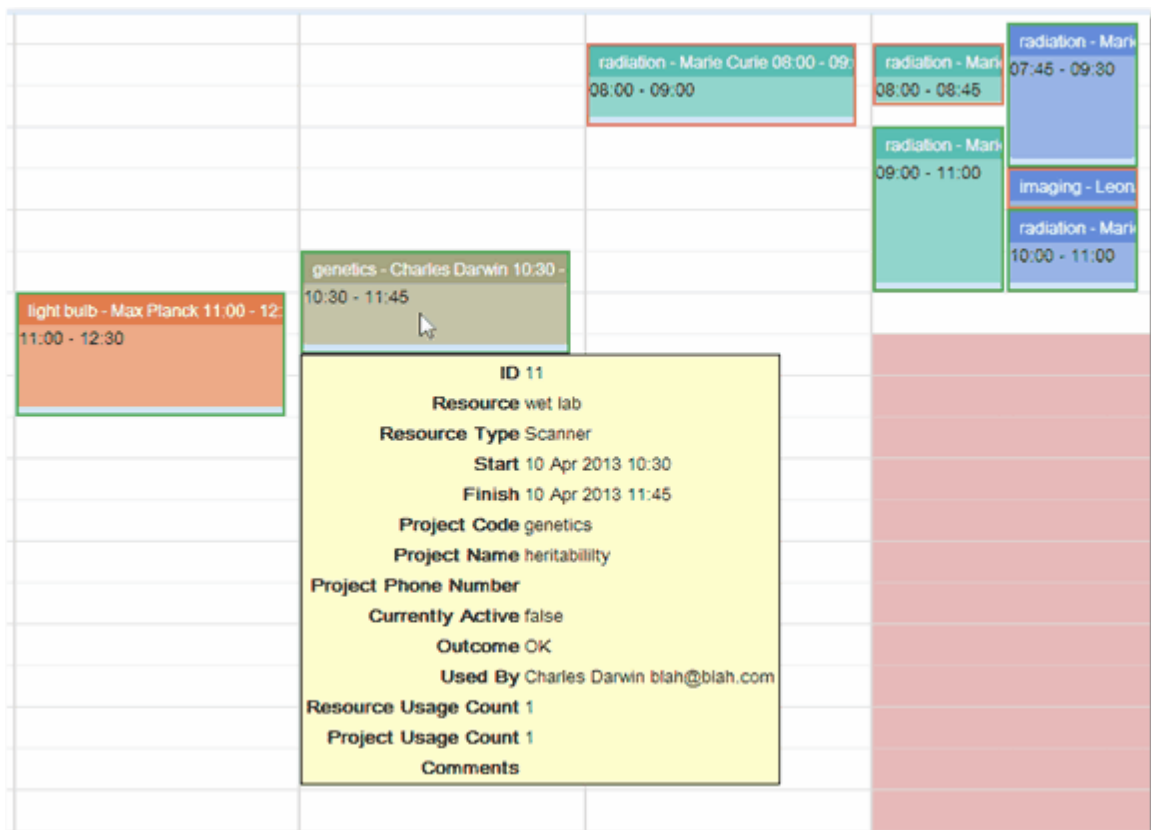
3.6.2 Resource Usage Calendar

The [Resource Usage Calendar](#)⁶⁵⁵ shows a calendar much like the [Bookings Calendar](#)⁶⁵², except that everything is displayed in the past. That's because [resource usage](#)⁶⁵⁵ that hasn't happened yet cannot be recorded. The calendar shows *usage* for any *resource* and gives the ability to filter that *usage* as required. If the user has suitable [permission](#)⁶⁵⁴, then they can also create and modify *resource usage* from the *calendar*. This may be necessary if something prevented *usage* being recorded as a *resource* was being used.

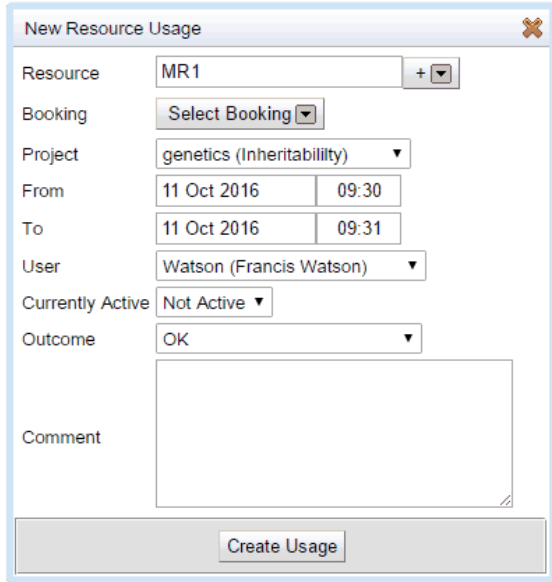


Filter by outcome allows the *resource usage*'s shown to be filtered by the outcome of the *resource usage*. These outcomes will have been configured by the administrator.

By moving the cursor over a usage the *resource usage* information can be viewed.



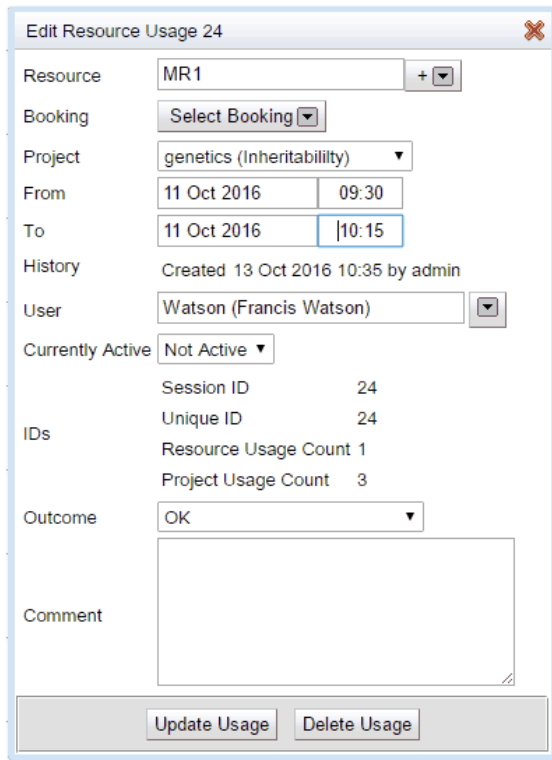
The user may need to create a *resource usage* record after the fact due to problems at the time, to do this select the time area for the resource usage and the resource usage create pop up will appear Fill in the *Resource*, *Booking*⁶⁵², *Project*⁶⁵⁴ and *User* information. Once the **Outcome** is chosen the **Create Usage** button becomes active and can be pressed to create the *resource usage* information.



The 'New Resource Usage' dialog box contains the following fields and controls:

- Resource:** Text field with 'MR1' and a dropdown arrow.
- Booking:** Button labeled 'Select Booking' with a dropdown arrow.
- Project:** Dropdown menu showing 'genetics (Inheritability)'.
- From:** Date and time fields showing '11 Oct 2016' and '09:30'.
- To:** Date and time fields showing '11 Oct 2016' and '09:31'.
- User:** Dropdown menu showing 'Watson (Francis Watson)'.
- Currently Active:** Dropdown menu showing 'Not Active'.
- Outcome:** Dropdown menu showing 'OK'.
- Comment:** Large text area for notes.
- Create Usage:** Button at the bottom right.

If the user has the *permissions* they can edit a resource usage's information. Click on the *resource usage* to be edited and choose edit from the pop up menu



The 'Edit Resource Usage' dialog box contains the following fields and controls:

- Resource:** Text field with 'MR1' and a dropdown arrow.
- Booking:** Button labeled 'Select Booking' with a dropdown arrow.
- Project:** Dropdown menu showing 'genetics (Inheritability)'.
- From:** Date and time fields showing '11 Oct 2016' and '09:30'.
- To:** Date and time fields showing '11 Oct 2016' and '10:15'.
- History:** Text field showing 'Created 13 Oct 2016 10:35 by admin'.
- User:** Dropdown menu showing 'Watson (Francis Watson)'.
- Currently Active:** Dropdown menu showing 'Not Active'.
- IDs:** Section containing:
 - Session ID: 24
 - Unique ID: 24
 - Resource Usage Count: 1
 - Project Usage Count: 3
- Outcome:** Dropdown menu showing 'OK'.
- Comment:** Large text area for notes.
- Update Usage:** Button at the bottom left.
- Delete Usage:** Button at the bottom right.

Once the required edits are completed, press **Update Usage** to make the changes.

3.6.3 Searching For Resource Usage

Calpendo has multiple ways of finding out information about [resource usage](#)⁶⁵⁵.

3.6.3.1 Using the Resource Usage Searches

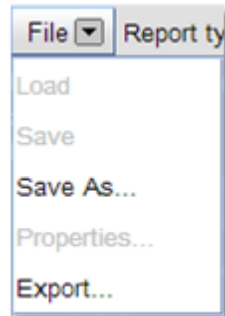
All the [usage](#)⁶⁵⁵ search pages work in a similar way. This chapter explains the basic mechanisms for all the *resource usage* searches. To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

When entered any of the *resource usage* search pages you will automatically start a search of the appropriate type with a date range of today. This is what a search will look like when it is completed.

File	Report type: List Report	10 Feb 2014 00:00	20 Feb 2014 23:59	Resource...	Project...	Conditions...	Columns...	Reset	Go
Resource Usages where (dateRange between 10 Feb 2014 and 20 Feb 2014) and (user equals curie (Marie Curie))									
Found 2 resource usages									
<input type="checkbox"/>	Resource	User	Project	Outcome	Date Range	Created	Creator	Modified	Modifier
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	Space-1 (Space)	OK	20 Feb 2014 11:00 - 20 Feb 2014 13:00	20 Feb 2014 13:10	curie (Marie Curie)		
<input type="checkbox"/>	3T	curie (Marie Curie)	Rad-1 (Radiation)	OK	20 Feb 2014 09:00 - 20 Feb 2014 11:00	20 Feb 2014 13:11	curie (Marie Curie)		

↑ Edit Delete

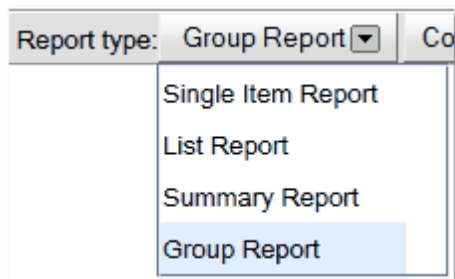
The following section goes through the options available after the search has completed, enabling the search parameters to be changed, saved, or the data exported, as well as change the information viewed by the search. Not all options will be available for every *resource usage* search type, but that will be outlined in the specific search types chapter. For more information on any of the features shown read the [Search](#)¹¹⁸ chapter.



Save As: Saves the report for later use.

Export: Export the information found in the search to a file.

This button determines what type of report will be seen:



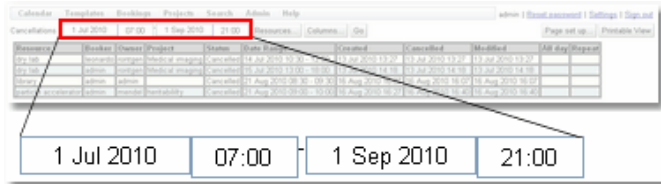
[Single Item Report](#)⁶⁵⁵ Displays a single record that matches the search. Does not appear for innumerable objects such as usage.

[List Report](#)⁶⁵³: Lists each record found that matches the search one under the other.

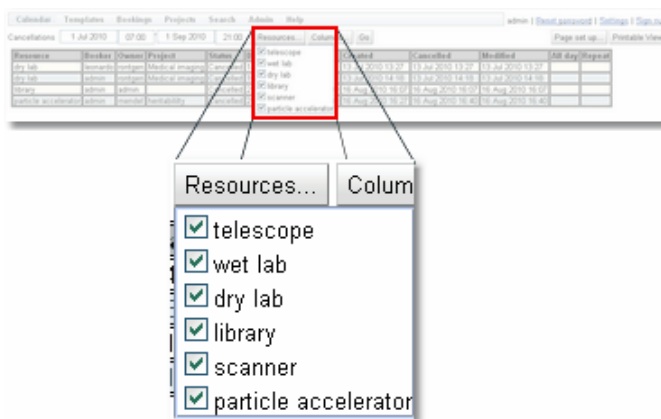
[Summary Report](#)⁶⁵⁶: Allows a summary of the information found.

Group Report⁶⁵³: Allows the report to be shown by groupings with a count of the number of records that fit each group.

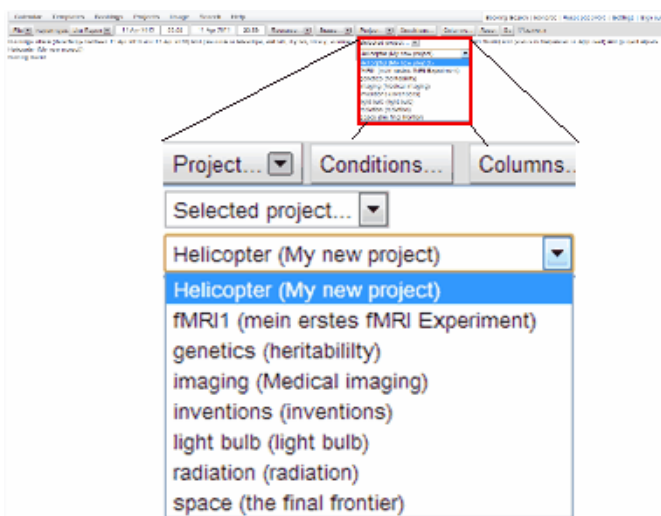
To change the search parameters



First select an appropriate range of dates for the new search.

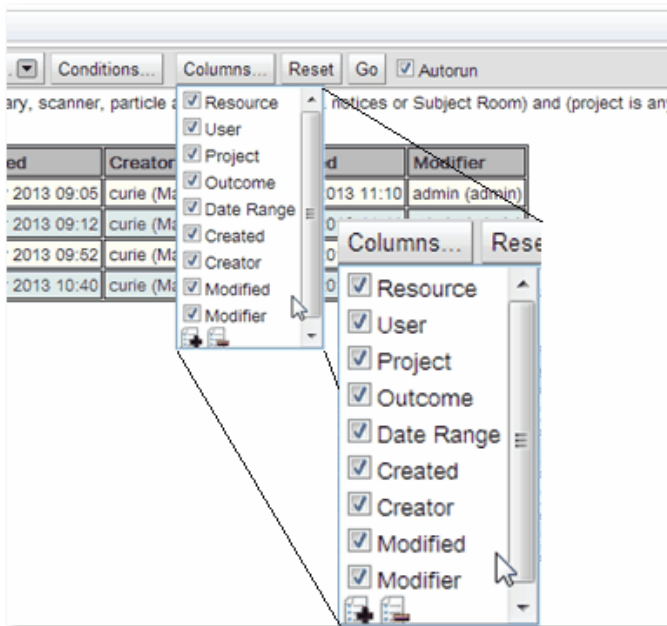
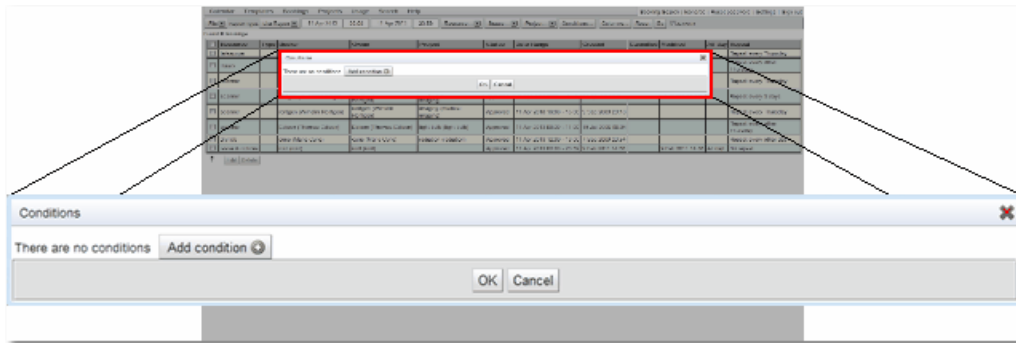


Then select the **resources**⁶⁵⁵ required.



Then select the **project**⁶⁵⁴ required.

Additional **conditions**⁶⁵³ for the search may also be set up. For more information on how to set up **conditions** read the information in the **Search**¹¹⁸ page.



Finally change the columns displayed in the output.

Reset: Resets the search *conditions* to the defaults set up on starting the search.

Go: Runs the search with the current *conditions*



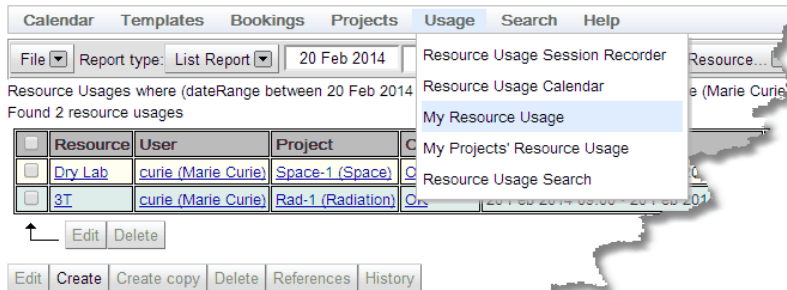
Autorun: If this button is ticked then searches will run as soon as any *conditions* change. If a number of the *conditions* of the search will be changed then it is more efficient if this is not ticked as each change will start a search and the appropriate transfer of data from the server.

For a more complete explanation on how to edit records in a *usage* search view read the [How to Edit Multiple Items At Once](#)¹⁴⁰ section of the [Data Explorer](#)¹³⁹ chapter.

3.6.3.2 My Resource Usage

The **My Resource Usage** search page searches for [resource usage](#)⁶⁵⁵, restricting any output to *resource usage* used by the current user.

By default, the **My Resource Usage** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **My Resource Usage** page looks like once a search has been done:

File	Report type: List Report	10 Feb 2014 00:00	20 Feb 2014 23:59	Resource...	Project...	Conditions...	Columns...	Reset	Go
Resource Usages where (dateRange between 10 Feb 2014 and 20 Feb 2014) and (user equals curie (Marie Curie))									
Found 2 resource usages									
<input type="checkbox"/>	Resource	User	Project	Outcome	Date Range	Created	Creator	Modified	Modifier
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	Space-1 (Space)	OK	20 Feb 2014 11:00 - 20 Feb 2014 13:00	20 Feb 2014 13:10	curie (Marie Curie)		
<input type="checkbox"/>	3T	curie (Marie Curie)	Rad-1 (Radiation)	OK	20 Feb 2014 09:00 - 20 Feb 2014 11:00	20 Feb 2014 13:11	curie (Marie Curie)		

The **My Resource Usage** search is run with the following default parameters:

Resource Usages where (**dateRange** between **start_date** and **end_date**) and (**user** equals **current_user**)

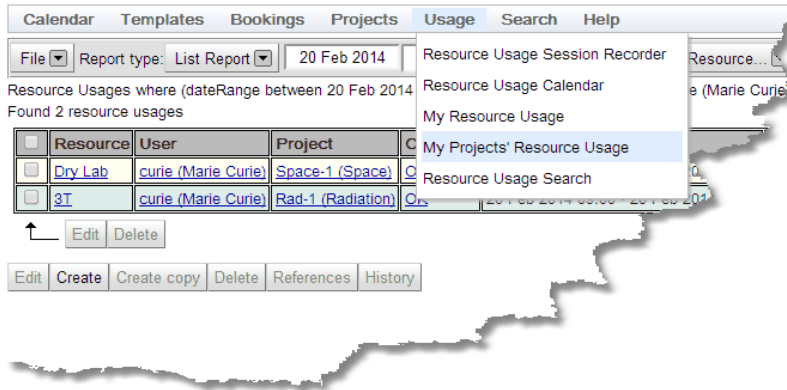
All the possible buttons are available to change parameters of the search.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.6.3.3 My Projects' Resource Usage

The **My Projects' Resource Usage** search page lets you search for [resource usage](#)⁶⁵⁵ restricted output to *resource usage* used by the current user's [projects](#)⁶⁵⁴.

By default, the **My Projects' Resource Usage** page appears on the menu here:



However, your administrator may have configured **Calpendo** so that the menu is different.

This is what the **My Projects' Resource Usage** page looks like once a search has been done:

File	Report type: List Report	20 Feb 2014	00:00	20 Feb 2014	23:59	Resource...	Conditions...	Columns...	Reset	Go	<input checked="" type="checkbox"/> Autorun
Resource Usages where (dateRange between 20 Feb 2014 and 20 Feb 2014) and (project is one of my projects)											
Found 2 resource usages											
<input type="checkbox"/>	Resource	User	Project	Outcome	Date Range	Created	Creator	Modified	Modifier		
<input type="checkbox"/>	Dry Lab	curie (Marie Curie)	Space-1 (Space)	OK	20 Feb 2014 11:00 - 20 Feb 2014 13:00	20 Feb 2014 13:10	curie (Marie Curie)				
<input type="checkbox"/>	3T	curie (Marie Curie)	Rad-1 (Radiation)	OK	20 Feb 2014 09:00 - 20 Feb 2014 11:00	20 Feb 2014 13:11	curie (Marie Curie)				

The **My Projects' Resource Usage** search is run with the following default parameters:

Resource Usages where (**dateRange** between **start_date** and **end_date**) and (*project* is **one of my projects**)

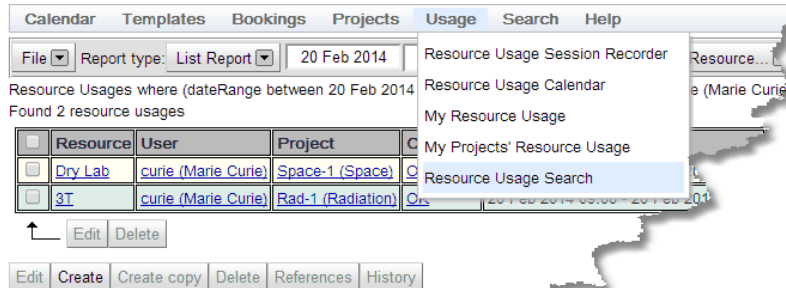
Because the *project* parameters is set specifically to **one of my projects** there is no **Project** button available but the other buttons can be used to change the parameters of the search.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.6.3.4 Resource Usage Search

The **Resource Usage Search** page does a search through the [resource usage](#)⁶⁵⁵ without any restrictions (apart from whatever [Permissions](#)⁶⁵⁴ have been configured).

By default, the **Resource Usage Search** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **Resource Usage Search** page looks like once a search has been done:

Resource	User	Project	Outcome	Date Range	Created	Creator	Modified	Modifier
Dry Lab	curie (Marie Curie)	Space-1 (Space)	OK	20 Feb 2014 11:00 - 20 Feb 2014 13:00	20 Feb 2014 13:10	curie (Marie Curie)		
3T	curie (Marie Curie)	Rad-1 (Radiation)	OK	20 Feb 2014 09:00 - 20 Feb 2014 11:00	20 Feb 2014 13:11	curie (Marie Curie)		
3T	einstein (Albert Einstein)	Rad-1 (Radiation)	OK	19 Feb 2014 11:30 - 19 Feb 2014 14:00	20 Feb 2014 13:21	admin2 (admin2)		
Electron Microscope	darwin (Charles Darwin)	Gene-1 (Genes)	Operator error	19 Feb 2014 14:00 - 19 Feb 2014 16:30	20 Feb 2014 13:21	admin2 (admin2)		

The **Resource Usage Search** is run with the following default parameters:

Resource Usages where (dateRange between **start_date** and **end_date**)

All the possible buttons are available to change parameters of the search.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.7 Services

While **Calpendo** is primarily about [bookings](#)⁶⁵², it can also be used to book services. A **Service** is additional functionality that a user can buy as part of their [project](#)⁶⁵⁴ work. These can be such things as blood tests, checking scans, Chromosome painting etc.

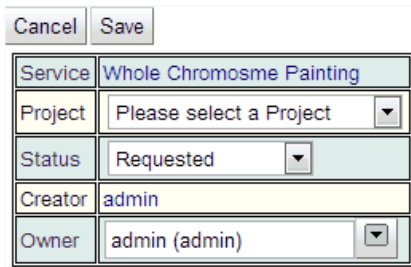
The administrator will have configured which [services](#)⁶⁵⁵ are available for a user to order, created the order forms and set up the pricing structure for the cost of those *services*.

3.7.1 Available Services

This page lists all the [services](#)⁶⁵⁵ that are available to be purchased at the facility. These can be filtered by location and provider. To order a *service* click on the **Order** link and the order process will start.

Location <input checked="" type="checkbox"/> No location <input checked="" type="checkbox"/> Bergen <input checked="" type="checkbox"/> Tromsø <input checked="" type="checkbox"/> Trondheim	Spectral Karotyping (SKY) Cytogenetics Trondheim Produce a colour image of chromosomes.
Provider <input checked="" type="checkbox"/> No service provider <input checked="" type="checkbox"/> Cytogenetics <input checked="" type="checkbox"/> Radiology	Whole Chromosome Painting Cytogenetics Trondheim Chromosome painting allows highly sensitive and specific visualization of individual chromosomes in metaphase or interphase cells and the identification of both numerical and structural chromosomal aberrations in human pathology.
	Fluorescence in situ Hybridization Cytogenetics Trondheim <p>First, a probe is constructed. The probe must be large enough to hybridize specifically with its target but not so large as to impede the hybridization process. The probe is tagged directly with fluorophores, with targets for antibodies or with biotin. Tagging can be done in various ways, such as nick translation, or PCR using tagged nucleotides.</p> <p>Then, an interphase or metaphase chromosome preparation is produced. The chromosomes are firmly attached to a substrate, usually glass. Repetitive DNA sequences must be blocked by adding short fragments of DNA to the sample. The probe is then applied to the chromosome DNA and incubated for approximately 12 hours while hybridizing. Several wash steps remove all unhybridized or partially hybridized probes. The results are then visualized and quantified using a microscope that is capable of exciting the dye and recording images.</p> <p>If the fluorescent signal is weak, amplification of the signal may be necessary in order to exceed the detection threshold of the microscope. Fluorescent signal strength depends on many factors such as probe labelling efficiency, the type of probe, and the type of dye. Fluorescently tagged antibodies or streptavidin are bound to the dye molecule. These secondary components are selected so that they have a strong signal.</p> <p>FISH experiments designed to detect or localize gene expression within cells and tissues rely on the use of a reporter gene, such as one expressing green fluorescent protein, to provide the fluorescence signal</p>
	Radiological Assessment Radiology Bergen Radiologist to look over <i>scan</i> for abnormalities.

Once the order option is clicked the appropriate [Service Order](#)⁶⁵⁵ form will appear.

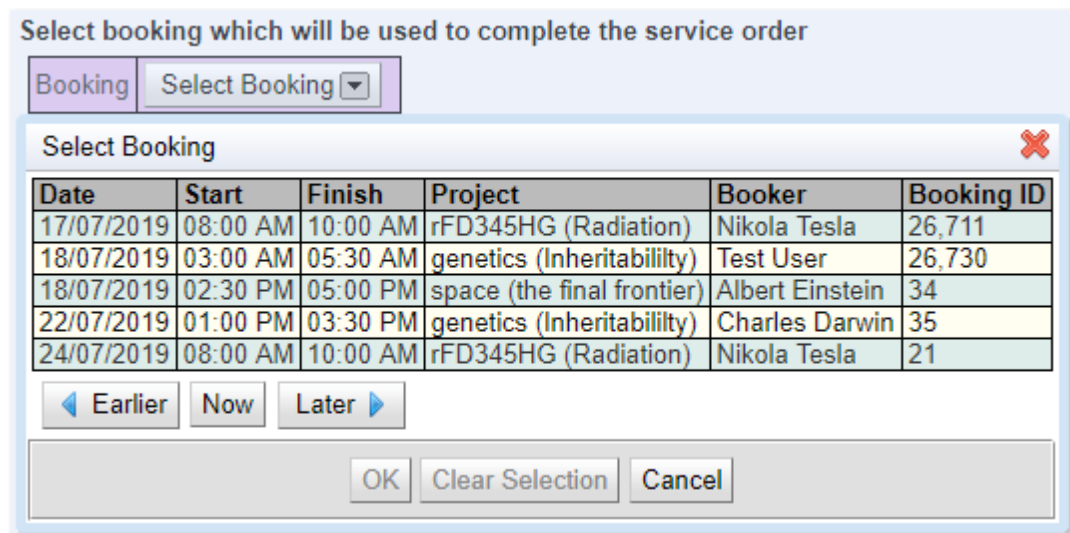


A screenshot of the 'Service Order' form. It has a 'Cancel' button and a 'Save' button at the top. The form contains several fields: 'Service' with the value 'Whole Chromosome Painting', 'Project' with a dropdown menu showing 'Please select a Project', 'Status' with a dropdown menu showing 'Requested', 'Creator' with the value 'admin', and 'Owner' with a dropdown menu showing 'admin (admin)'.

Fill in the order form and **Save** for the *service order* to be created. Above is the basic order form this can be changed by the administrator by adding new [properties](#)⁶⁵⁵ either to the *Service Order Bisket Type*⁶⁵² or creating a new *Biskit Type* that derives from *Service Order* and includes any required additional *properties*.

Linking Services to Bookings

If the Service Order is configured to be linked with bookings, when entering the service order, the user will see a button labelled "Select Booking" which lets them choose a booking for the appropriate resource.



A screenshot of the 'Select booking which will be used to complete the service order' dialog. It has a 'Booking' label and a 'Select Booking' dropdown button. Below this is a table with the following data:

Date	Start	Finish	Project	Booker	Booking ID
17/07/2019	08:00 AM	10:00 AM	rFD345HG (Radiation)	Nikola Tesla	26,711
18/07/2019	03:00 AM	05:30 AM	genetics (Inheritabililty)	Test User	26,730
18/07/2019	02:30 PM	05:00 PM	space (the final frontier)	Albert Einstein	34
22/07/2019	01:00 PM	03:30 PM	genetics (Inheritabililty)	Charles Darwin	35
24/07/2019	08:00 AM	10:00 AM	rFD345HG (Radiation)	Nikola Tesla	21

Below the table are buttons for 'Earlier', 'Now', and 'Later'. At the bottom are buttons for 'OK', 'Clear Selection', and 'Cancel'.

If the service order has a project, then the user will only be offered bookings that are set up to use that project.

When an order is updated to reference a booking, then a check is made to see if the booking has a reference to a Service Order. If so, then the booking will be updated to reference the order it relates to.

3.7.2 Searching For Orders

Calpendo has multiple ways of finding out information about the [service orders](#)⁶⁵⁵ that have been created..

3.7.2.1 Using The Order Searches

All the order search pages work in a similar way. This chapter explains the basic mechanisms for all the order searches. To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

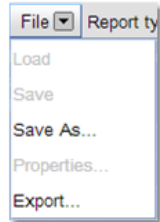
When entered any of the order search pages you will automatically start a search of the appropriate type. This is what a search will look like when it is completed.

Found one service order

<input type="checkbox"/>	Creator	Owner	Service	Status	Project	Created	Modified	Cancelled
<input type="checkbox"/>	leonardo (Leonardo Da Vinci)	leonardo (Leonardo Da Vinci)	Spectral Karyotyping (SKY)	Requested	inventions (inventions)	6 Sep 2013 11:41		

[↑](#) [Edit](#) [Delete](#)

The following section goes through the options available after the search has completed, enabling the search parameters to be changed, saved, or the data exported, as well as change the information viewed by the search. Not all options will be available for every order search type, but that will be outlined in the specific search types chapter. For more information on any of the features shown read the [Search](#)¹¹⁸ chapter.



Save As: Saves the report for later use.

Export: Export the information found in the search to a file.

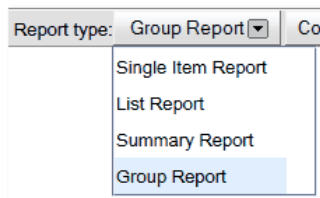
This button determines what type of report will be seen:

[Single Item Report](#)⁶⁵⁵: Displays a single record that matches the search. Does not appear for innumerable objects such as orders.

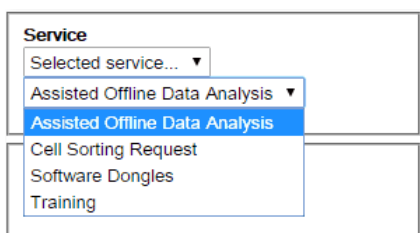
[List Report](#)⁶⁵³: Lists each record found that matches the search one under the other.

[Summary Report](#)⁶⁵⁶: Allows a summary of the information found.

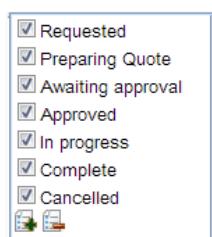
[Group Report](#)⁶⁵³: Allows the report to be shown by groupings with a count of the number of records that fit each group.



To change the search parameters

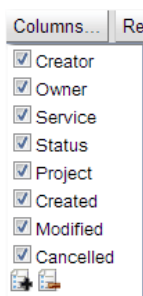
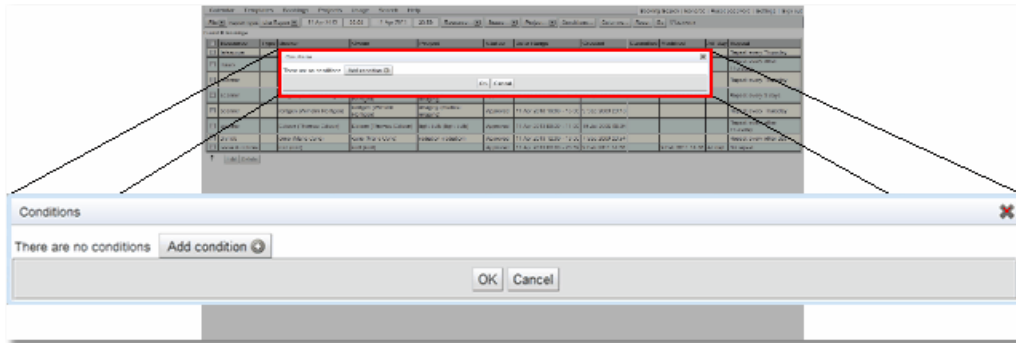


There are a number of options that can be used to filter the services. They all provide a drop down menu so that a choice between any, none or a particular item can be made. The filters are: **Owner, Creator, Service, Provider, Location.**



The final filter is **Status**. This filter allows you to use tick boxes to choose which status('s) to filter on.

Additional conditions for the search may also be set up. For more information on how to set up conditions read the information in the [Search](#) ¹¹⁸ page.



Finally change the columns displayed in the output.



Reset: Resets the search conditions to the defaults set up on starting the search.

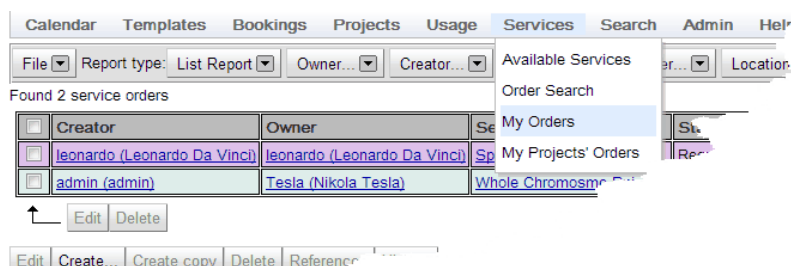
Go: Runs the search with the current conditions

Autorun: If this button is ticked then searches will run as soon as any conditions change. If a number of the conditions of the search will be changed then it is more efficient if this is not ticked as each change will start a search and the appropriate transfer of data from the server.

3.7.2.2 My Orders

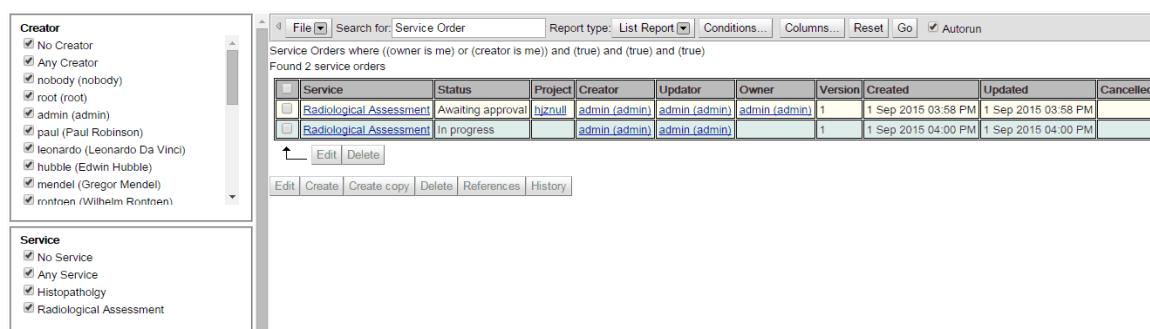
The **My Orders** search page does a search for orders that the user has created.

By default, the **My Orders** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **My Orders** page looks like once a search has been done:



The my orders search is run with the following default parameters:

Service Orders where ((**owner is me**) or (**creator is me**))

Because the *owner* parameter is set to be **me** there is no **Owner** filter available, but the other filters found down the left of the page can be used to change the parameters of the search.

Once an order has been found, click on it for the full details to be shown. Once an order has been created if the administrator implemented **Notes** both the users dealing with the order and the initiator of the order can add **Notes**. To add a note, write in the box marked **Add Note To This Order** and then press the button. There is no need to be in edit mode for this to work.

Edit	Create	Create copy	Delete	References	History
------	--------	-------------	--------	------------	---------

Service	Radiological Assessment
Status	In progress
Project	
Sex	No sex specified

Add Note To This Radiological Assessment Service Order

Add Note

Service Details

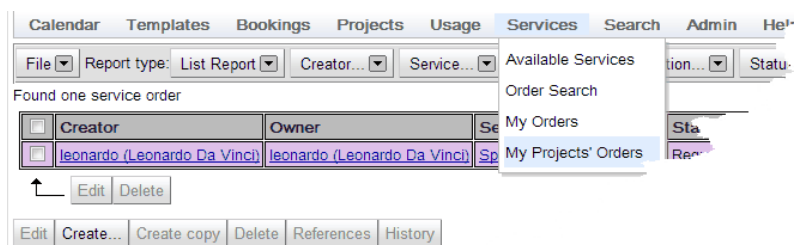
Sequences	No sequence selected
Other Sequence Type	
Age	
Subject Type	No subject type selected
Image Source	No image source specified
Pacs ID	

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.7.2.3 My Projects' Orders

The **My Projects' Orders** search page does a search for orders that are associated with [projects](#)⁶⁵⁴ that the user is also associated with.

By default, the *My Projects' Orders* page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **My Projects' Orders** page looks like once a search has been done:

The screenshot shows the 'My Projects' Orders' search results page. The page includes filters for Creator, Service, Provider, and Location. The search results table shows 4 service orders.

Service	Status	Project	Creator	Updater	Owner	Version	Created	Updated	Cancelled
Radiological Assessment	Awaiting approval	radiation19 Aug 2015	curie (Marie Curie)	curie (Marie Curie)	Einstein (Albert Einstein)	1	5 Aug 2015 01:21 PM	5 Aug 2015 01:21 PM	
Radiological Assessment	Requested	radiation19 Aug 2015	curie (Marie Curie)	curie (Marie Curie)	curie (Marie Curie)	1	5 Aug 2015 01:23 PM	5 Aug 2015 01:23 PM	
Histopathology	Requested	radiation19 Aug 2015	curie (Marie Curie)	admin (admin)	curie (Marie Curie)	1	5 Aug 2015 01:25 PM	16 Oct 2015 10:41 AM	
Radiological Assessment	Requested	radiation19 Aug 2015	curie (Marie Curie)	curie (Marie Curie)	curie (Marie Curie)	1	5 Aug 2015 01:44 PM	5 Aug 2015 01:44 PM	

The my orders search is run with the following default parameters:

Service Orders where (*project* is **one of my projects**) and (*creator* is **anything**) and (*service* is **anything**) and (*service.provider* is **anything**) and (*service.location* is **anything**)

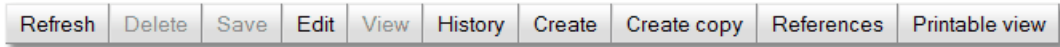
Because the *project* option is **one of my projects** there is no **Owner** filter available but the other filters down the left of the page can be used to change the parameters of the search.

See the section on [My Orders](#)¹⁰¹ to learn how to add comments to an order.

To learn more about the advanced features available with searches and how to select and edit data returned from a search then go to the [Search](#)¹¹⁸ chapter.

3.8 Toolbar Button Standard Definitions

Calpendo uses toolbars in a number of places to provide options for editing and displaying information. This chapter will define the standard button operations.



When a toolbar is being described, the text will refer to this chapter, and describe either those buttons not mentioned here or whose operation is significantly different from the norm.

Editing Buttons

Button	Description
Edit	The Edit button is greyed out if there is no item selected, or the user does not have permission ⁶⁵⁴ to edit the item. Pressing the button puts the item into Edit mode. There will be a warning if another user is already editing the <i>Biskit</i> , and the <i>Biskit</i> has the Version Property meta-property ⁶⁵⁴ set up.
Apply	The Apply button is active only while an item is in edit mode. Press it to save the item, and stay in edit mode.
Save/OK	The Save/OK button is active only while an item is in edit mode. Press it to save the item, and the item will revert back to a read-only view.
Cancel	The Cancel button is active only while an item is in edit mode. Press it to throw away any changes since the last Apply (if there is an Apply option on the menu) and move out of edit mode.
View	The View button is used to revert back to a view of the item's details, in read-only mode. If there is no item selected, or the item's details are already shown in read-only mode, then the button will be greyed out.
Delete	The Delete button is greyed out if there is no item selected, the selected item cannot be deleted due to <i>Permissions</i> , or it is being referenced ⁶⁵⁵ by something else. If there are many things in the database that reference the item, then it may be difficult to delete it, because the <i>references</i> must be removed first. Press to delete the selected items from the database.
Create	The Create button will create a new item of the selected type, but it will be greyed out if the user does not have <i>permission</i> to create items of that type. Once pressed, the item will go into edit mode. The item will not exist in the database until either Save or Apply are pressed.
Create Copy	The Create Copy button will create a copy of the selected type, but it will be greyed out if the user does not have <i>permission</i> to create items of that type. Once pressed, the item will go into edit mode. The item will not exist in the database until either Save or Apply are pressed.
Cut	The Cut button cuts the currently selected item, removing it from its current position and puts it in the edit buffer for later use.
Copy	The Copy button takes a copy of the currently selected item and puts it in the edit buffer for later use.
Paste	The Paste button pastes the current paste buffer contents to the currently selected position.
Full Screen	The Full Screen button shows a Detail view without seeing the List view. Shift-click on a list will do the same for the selected row.

Viewing Buttons

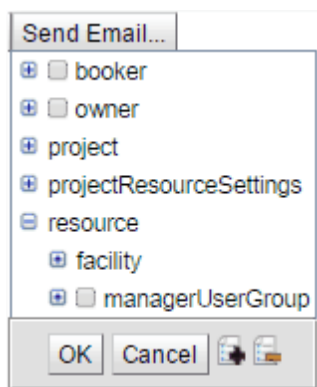
Button	Description
Refresh	The Refresh button reloads the current page from the database.
Open All	The Open All button opens up the whole tree so that all the information can be seen.
Close All	The Close All button closes the tree information so only the top of each tree can be seen.
Printable View	The Printable View button will reformat the page to make it printer friendly, including removing the list of items and retaining only the selected item. Then use the browser's Print option to print the item's details. Click anywhere on the page to revert back to the normal view.
Columns..	The Columns... button allows the user to decide which properties ⁶⁵⁵ will be displayed. The default is always the Name property .

The Send Email Button

This button will allow selection from the *Biskit* list the paths of the people to be emailed.

<input type="checkbox"/>	Resource	Type	Booker	Owner	Project
<input checked="" type="checkbox"/>	MEG	Pig	admin (admin)	rontgen (Wilhelm Rontgen)	imaging (Medical imaging)
<input checked="" type="checkbox"/>	MEG	Pig	admin (admin)	rontgen (Wilhelm Rontgen)	imaging (Medical imaging)

Click on the **Send Email** button to get a drop down menu from where the user can define who the **Email** is going to go to. Once all the appropriate paths are selected click **OK**.



This will take the user to the [Send Email](#)¹⁸⁶ page where the user can input the **Subject** and **Message** for the **Email** and check to see who the **Email** will be sent to.

The References Button

This button will show a list of everything that *references* the currently selected item. It will be greyed out if there is nothing referring to the item. Once pressed a list of the *references* will be viewed:

Refresh	Delete	Save	Edit	View	History	Create	Create copy	References	Printable view
---------	--------	------	------	------	---------	--------	-------------	------------	----------------

References to User 36

Type of Referrer	ID of Referrer	Referrer
UserGroup	1	20th Century
Project	8	light bulb (light bulb)
	3	radiation (radiation)

This shows a table with the [Biskit Type](#)⁶⁵² of the referrer, the unique database identifier of the referrer and its name. Either press the **References** button again, or press the **Back** button on the browser, to return to the original view.

The History Button

This will show a change [history](#)⁶⁵³ of the selected item, like this:

Refresh	Delete	Save	Edit	View	History	Create	Create copy	References	Printable view
---------	--------	------	------	------	---------	--------	-------------	------------	----------------

History of User 6

Log date	Editor	Change Type	IP Address	Login name	Roles	Given name	Other name	Family name	Email address	User Type	Status
14 Jul 2009 20:46	0	CREATE	192.168.0.198	leonardo	Nothing selected	Leonardo	De	Vinci	blah@blah.com		Normal
14 Jul 2009 21:10	clare	UPDATE	192.168.0.198	leonardo	Nothing selected	Leonardo		Da Vinci	blah@blah.com		Normal
9 Sep 2009 20:34	clare	UPDATE	192.168.0.198	leonardo	Nothing selected	Leonardo		Da Vinci	blah@blah.com	Physics	Normal
13 Jul 2010 11:56	admin	UPDATE	192.168.0.198	leonardo	User	Leonardo		Da Vinci	blah@blah.com	Physics	Normal
13 Jul 2010 13:26	admin	UPDATE	192.168.0.198	leonardo	User	Leonardo		Da Vinci	blah@blah.com	Physics	Normal
18 Aug 2010 21:22	admin	UPDATE	192.168.0.198	leonardo	User	Leonardo		Da Vinci	blah@blah.com	Physics	Password must be reset at next login
18 Aug 2010 21:41	admin	UPDATE	192.168.0.198	leonardo	User	Leonardo		Da Vinci	blah@blah.com	Physics	Password must be reset at next login

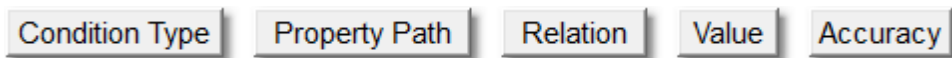
This shows who has changed the item and when, and also shows some of the item's [properties](#)⁶⁵⁵ as they changed over time. The "complex" *properties* are not shown. **Calpendo** does not record the history of *properties* that store lists of items, like lists of [projects](#)⁶⁵⁴ and lists of [user groups](#)⁶⁵⁶. The **History** button will be greyed out if there is no item selected or if the item's *history* is already being shown. To revert back to showing the item's details, press the **View** button or the **History** button.

3.9 Conditions

[Conditions](#)⁶⁵³ are used in several places within **Calpendo**, most notably [Searching](#)¹¹⁸, [WorkFlows](#)³³⁴ and [Permissions](#)³¹⁴ and [Booking Rules](#)²³⁵. Understanding *Conditions* is essential in order to be able to configure **Calpendo** properly because of their pervasive use. Their purpose is to allow you to specify what situations activate a [Workflow](#)³³⁴, or to which situations a [Permission](#)³¹⁴ or [Booking Rule](#)²³⁵ applies. This chapter will show how to use *conditions* for the more complex tasks required in [WorkFlows](#)³³⁴, and [Permissions](#)³¹⁴ and [Booking Rules](#)²³⁵.

The Anatomy Of A Condition

A [Condition](#)⁶⁵³ extracts one or two [properties](#)⁶⁵⁵ from the context and checks whether a [relation](#)⁶⁵⁵ holds true. If it does, the *Condition* is said to match. A *Condition* consists of the following items, in this order:



The **Condition Type** indicates the source of a *property*.

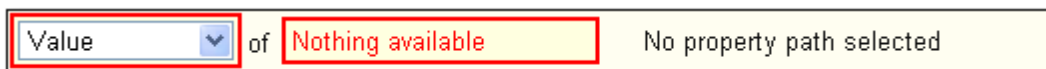
The [Property Path](#)⁶⁵⁵ tells how to reach a *property* from that source.

The *Relation* provides an operator to compare the value of that *property* to another value.

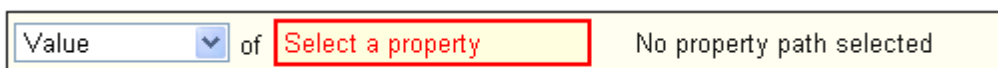
The **Value** provides a value to do the comparison against, this may be another *property*.

The **Accuracy** specifies how accurate the comparison should be (used with dates).

The **Condition Type** and *Property Path* are always present, but the others may not be. For example, if the user hasn't selected a *Property Path*, then the *Relation*, **Value** and **Accuracy** do not show. If the context does not specify a [Biskit Type](#)⁶⁵², then when a *Condition* is first created, it will look like this:



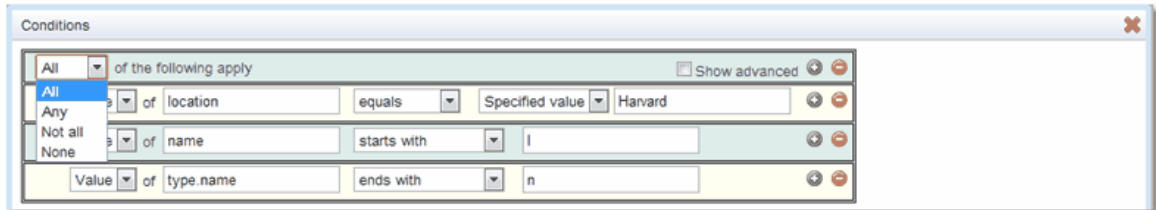
The above picture shows the default **Condition Type** of **Value** with a red border because **Value** cannot be used without a *Biskit Type* being available in the context. The *Property Path* also has a red border and shows there are no properties available for the same reason. Choosing a *Biskit Type* of anything other than **Any data type** changes the *Property Path* selector to show there are now *properties* that can be selected: :



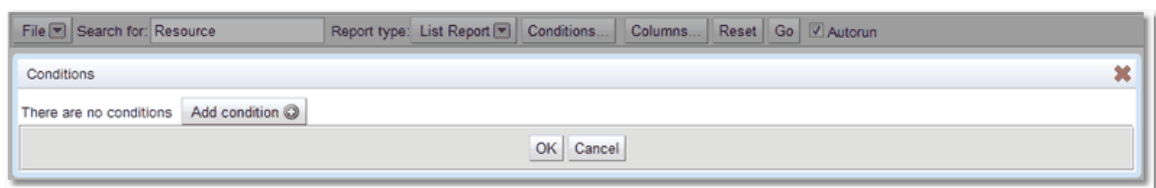
The *Property Path* selector still shows in red because a *Property Path* is required, but hasn't been selected yet. Clicking where it says **Select a property** would now generate a drop-down menu with a list of all the *properties* that exist on the [Biskit Type](#)⁶⁵² selected in the current context.

The Conditions Panel

Everywhere that *conditions* are used, they appear in a panel that looks something like this in a writeable context:



When there are no *conditions*, it looks like this in a writeable context:



When editing *Conditions*, add a new *Condition* by pressing the **Add Condition** button. At any time clicking on the **green +** button will add a new *condition* after the current one, clicking the **red -** button will delete the *condition* the button is associated with.

Whenever there are multiple *Conditions*, they will appear in the **Conditions Panel**, each in their own row.

The Conditions Context

A **Conditions Panel** is subject to the context within which it is used. There are two parts to the context that affect the way the **Conditions Panel** behaves: the [Bisikit Type](#)⁶⁵² the *condition* is being placed upon, and whether used for an update so that there are two instances of the data available (the **Old Value** and the **New Value**) or whether there is just one instance available (the **Value**).

For a [Permission](#)³¹⁴, it's a very similar situation. The context *Bisikit Type* is the *Bisikit Type* that the [Permission](#)⁶⁵⁴ controls access to, and if the *Permission* applies to any *Bisikit Type*, then the context has no *Bisikit Type*. The **New Value** is available when the action being controlled is **Update** and not when the action is anything else.

For a [Search](#)¹¹⁸ the context *Bisikit Type* is the *Bisikit Type* that the **Search** is on, and the only **Condition Type** available is **Value**.

For a WorkFlow the context contains the information from all the previous [triggers](#)⁶⁵⁶.

For [Booking Rules](#)²³⁵, the context *Bisikit Type* is [Booking](#)⁶⁵². The **New Value** is available when the action being controlled is **Update** and not when the action is anything else.

The Conditions Type

When a *condition* is created or edited, the type has a number of possible values:

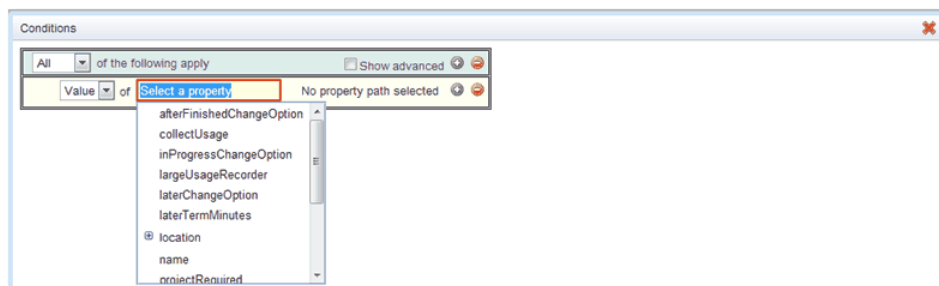
Type
Value
Old Value
New Value
Meta-Property
Source#

Condition Type: Value

A **Condition Type** of **Value** means the *condition* will be placed upon the value of a [property](#)⁶⁵⁵ where there is no probability of change. For a [Permission](#)³¹⁴ this means the value of the *property* on an object somebody is trying to **Create** or **Delete**.

A **Condition Type** of **Value** is only suitable when the *condition* context specifies a *biskit type*.

When a **Condition Type** of **Value** is specified, the [property path](#)⁶⁵⁵ will show all the *properties* of the type defined by the context's *biskit type*, and any *properties* that can be reached from those *properties*. When the context *biskit type* is set to a **Resource**, clicking where it says **Select a property** shows this:



A [Resource](#)⁶⁵⁵ has a number of *properties*, as shown. One of them, **Location**, is itself an object that has *properties* of its own that can be selected. As soon as you select a *property* from the drop-down, the [Relation](#)⁶⁵⁵ part of the *condition* will change from **No property path selected** to something appropriate for the *property* selected (described further below).

Condition Type: Old Value

A **Condition Type** of **Old Value** means the *condition* will be placed upon the value of a [property](#)⁶⁵⁵ before any change has been made to it. For a [Permission](#)³¹⁴ this means the value of the *property* on an object somebody is trying to **Update**.

The *property path* will display the same as for a **Condition Type** of **Value**.

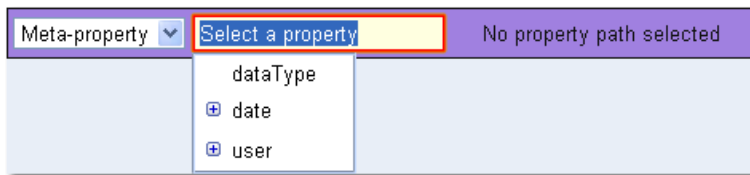
Condition Type: New Value

A **Condition Type** of **New Value** is only available when a *bisikit* type has been specified, and even then only in some contexts. For a [Permission](#)³¹⁴ or [Booking Rule](#)²³⁵, a **Condition Type** of **New Value** is only available for the **Update** action. Selecting **New Value** means that the *condition* will be placed on the value of the *property* after the update has occurred.

The *property path* will display the same as for a **Condition Type** of **Value**.

Condition Type: Meta-Property

A **Condition Type** of **Meta-Property** is available in all contexts. When this type is selected, the *property path* displays the following:



A [Meta-Property](#)⁶⁵⁴ lets you place restrictions on:

- the type of the data that was **Created**, **Updated** or **Deleted**
- the date and time that the action took place
- the user that caused the action

This is called a *Meta-Property* because it's not a *property* of whatever was created, updated or deleted, but rather, it's a *property* of the action itself.

Condition Type: Source#

A **Condition Type** of **Source#** is only available in [Workflows](#)³³⁴ and it shows all the *properties* available on the *Bisikit* provided as output to the *trigger* with the **Source#**. If the action is not **Update** then the **New** and **Old** *properties* will hold identical data.

The Property Path

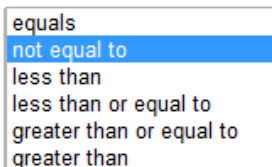
As described above, the *properties* shown in the *property path* depend upon the **Condition Type** selected. However, each *property* that is shown has its own way of being displayed, regardless of how it came to be there. For example, any date/time *property* that is shown will display with sub-properties.



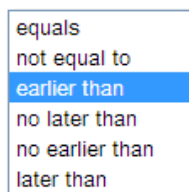
A [Resource](#)⁶⁵⁵ has a number of properties, as shown. One of them, **Location**, is itself an object that has *properties* of its own that can be selected. The *Relation*, **Value** and **Accuracy** part of the *condition* will not show unless a *property path* has been selected. Once a *property* is selected, the extra parts of the *condition* appear, and they reflect the type of the *property* selected and the choice of **Condition Type**.

The Relation

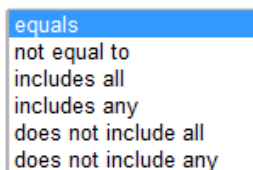
The *Relation* is only shown when there's a *bisikit* type in the context and a *property path* has been selected. The options displayed in the *Relation* drop-down menu differ depending on the type of *property* selected by the *property path*.



This is the standard choice of relations for an integer or floating point number property.

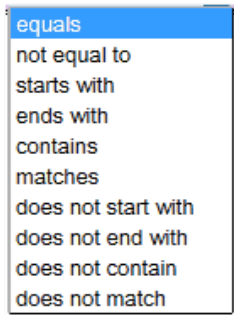


A date *property* displays with semantically identical options, but using different words to match the English you would expect.



Not all integer *properties* display using the above options. That's because some integer *properties* are defined as a bit set of values of named integers (see [Bit Sets](#)⁶⁵⁴ for details). A good example of this is a [user's roles](#)⁶⁵⁶. There are up to 32 roles defined, and a user may be deemed to have any number of these roles. For such *properties* it doesn't make sense to compare two sets of roles using greater than or less than, and so the available *relations* are changed to reflect this.

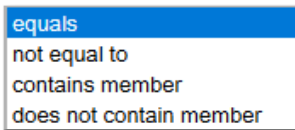
A text *property* displays with options for comparison with the starting, ending or interior characters. Also available is pattern matching using the following:



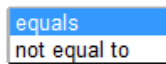
^	The beginning of the line.
\$	The end of the line.
.	Any character.
*	Zero or more times.
+	One or more times.
?	Once or not at all.
X Y	Either X or Y.
{n}	Exactly <i>n</i> times.
{n,m}	At least <i>n</i> times but not more than <i>m</i> times.
[a-z]	Characters a to z.
[^a-z]	Any character other than a to z

[Search](#)¹¹⁸ conditions are not case sensitive, while non-search ([WorkFlows](#)³³⁴, [Permission](#)³¹⁴ or [Booking Rule](#)²³⁵) conditions are case sensitive.

Depending on whether the pattern matching is going to be done in Java ([WorkFlows](#)³³⁴, [Permission](#)³¹⁴ or [Booking Rule](#)²³⁵) or SQL ([Search](#)¹¹⁸) will depend on which additional matching features are available. It is recommended that users only use the options listed above.



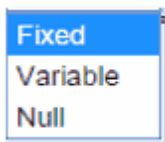
If the *property* is of type [Biskit](#)⁶⁵² and is a collection of any *Biskit* contains relations are available.



All the remaining *property types* don't support any notion of an ordering that could be used to specify a **less than relation** etc. So for these, the relations available are: equal, not equal to.

The Value

A drop-down menu is shown giving the choice for the **Value**. For any *property* that is not a date or date-time, it shows this:



This gives a choice between a value that is **Null** (a special value meaning there is no value) and **Fixed** or **Variable**.

If **Fixed** is chosen then a further item will appear so that the value can be chosen. The item shown for the value entry will be appropriate to the type of the *property* indicated by the *Property Path*.

The screenshot shows a 'Conditions' dialog box. At the top, it says 'All of the following apply' with a 'Show advanced' checkbox. Below this, there is a row of fields: 'Value' (dropdown), 'of' (text), 'status' (text), 'equals' (dropdown), 'Fixed' (dropdown), and a dropdown menu that is currently open. The dropdown menu lists 'Please select a Status', 'Requested', 'Approved' (highlighted), 'Denied', 'Cancelled', and 'Tentative'. There are also '+' and '-' buttons next to the 'Fixed' dropdown.

If **Variable** is chosen then the user can choose another [Condition Type](#)¹⁰⁹ and repeats the process selecting either a *property* or *meta-property* to be compared to the original *property* or *meta-property*.

The screenshot shows the 'Conditions' dialog box with a different configuration. It says 'All of the following apply' with a 'Show advanced' checkbox. Below this, there is a row of fields: 'Value' (dropdown), 'of' (text), 'booker' (text), 'equals' (dropdown), 'Variable' (dropdown), 'Value' (dropdown), and 'project.owner' (text). There are also '+' and '-' buttons next to the 'Variable' and 'Value' dropdowns.

Date Properties And Accuracy

When dealing with dates there are some other options that are available. Whenever a *property* is asked for which stores a date two *condition* lines will appear, one for the start date and one for the finish date, the *relation* can then be set (see table earlier in the chapter for the full list available) :

Then set how accurate the date checking needs to be. There are four options against which the comparison can be made and then a level of accuracy to be used.

Date Accuracy	Description
now	The time specified but with a level of accuracy.
now plus	Within now plus a certain amount of time with a level of accuracy.
now minus	Within now minus a certain amount of time with a level of accuracy.
specified date	The specified date exactly. (No level of accuracy may be set.

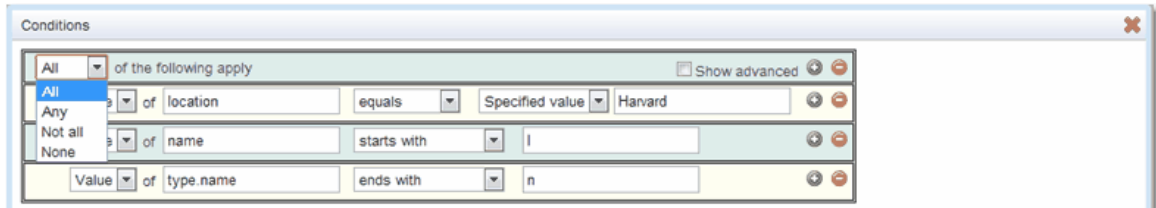
The **now** option means that the value of the *property* to be compared against is to be the date and time that the comparison is done. The **now plus** and **now minus** options indicate that the date to be calculated is relative to the time the comparison is done. When selecting either of these options, a further item is displayed to input how much time to add or subtract from the time of the comparison to reach the date-time that the *property* is being compared against. If the **specified date** option is selected, then a date-time entry is displayed to input the exact point in time to compare the *property* against.

The **Accuracy** is only shown for date or date-time properties, and it specifies how accurately the comparison is to be. For example now plus 1 month to the month, if run in December will find everything in February.

Then, when the *Condition* is checked, and **Calpendo** compares two dates, it first rounds both dates down to the year, month, day, hour, minute or second. After rounding both dates, it then compares the dates using the specified *relation*.

Combination Option

Once all the *conditions* have been created, choose how logically the *conditions* will work together (the combination option). Click on the drop down button in the top left corner of the *condition* box.



Combination Option	Description (For the combination to be TRUE)
All	All the individual <i>conditions</i> must be true. (Logical AND)
Any	At least one of the individual <i>conditions</i> must be true. (Logical OR)
Not All	As long as not all the individual <i>conditions</i> are true, this includes none of the <i>conditions</i> being true. (Logical NAND)
None	As long as none of the individual <i>conditions</i> are true. (Logical NOR)

This creates the following search *condition*:

Resources where (**location.name** starts with **h**) and (**type.name** equals **Room**) and (**requireCancellationReason** equals **True**)

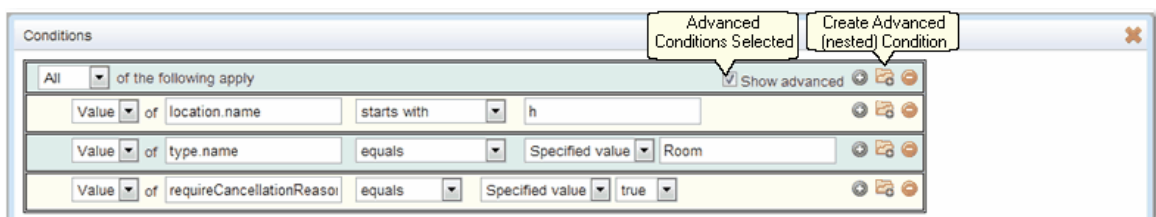
At this point use **OK** to save the *condition* and **Cancel** to start again.

Nested Conditions

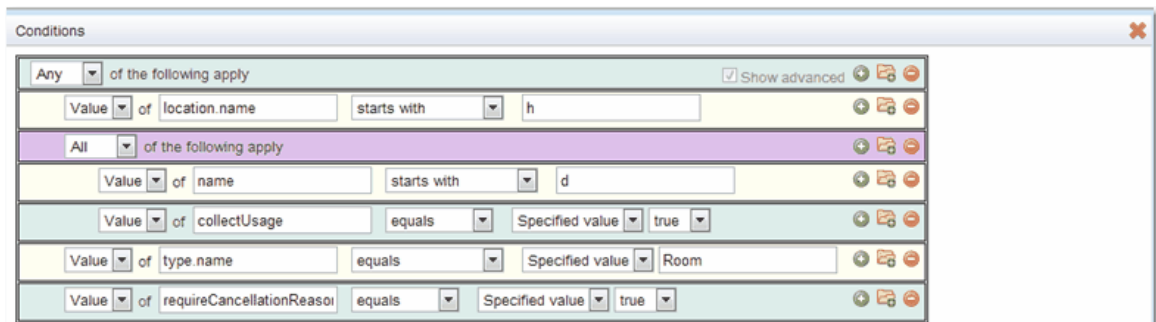
Nested *conditions* allow as many levels of *conditions* as the user requires. Each set of nested *conditions* also has its own combination option allowing the user to create as complex an overall *condition* as possible.

Each nested set of *conditions* is evaluated separately, and the overall result of the nested *condition* is then used as an individual *condition* result as part of the level above's *condition* calculation.

In order to use advanced *conditions* tick the **Show Advanced** box. Once ticked the **Create Advanced Condition** icon appears at the top of the *condition* list and on each of the current *conditions*.



Below the *conditions* have been expanded using a nested *condition*. The nested *condition* has two parts, **name** and **collectUsage** and it will be **True** if both sub *conditions* are **True**.



This creates the following search *condition*:

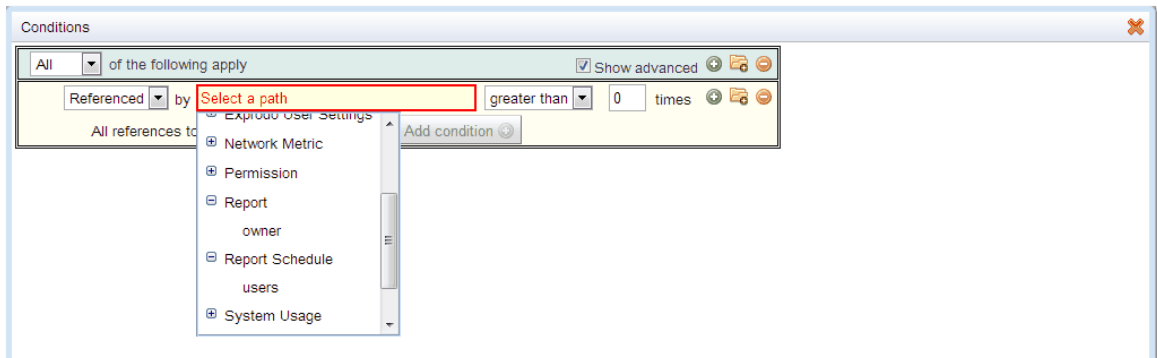
Resources where (**location.name** starts with **h**) or ((**name** starts with **d**) and (**collectUsage** equals **true**)) or (**type.name** equals **Room**) or (**requireCancellationReason** equals **true**)

In order to test the *condition* the following steps will be used

1. Check the nested *condition*, if both *conditions* are **True** then the nested *condition* is **True** otherwise **False** (**All conditions** apply)
2. Check the top level *condition*, if any of the three top level *conditions* are **True** or the nested *condition* is **True** then the whole condition is **True** (**Any condition** applies)

Referenced

Ticking **Show advanced** allows the use of **Referenced** as a **Condition Type**. Using **Referenced** it is possible to search for [BiskitDefs](#)⁶⁵² that are referenced by other *BiskitDefs*. Using the **Relation Statement** and the **Value** this can be further refined to find those that are referenced a number of times. e.g. more than once, less than three times etc.



When using **Referenced** the application looks for all those *BiskitDefs* that have a *property* which points to the *BiskitDef* that is the object of the search. In this case the search *BiskitDef* is **Exprodo User** the application will give a drop down that lists all the *BiskitDefs* that have a *property* that points to an **Exprodo User** *BiskitDef*. Expanding a *BiskitDef* will list the *properties* that are available. For instance **Report** has an **owner** *property* and **Report Schedule** has a **users** *property*.

As an example to find all those users that have created **Reports**, set **Exprodo User** as the *BiskitDef* to be searched for and **Report.owner** (**owner** is a pointer to an **Exprodo User**) as the *property* defined in the **Property Path**, returned will be a list of **Exprodo Users** that currently have **Reports** in the **Report Manager**. If the **Relation Statement** and **Value** are set to **greater than 1 times**, then only those users with two or more reports will be returned.

See Also:

- The examples in [Permissions](#)³¹⁴

3.10 Searching For Information

There are various ways to search for information in **Calpendo**, and several ways to display the information found. This section describes the generic search options that apply to any type of data.

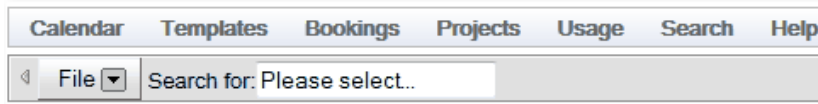
See Also

- [Searching For Bookings](#)⁶⁶ for how to search for information about [bookings](#)⁶⁵².
- [Searching For Projects](#)⁷⁸ for how to search for information about [projects](#)⁶⁵⁴.
- [Search For Usage](#)⁹⁰ for how to search for information about [resource usage](#)⁶⁵⁵.

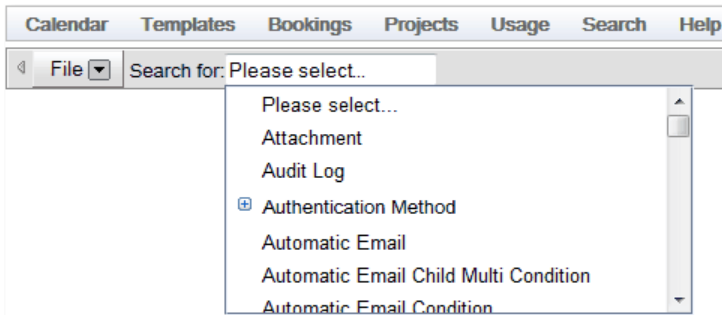
3.10.1 Search

The **Search** page allows searches for any type of data within **Calpendo**. By default, it appears under the **Search** menu, with a sub menu item also labelled **Search**. However, the menu may have been configured so that the **Search** page is not visible or in a different place.

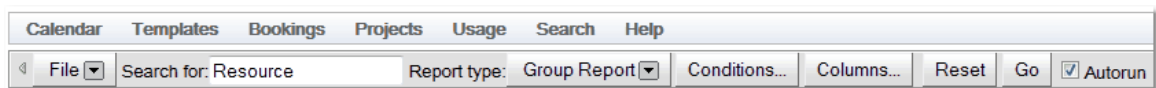
When you enter the **Search** page the following will be seen:



Choose which [Biskit Type](#)⁶⁵² to search for. Clicking on **Please Select** will provide a drop down menu with the complete list of types available to search.



Once a *Biskit Type* is selected additional options will appear.



There are four report types: [List](#)⁶⁵³, [Summary](#)⁶⁵⁶, [Single Item](#)⁶⁵⁵ and [Group](#)⁶⁵³. The user can also add [conditions](#)⁶⁵³ to the search (not *Single Item*), change the columns that are going to be viewed in the returned search (not *Summary* or *Single Item*), or reset the search *conditions* to the default (not *Single Item*).



This icon will toggle whether the **File** and **Report Type** options are available.

The **Go** button will run the current search set up.

Autorun if selected will automatically run new search set ups as they change, so changing **Columns**, **Conditions**, **Report Type** and even what is being searched for, will get the search to be re-run as each setting is changed. If searches are being done on large amounts of data or the user needs to make multiple changes between search runs it is more efficient if **Autorun** is not selected. If at any time having made changes and **Autorun** is selected but a new search has not run then press the **Go** button.

3.10.1.1 Single Item Report

Shows the result of the search as a single record.

The screenshot shows the 'Single Item Report' interface. At the top, there is a search bar with 'User' entered, a 'Report type' dropdown set to 'Single Item Report', and a selection box containing 'Timb (Tim Bilder)' with a dropdown arrow. A 'Go' button is to the right. Below the search bar, it says 'Users where id equals 32'. A row of buttons includes 'Edit', 'Create', 'Create copy', 'Delete', 'References', and 'History'. The main area contains a table with user details:

Roles	User
Self	Timb (Tim Bilder)
Given name	Tim
Other name	
Family name	Bilder
Email address	ben@exprodo.com
User Type	
Expiry Date	
Password	•
Status	Blocked
Version	1
Created	4 Feb 2015 11:17
Updated	4 Feb 2015 11:17
Requested Project Code(s)	

Below the table are three tabs: 'Identity' (selected), 'Projects', and 'Groups'. Under the 'Identity' tab, it shows 'Local/Timb'. To the right of the table, two callout boxes provide instructions: 'Type in here to get matching report list' pointing to the search bar, and 'Drop down to choose single item to be reported on' pointing to the dropdown arrow next to 'Timb (Tim Bilder)'.

Either use the drop down to get a full list of possible options to be scrolled through or type in the input box to get a shortened list of possible matching objects.

If the user has the correct [permissions](#)⁶⁵⁴ they may edit the individual records or make changes to multiple records. All of this will be covered in the chapter on [Editing Search Information](#)¹²⁹

3.10.1.2 List Report

Shows the result of the search as a list of records

File Search for: Resource Report type: List Report Conditions... Columns... Reset Go ☒ Autorun

Found 8 resources

<input type="checkbox"/>	Name	Location	Type	Project Required	Collect Actual Usage
<input type="checkbox"/>	telescope	Harvard	Scanner	Project Required	true
<input type="checkbox"/>	wet lab	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	dry lab	Harvard	Room	Project Required	true
<input type="checkbox"/>	library	Tuscany	Room	Project Not Required	true
<input type="checkbox"/>	scanner	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	particle accelerator	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	leave & notices		Notices	Project Not Required	true
<input type="checkbox"/>	Subject Room	Harvard	Notices	Project Required	true

↑ Edit Delete

Specify which columns are viewed by clicking on the **Columns...** button and get a drop down list of tick box's for all available [properties](#)⁶⁵⁵, those who have their boxes ticked will be shown.

File Search for: Resource Report type: List Report Conditions... Columns... Reset Go ☒ Autorun

Found 8 resources

<input type="checkbox"/>	Name	Location	Type	Project Required	Collect Actual Usage
<input type="checkbox"/>	telescope	Harvard	Scanner	Project Required	true
<input type="checkbox"/>	wet lab	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	dry lab	Harvard	Room	Project Required	true
<input type="checkbox"/>	library	Tuscany	Room	Project Not Required	true
<input type="checkbox"/>	scanner	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	particle accelerator	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	leave & notices		Notices	Project Not Required	true
<input type="checkbox"/>	Subject Room	Harvard	Notices	Project Required	true

↑ Edit Delete

☒ Name
☒ Location
☒ Type
☒ Project Required
☒ Collect Actual Usage

If an individual record is clicked an expanded view of that record will appear in a pop-up (press the Escape key or click the red X to close it). If you shift-click, then the record will display full screen, and if you CTRL-Click, then the record will appear below the list.

File Search for: Resource Report type: List Report Conditions... Columns... Reset Go ☒ Autorun

Found 8 resources

<input type="checkbox"/>	Name	Location	Type	Project Required	Collect Actual Usage
<input type="checkbox"/>	telescope	Harvard	Scanner	Project Required	true
<input type="checkbox"/>	wet lab	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	dry lab	Harvard	Room	Project Required	true
<input type="checkbox"/>	library	Tuscany	Room	Project Not Required	true
<input type="checkbox"/>	scanner	planet earth	Scanner	Project Required	true
<input checked="" type="checkbox"/>	particle accelerator	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	leave & notices		Notices	Project Not Required	true
<input type="checkbox"/>	Subject Room	Harvard	Notices	Project Required	true

↑ Edit Delete

Edit Create Create copy Delete References History

Name	particle accelerator
Location	planet earth
Type	Scanner
Project Required	Project Required
Require Reason for Cancellations	<input type="checkbox"/>
Allow Old Changes	<input type="checkbox"/>
Collect Actual Usage	<input checked="" type="checkbox"/>

Actual Usage

Usage Recorder Options

Font size	Use large font
Access protection	Allow any IP Address

Usage Session ID Template

[Usage.id]

If the user has the correct [permissions](#)⁶⁵⁴ you may edit the individual records or make changes to multiple records. All of this will be covered in the chapter on [Editing Search Information](#)¹²⁹.

3.10.1.3 Summary Report

Shows the report as a table, where the user can define the [properties](#)⁶⁵⁵ that are used for the rows and the columns of the table, and how the content of each cell will be calculated.

When the [Summary Report](#)⁶⁵⁶ view appears the user will be asked to select the *properties* that will be used for the rows and for the column. Use the green plus to choose extra rows and the red minus to remove those not required.

The screenshot shows a web interface for configuring a 'Summary Report'. At the top, there is a 'File' dropdown menu, a 'Search for:' field containing the text 'Booking', and a 'Report type:' dropdown menu set to 'Summary Report'. Below these are three main sections: 'Rows', 'Columns', and 'Content'. The 'Rows' section contains two input fields: 'Row: booker' and 'Row: status', each followed by a green plus icon and a red minus icon. The 'Columns' section contains one input field: 'Column: resource'. The 'Content' section contains a 'Content:' dropdown menu set to 'Count' and a 'Go' button.

Once the *properties* have been selected, then select how the content is to be displayed.

This screenshot is similar to the previous one, but the 'Content:' dropdown menu is open, showing a list of options: 'Count', 'Count Distinct', 'Sum', 'Minimum', 'Maximum', 'Average', and 'Standard Deviation'. The 'Count' option is highlighted in blue. The 'Go' button is still visible below the dropdown.

The default content display is count, this will display the content as a simple count of how many records match the row/column combination, with totals for each row and column.

Rows

Row: + -

Row: + -

Columns

Column:

Content

Content: ▼

Go

booker	status	resource		Count of Bookings
		Leica SP8 confocal	Zeiss 780 confocal	
admin (admin)	Approved	6	2	8
admin (admin)	Cancelled	7		7
root (root)	Approved	1		1
	Count of Bookings	14	2	16

Complex Content Types

All the other content types require a numerical or date property as an additional parameter (the exception is **Count Distinct** which will work with any type) . The value of this parameter will be used to determine the value stored in each cell.

The screenshot shows a web application interface with a menu bar (Calendar, Templates, Bookings, Projects, Usage, Search, Help) and a toolbar (File, Search for: Resource, Report type: Summary Report, Conditions..., Reset). Below the toolbar, there are input fields for 'Row: location.name' and 'Column: projectRequired'. A 'Content:' dropdown menu is open, showing 'Sum' as the selected option. A list of properties is displayed below the dropdown, with 'shortTermMinutes' highlighted. There are also checkboxes for 'Scaled' and 'Rounded'. A 'Go' button is visible at the bottom left of the configuration area.

Content Type	Description
Count Distinct	How many different values to the <i>property</i> are there for this cell.
Sum	The total of this <i>property</i> for this cell (numerical only)
Minimum	The smallest value for this <i>property</i> for this cell
Maximum	The largest value of this <i>property</i> for this cell
Average	The average value of this <i>property</i> for this cell
Standard Deviation	The standard deviation of the values of this <i>property</i> for this cell.
Value At Minimum Of	Choose a property to find the minimum of, and the value shown in this column will be the value of the current property when the property being compared with is at a minimum.
Value At Maximum Of	Choose a property to find the maximum of, and the value shown in this column will be the value of the current property when the property being compared with is at a maximum.

Below is an example of a table showing the total booking duration in minutes for each [project](#)⁶⁵⁴ for each resource

File Search for: Booking Report type: Summary Report Conditions... Reset

Row: resource.name

Column: project.name

Content: Sum durationInMinutes ☐ Scaled ☐ Rounded

Go

Property to be calculated

Scale the result by a user defined factor

Round the results to integers

Sum of Booking.durationInMinutes	Medical imaging	heritability	inventions	light bulb	radiation	the final frontier		Total
Subject Room							300	300
dry lab	360		630	780	480	1080		3330
leave & notices							12951	12951
library		300		300	720	300	390	2010
particle accelerator	480			360		360		1200
scanner	3360	300	300	300	1620			5880
telescope						4090		4090
wet lab		1470						1470
Total	4200	2070	930	1740	2820	5830	13641	31231

Use the **Rounded** option to round numbers to integers

Use the **Scaled** option to scale the results by the user requested factor (i.e if the scaling factor is 10 divides all results by 10).

Viewing Records

Records associated with each cell can be viewed by clicking on the number in the cell. This will give a [List Report](#)⁶⁵³ of the records for that cell beneath the table.

File Search for: Resource Report type: Summary Report Conditions... Reset

Row: location.name

Column: projectRequired

Content: Count Scaled

Go

Count of Resources	PROJECT_NOT_REQUIRED	PROJECT_REQUIRED	Total
Harvard		3	3
Tuscany	1		1
planet earth		3	3
	1		1
Total	2	6	8

Resources where location.name = planet earth and projectRequired = PROJECT_REQUIRED

<input type="checkbox"/>	Name	Location	Type	Project Required	Collect Actual Usage
<input type="checkbox"/>	wet lab	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	scanner	planet earth	Scanner	Project Required	true
<input type="checkbox"/>	particle accelerator	planet earth	Scanner	Project Required	true

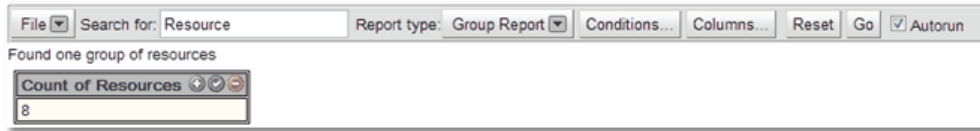
↑ Edit Delete

As with the [List Report](#)¹²⁰ clicking on an individual record will give an expanded view of that record below the list.

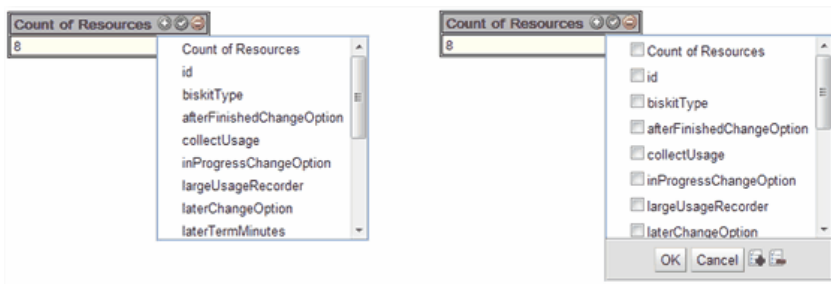
If the user has the correct [permissions](#)⁶⁵⁴ they may edit the individual records or make changes to multiple records. All of this will be covered in the chapter on [Editing Search Information](#)¹²⁹

3.10.1.4 Group Report

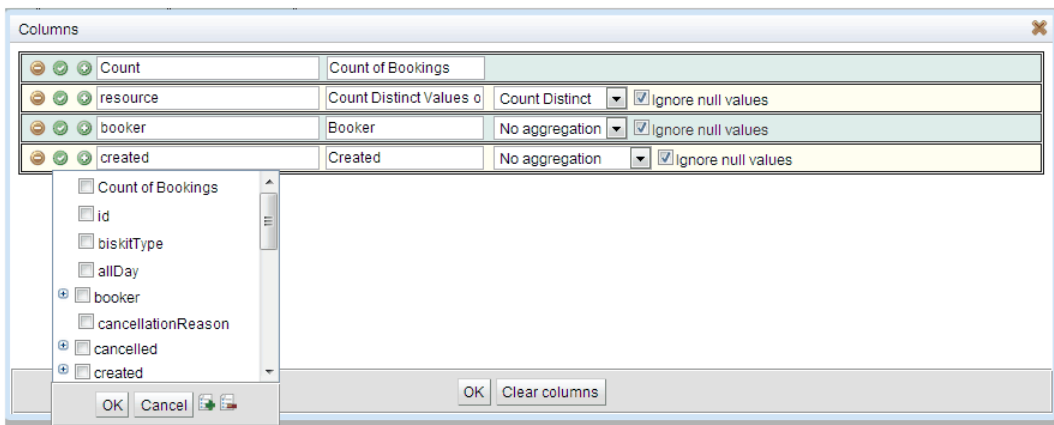
Shows the report as a table with one column showing a **Count of Biskit Type**. The table can then be extended to show other columns for [properties](#)⁶⁵⁵ available within the [Biskit Type](#)⁶⁵² chosen.



To choose more columns either use the **Columns...** button on the toolbar or the **green +** (to add one column), **red -** (to delete the current column), **green tick** (to add multiple columns, tick all columns that are required and then click **OK**), or use the **green +** icon on the menu bar to select all the columns or the **red -** icon to deselect all the options).



If the **Columns...** button is clicked a pop up appears, in which the columns can be chosen using the same three buttons as described above. At the same time the user can decide what aggregation to use, **No aggregation** or **Count Distinct**, as well as whether to **Ignore null values**.



Once the columns have been chosen the table will redisplay using the appropriate number of rows to display the records, grouping records by the columns chosen giving a count in the first column.

File Search for: Resource Report type: Group Report Conditions... Columns... Reset Go [x] Autorun

Found 4 groups of resources

Count of Resources	location.name	Project Required
1	Tuscany	Project Not Required
1		Project Not Required
3	Harvard	Project Required
3	planet earth	Project Required

Once the columns are displayed, move the cursor over the name of a column and a pop up will appear with **Change** in it. Click on **Change** in order to change

1. The column header
2. The column content

Sum of Hits [x] [x] [x] Date [x] [x] [x]

OK Cancel

Column Content

hits

Sum

☐ Ignore null values

☐ Scaled

Column Heading

Sum of Hits

The column content can be changed to

1. Another *property*.
2. A complex content type, see [Complex Content Type](#)¹²⁴ in the [Summary Report](#)¹²² section for more details on the types available.
3. Whether null values are ignored.
4. Whether the result should be scaled see [Complex Content Type](#)¹²⁴ in the [Summary Report](#)¹²² section for more details on scaling.

Clicking on a cell will produce a [List Report](#)⁶⁵³ of the records associated with that cell below the table.

File Search for: Resource Report type: Group Report Conditions... Columns... Reset Go [x] Autorun

Found 4 groups of resources

Count of Resources	location.name	Project Required
1	Tuscany	Project Not Required
1		Project Not Required
3	Harvard	Project Required
3	planet earth	Project Required

Name	Location	Type	Project Required	Collect Actual Usage
wet lab	planet earth	Scanner	Project Required	true
scanner	planet earth	Scanner	Project Required	true
particle accelerator	planet earth	Scanner	Project Required	true

↑ Edit Delete

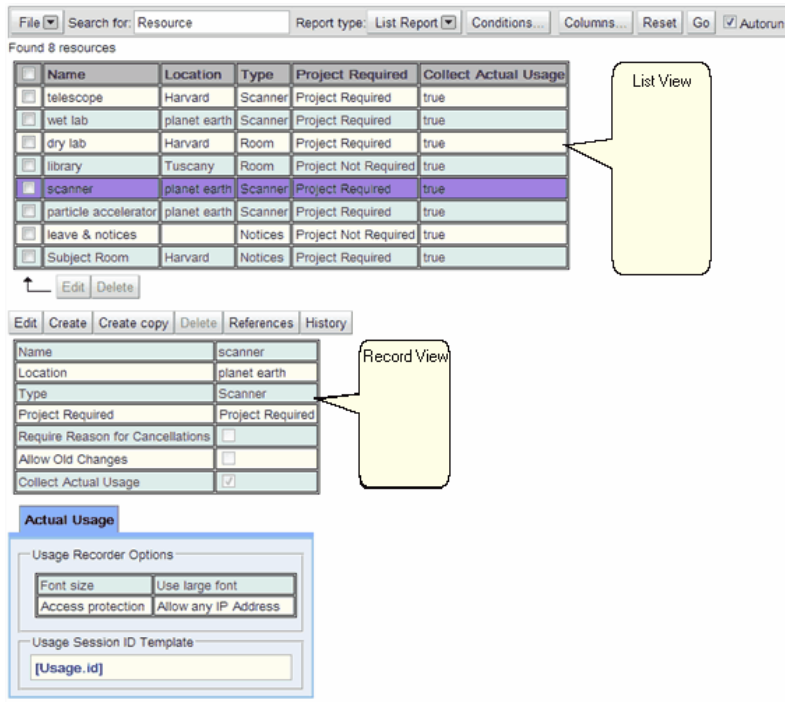
As with the [List Report](#)¹²⁰ clicking on an individual record will give an expanded view of that record below the list.

If the user has the correct [permissions](#)⁶⁵⁴ they may edit the individual records or make changes to multiple records. All of this will be covered in the chapter on [Editing Search Information](#)¹²⁹

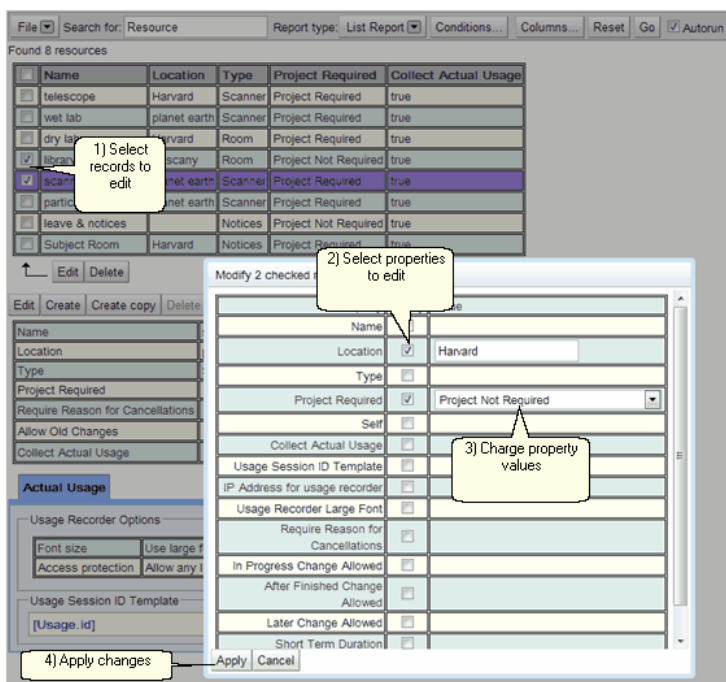
3.10.1.5 Editing Search Information

Once the search information is in either a [List Report](#)⁶⁵³ view or as an individual record the information can be edited if the user has the [permissions](#)⁶⁵⁴ to do so. The administrator will have set up the *permissions* for your **Calpendo**.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.



For a more complete explanation on how to edit records in the list view read the [How to Edit Multiple Items At Once](#)¹⁴⁰ section of the [Data Explorer](#)¹³⁹ chapter.



For a more complete explanation on how to edit records in the record view read the [How to Edit A Single Item](#)¹⁴² section of the [Data Explorer](#)¹³⁹ chapter.

The screenshot shows the 'Edit' form for a record named 'scanner'. The form has a tabbed interface with tabs for 'Edit', 'Create', 'Create copy', 'Delete', 'References', and 'History'. The 'Edit' tab is active, showing a table of properties:

Name	scanner
Location	planet earth
Type	Scanner
Project Required	Project Required
Require Reason for Cancellations	<input type="checkbox"/>
Allow Old Changes	<input type="checkbox"/>
Collect Actual Usage	<input checked="" type="checkbox"/>

Below the table is a section titled 'Actual Usage' with a sub-section 'Usage Recorder Options' containing:

Font size	Use large font
Access protection	Allow any IP Address

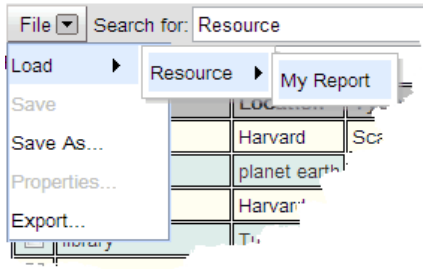
Below this is a 'Usage Session ID Template' section with a text box containing '[Usage.id]'.

Once in edit mode the changes can be made.

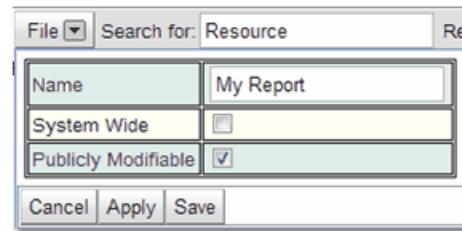
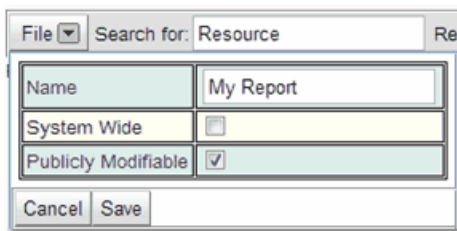
This screenshot shows the same 'Edit' form, but with the 'Actual Usage' section expanded. The 'Usage Recorder Options' section now includes dropdown menus for 'Font size' (set to 'Use large font') and 'Access protection' (set to 'Allow any IP Address'). The 'Usage Session ID Template' section still shows '[Usage.id]'.

3.10.1.6 Saving And Reusing Searches

Once a search has been created that is useful the user may wish to save it for later use. This can be done using the drop down button from the **File** option on the toolbar and choosing **Save As**. This will save the search as a report accessible from the [Search->Reports](#) page. Using the **File** option the user can also load saved searches to run them, edit the [properties](#) of any saved searches and export search information to a file called **report.csv**.



When a search has been created use the **File->Save As** to save the search. Give it a **Name**, decide whether the search is accessible to all users (**System Wide**) and whether anyone can edit the search (**Publicly Modifiable**). Then save the search.



When a search is saved it can be reloaded using **File->Load**. If a search is loaded then the *properties* can be edited using the **File->Properties** menu option. When edited they can be saved. In this case **Save** and **Apply** do the same thing, that is save the changed information and leave edit mode.

Exporting Search Information

A user can export any data they have found through a search to a .csv, a tab separated file or native **Excel** file (either pre 2007 xls or post 2007 xlsx) which can then be loaded into such applications as **Excel**.

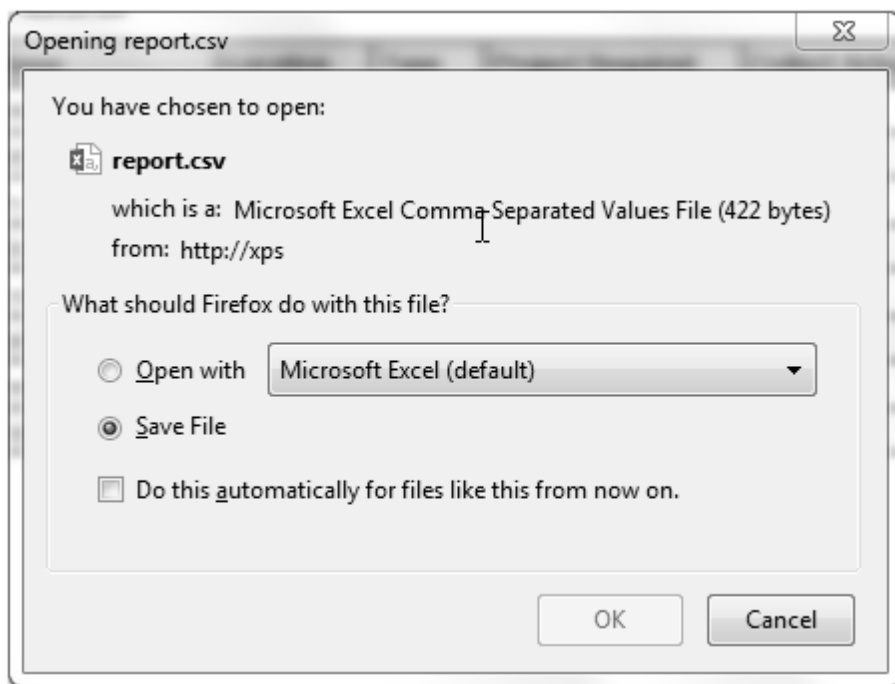
Using **File->Export** the current search data will be exported to a file.

There are three options:

1. How the file will be formatted, choose between **Comma Separated Values**, **Excel xls**, **Excel xlsx** and **Tab Separated Values**.
2. What headings will be used, these can be **Show labels** or **Show property names**. Labels are those names that have been assigned by the administrator to a [Biskit Type](#)⁶⁵², the *property* name is the name of the *property* in the database.
3. Whether enumerated types have their names shown, **Show value labels** where defined or the value as stored in the database, Show underlying data value.



When the format for the export is chosen press **Export**, then choose whether to open the exported information directly into another application, or whether to save the file. If the file is saved it will be called **report.csv** and found in the users **Downloads** folder.

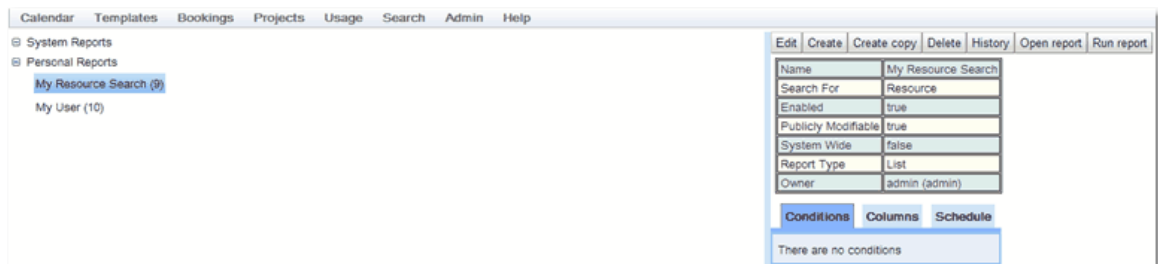


3.10.2 Report Manager

The **Report Manager** page allows the management of reports within **Calpendo**. By default, it appears under the **Search** menu, with a sub menu item labelled **Reports**. However, the menu may have been configured so that the **Reports** page is not visible by the administrator.

Once useful searches have been created they can be saved for future use as reports (see [Saving and Reusing Searches](#)¹³¹). The reports page allows you to view, manage and schedule your reports as well as creating new ones.

The page is split into two. On the left is a list of the **System Reports** (those that anyone can see) and the users **Personal Reports**. On the right is the area where to edit, schedule and create reports, when the user enters the **Reports** page this area is initially empty. Select a report in the left hand pane to see the [properties](#)⁶⁵⁵ of that report in the right hand pane. Where the page splits can be moved to change the space available for each of the sides.



A report has a number of basic properties:

Property	Description
Name	The name of the report as assigned by the user.
Search For	The biskit type ⁶⁵² that will be searched for.
Enabled	If the report will run if scheduled.
Publicly Modifiable	If a system wide report then anyone can modify it.
System Wide	Can everyone see the report or is it a personal one. If this is true the report will appear under System Reports rather than Personal Reports .
Report Type	Will the report display as List ⁶⁵³ , Summary ⁶⁵⁶ or Group ⁶⁵³ report
Owner	The user who created the report and if Publicly Modifiable is false, the only person who can edit a report if it is also System Wide
Page Token	Records which page the report was created with.

and tabs to see other properties:

Tab	Description
Conditions	The conditions ⁶⁵³ (if any) associated with the report. (see the Setting Search Conditions chapter for more information about <i>conditions</i> and how to edit them)
Columns	The columns that will be displayed in the report
Schedule	Whether the report is scheduled to run and, when it will run, who the report will be e-mailed to and the format of that e-mail.

The Menu Bars

The menu bar has two variations. In **View** mode:



after you press the **Edit** button and enter edit mode:



For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Report Menu Bar	Description
Open Report	Opens the Search page in a state ready to run your report. Click Go to run the report. This will be greyed out in edit mode
Run report	Opens the Search page and runs the report. This will be greyed out in edit mode

Editing Reports

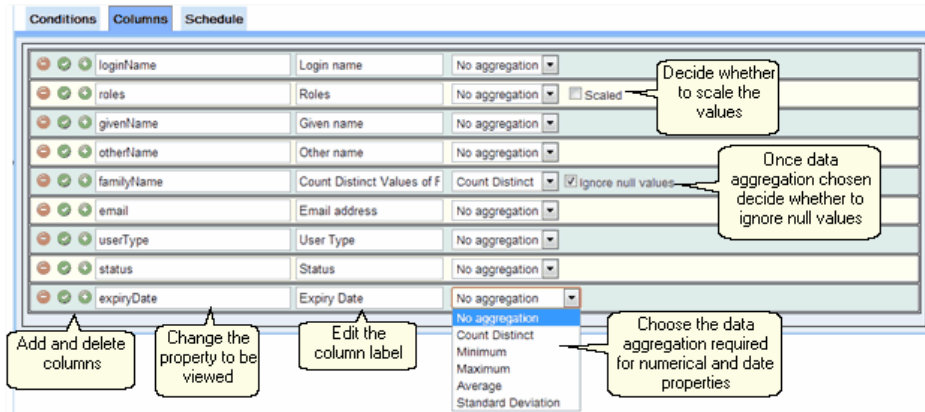
Once a report is in edit mode the basic *properties* can be changed in the top section, all have drop down menus for the options available other than the report name which can be overwritten.

Name	Resource Report
Search For	Resource
Enabled	true
Publicly Modifiable	true
System Wide	false
Report Type	List Report
Owner	admin (admin)

Use the tabs to add or change the *conditions* (see the Setting Search Conditions chapter for more information about *conditions* and how to edit them).

Conditions	Columns	Schedule
All of the following apply		
Value	of name	starts with
		l

Use the tabs to change the columns that are viewed, what is viewed in them and how they are labeled. Here is the Group report version. The formatting is different for the Summary and List report versions.



The first column allows the user to add or delete columns from the report, the **green +** (to add one column), **green tick** (to add multiple columns, tick all columns required to be added then click **OK**), **red -** (to delete the current column). (see the [Group Report](#)¹²⁷ chapter for more information on how they work)

The second column allows the user to change the *properties* that will be viewed in a column.

The third column allows the user to change the label at the top of the column in the report.

The fourth column allows the user to decide whether absolute values or data aggregation is required. See [Summary Report](#)¹²² chapter, [Complex Content Types](#)¹²⁴ section for more information. If aggregation of the data has been chosen, then decide whether to ignore null (empty or 0) values.

Scheduling Reports

If box marked **Run report on a schedule** is ticked, options allowing for the setting up of the scheduling of the report appear.

Run Now button to run the report now rather than wait for the scheduled time.

Choose whether to attach the report as a file to the email or whether the body of the email is the report data.

Choose the subject line to appear on the email. Select the **Recipients** tab to decide who to send the email to.

Select the individual users the mail is to go to.

Select the [User Types](#)⁶⁵⁶ the mail is to go to. The mail will be sent to all users of these types.

Select the [User Groups](#)⁶⁵⁶ the mail is to go to. The mail will be sent to all users in these groups.

Select the [User Roles](#)⁶⁵⁶ the mail is to go to. The mail will be sent to all users with these roles.

Emails will only be sent to an address once. So if multiple users have the same email address or with your selections of **Individual**, **Type**, **Group** or **Role** you would have accessed a user multiple times, they will only get one email.

Select the **When** tab to schedule the report.

Next Send: The time and date the mail will next be sent with the current settings.

Last Sent: The time and date the report was last sent.

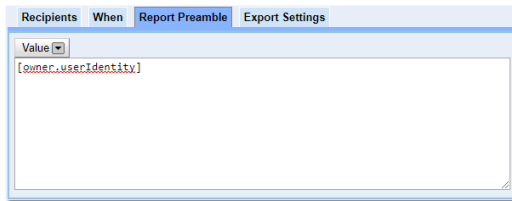
Select the frequency of the mail: Daily, Weekly, Monthly by Date, Monthly by Day, Yearly

Select when the emails will start

Specify the frequency: every other day, every third week etc.

Select when the emails will end if required.

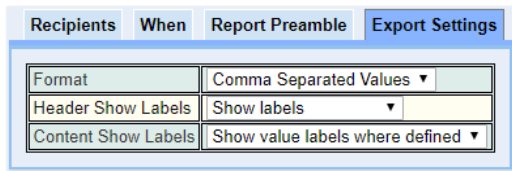
Select the **Report Preamble** tab to add some text to the detail of the email



The screenshot shows a window with four tabs: 'Recipients', 'When', 'Report Preamble', and 'Export Settings'. The 'Report Preamble' tab is active. Below the tabs is a text area labeled 'Value' containing the text '[owner.userIdentity]'.

Allows the addition of some text to the email detail.

Select the **Export Settings** tab to set up the format of the exported file.



The screenshot shows a window with four tabs: 'Recipients', 'When', 'Report Preamble', and 'Export Settings'. The 'Export Settings' tab is active. Below the tabs is a table with three rows: 'Format' (Comma Separated Values), 'Header Show Labels' (Show labels), and 'Content Show Labels' (Show value labels where defined).

1. **Format:** How the file will be formatted, choose between **Comma Separated Values**, **Excel xls**, **Excel xlsx** and **Tab Separated Values**.

1. **Header:** What headings will be used, these can be **Show labels** or **Show property names**. Labels are those names that have been assigned by the administrator to a *biskit* type, the property name is the name of the *property* in the database.
2. **Content:** Whether enumerated types have their names shown, **Show value labels where defined** or the value as stored in the database, **Show underlying data value**.

3.10.3 History

The **History** page allows the user to look into the audit log kept by **Calpendo** that shows what has been changed, when and by whom. This page is normally found on the **Search** menu as **Search-->History**. However, the menu may have been configured so that the **History** page is not visible.

To use the page, first enter the search criteria, and then press the **Go** button to see what matching changes can be found. The search criteria can be:

- **Data type**, choose what type of **Biskit**⁶⁵² has been changed. For example, if the user is interested in changes to **bookings**⁶⁵², then select **Booking**. If a **Biskit Type**⁶⁵² is not selected, then the search will include changes made to any type of **Biskit**.
- **Search from** and **Search to**, to restrict the search to particular periods of time. It is highly recommended to put limits on the time period using these options so that the amount of returned results are limited.
- **User**, choose to see changes made only by one particular user, or by anybody.
- **Data ID**, choose the unique ID number to search for. For example, if the user is interested in a **project**⁶⁵⁴, and they know its ID is 42, then enter 42 here to restrict results to changes made to project 42.
- **IP Address**, search by changes made from a particular internet address.

After pressing **Go**, a list of matching changes is displayed. Each one will represent a change to something within **Calpendo**. When the list appears, click on one of them to see all changes to the same **Biskit**. For example, if the search was for changes to **resources**⁶⁵⁵ made on a particular day, returned will be a list of changes that affected all the **resources** on that day. By clicking on one of them, the user would then see all changes made to that **resource**, regardless of when that change was made.

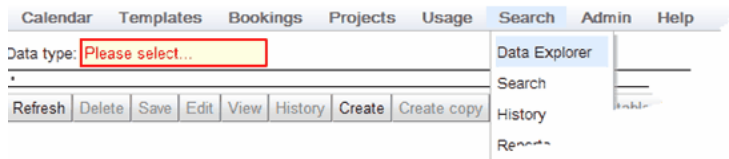
Log date	Editor	Change Type	IP Address	Data ID	Name	Location	Type	Project Required	Collect Actual Usage
10 Apr 2013 17:29	admin (admin)	DELETE	192.168.1.77	11	IRC-MIRI	Tuscany	Scanner	Project Required	true
10 Apr 2013 17:29	admin (admin)	DELETE	192.168.1.77	9	3TLab	Harvard	Scanner	Project Required	true
16 Apr 2013 14:41	admin (admin)	CREATE	192.168.1.77	12	new scanner	Harvard	Scanner	Project Required	true
16 Apr 2013 14:42	admin (admin)	DELETE	192.168.1.77	12	new scanner	Harvard	Scanner	Project Required	true
16 Apr 2013 14:42	admin (admin)	CREATE	192.168.1.77	13	scan	Harvard	Device	Project Required	true
16 Apr 2013 14:43	admin (admin)	DELETE	192.168.1.77	13	scan	Harvard	Device	Project Required	true
16 Apr 2013 14:43	admin (admin)	CREATE	192.168.1.77	14	scan	Harvard	Device	Project Required	true
16 Apr 2013 15:18	admin (admin)	UPDATE	192.168.1.77	8	Subject Room	Harvard	Notices	Project Required	true
16 Apr 2013 15:24	admin (admin)	DELETE	192.168.1.77	14	scan	Harvard	Device	Project Required	true

History of resource 8

Log date	Editor	Change Type	IP Address	Name	Location	Type	Project Required	Collect Actual Usage
26 Apr 2011 19:17	admin (admin)	CREATE	88.26.57.24	Subject Room	Harvard	Room	Project Required	true
16 Apr 2013 15:18	admin (admin)	UPDATE	192.168.1.77	Subject Room	Harvard	Notices	Project Required	true

3.10.4 Data Explorer

The **Data Explorer** page allows examination of the data within **Calpendo**. By default, it appears on the menu here:



Select a [Biskit Type](#)⁶⁵², and a table appears that shows all [Biskits](#)⁶⁵² of that type.

Note: The **Data Explorer** page does not yet handle well the situation when there are lots of *Biskits* of the selected type. It is highly recommended that the user does not attempt to use the **Data Explorer** to view [Bookings](#)⁶⁵², for example, since there may be many of them.

Apart from being able to choose the type of data to look at, the **Data Explorer** page is identical to several other pages in **Calpendo**, such as the list of users shown in Modifying Users.

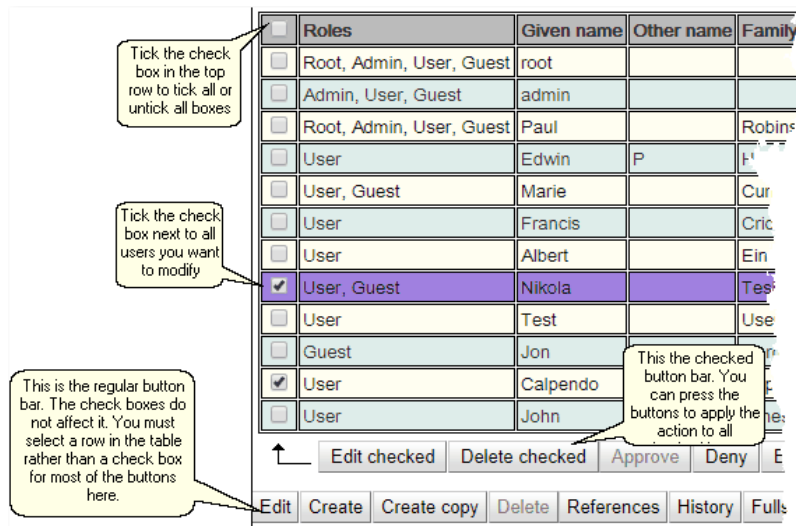
Select **User** as the *Biskit Type*, then a list of users will appear and the user will be able to examine or modify them ([permissions](#)³¹⁴ permitting).

<input type="checkbox"/>	Roles	Identity	Given name	Family name	Email address	User Type	Status	Expiry Date	Version	Created	Updated	Test Date
<input type="checkbox"/>	Guest, Root, User, Admin	Local/root	root				Normal		0	26 Sep 2014 12:53	26 Sep 2014 12:53	
<input type="checkbox"/>	Guest, User, Admin	Local/admin	admin		info@exprodo.com		Normal		0	12 May 2014 15:40	26 Sep 2014 12:54	
<input type="checkbox"/>	User	Local/paul	Paul	Robinson	paul@exprodo.com		Normal		1	26 Sep 2014 13:06	26 Sep 2014 13:06	
<input type="checkbox"/>	User	Local/leonardo	Leonardo	Da Vinci	blah@blah.com	Physics	Normal		1	26 Sep	The date and time when this item was created	
<input type="checkbox"/>	User	Local/hubble	Edwin	Hubble	blah@blah.com	Physics	Normal		0			
<input type="checkbox"/>	User	Local/mendel	Gregor	Mendel	blah@blah.com	Biology	Normal		0			
<input type="checkbox"/>	User	Local/roentgen	Wilhelm	Rontgen	blah@blah.com	Physics	Normal		0			
<input type="checkbox"/>	User	Local/newton	Isaac	Newton	blah@blah.com		Normal		3	6 Aug 2015 11:52	2 Sep 2015 13:15	14 Aug 2015
<input type="checkbox"/>	User	Local/darwin	Charles	Darwin	blah@blah.com	Biology	Normal		0			
<input type="checkbox"/>	User	Local/curie	Marie	Curie	ben@exprodo.com	Physics	Normal		4	11 Feb 2015 08:29	16 Mar 2015 08:24	
<input type="checkbox"/>	User	Local/galileo	Galileo	Galilei	blah@blah.com	Physics	Normal		0			
<input type="checkbox"/>	User	Local/Crick	Francis	Crick	blah@blah.com		Normal		1	2 Sep 2015 13:16	2 Sep 2015 13:16	
<input type="checkbox"/>	User	Local/Watson	Francis	Watson	blah@blah.com	Biology	Normal		0			
<input type="checkbox"/>	User	Local/Einstein	Albert	Einstein	jon@exprodo.com		Normal		2	9 Feb 2015 16:00	1 Sep 2015 14:46	
<input type="checkbox"/>	User	Local/Edison	Thomas	Edison	blah@blah.com	Physics	Normal		0			
<input type="checkbox"/>	User	Local/Tesla	Nikola	Tesla	blah@blah.com	Physics	Normal		0			
<input type="checkbox"/>	User	Local/Planck	Max	Planck	blah@blah.com		Normal		1	2 Sep 2015 14:07	2 Sep 2015 14:07	
<input type="checkbox"/>	User	Local/user	Test	User	blah@blah.com	Physics	Normal		1	4 Aug 2015 14:23	4 Aug 2015 14:23	
<input type="checkbox"/>	User	Local/garyl	gary	Lincoln	gary@exprodo.com	Biology	Normal		2	22 Jul 2015 15:12	22 Jul 2015 15:16	
<input type="checkbox"/>	User	Local/tester	test	tester	test@exprodo.com	Biology	Normal		2	22 Jul 2015 15:21	22 Jul 2015 15:44	

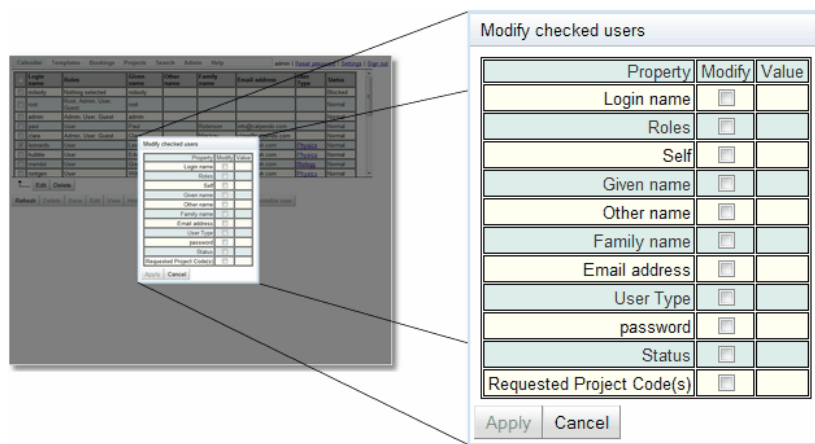
How To Edit Multiple Items At Once

You can edit many items in one go quite easily, using the check boxes and the checked button bar as follows:

1. Tick the check boxes next to each item to be changed, or tick the box in the header row to tick all boxes.
- 2.



3. Once one or more check boxes have been ticked, the checked button bar will change so it is no longer greyed out. Press the **Edit Checked** button to edit all the ticked items.
4. When you press the **Edit Checked** button, a pop-up appears showing most of the [properties](#)⁶⁵⁵ that are defined for the selected *Biskit Type*. However, some may be missing if they are too complex to edit in this pop-up (for example, *properties* that contain lists of things). If these *properties* are **Grouped** or have a **Layout** defined then the pop up will be hierarchical.



5. Tick the check boxes of one or more *properties* shown in the pop-up. Once done, the **Apply** button changes so it is no longer greyed out, and an editor appears in the **Value** column:

Property	Modify	Value
Login name	<input type="checkbox"/>	
Roles	<input type="checkbox"/>	
Self	<input type="checkbox"/>	
Given name	<input type="checkbox"/>	
Other name	<input type="checkbox"/>	
Family name	<input type="checkbox"/>	
Email address	<input type="checkbox"/>	
User Type	<input type="checkbox"/>	
password	<input type="checkbox"/>	
Status	<input checked="" type="checkbox"/>	Please select a Status ▼
Requested Project Code(s)	<input type="checkbox"/>	

Apply Cancel

6. Choose a value for each *property* whose check box you tick. When finished, press the **Apply** button. Every ticked item will then be modified, with only the ticked *properties* being changed. Whatever values selected for each *property* will be applied to all the ticked items. *Properties* that are not ticked are not affected by the change.

How To Edit A Single Item

While it is very useful being able to edit multiple items at once, the above mechanism doesn't let you edit every *property*. If the user wants to change the more complex *properties*, such as lists of things, then they need to use a different change mechanism.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

1. Click anywhere in the row for the item to edit, apart from its check box in the first column. The item's details are shown below the list of items.

The screenshot shows the 'Users' table with columns: Login name, Roles, Given name, Other name, Family name, Email address, User Type, and Status. The row for 'leonardo' is selected. Below the table, the 'Edit' form is displayed, showing details for user 'leonardo'.

Project Code	Type	Status	Owner	Name	Start	Finish	Other Investigators
space	Approved	gallio	the final frontier				
genetics	Approved	mandal	heritability				
imaging	Approved	rontgen	Medical imaging				
inventions	Approved	leonardo	inventions				
light bulb	Requested	Edison	light bulb				

2. Press the **Edit** button (not the one in the checked button bar)

The screenshot shows the same interface as before, but with the 'Edit' button in the toolbar highlighted by a red box. A red arrow points from the 'Edit' button to the 'Edit' form below the table.

3. The item is now shown with the *properties* editable. Note that, depending on how [permissions](#)⁶⁵⁴ have been set up, it is possible that some *properties* will still show as read-only.

Refresh Delete Save Edit View History Create Create copy References Printable view

User 6

Login name	leonardo
Roles	User
Given name	Leonardo
Other name	
Family name	Da Vinci
Email address	blah@blah.com
User Type	Physics
password	
Status	Password must be reset at next login
Requested Project Code(s)	

Projects Groups

Please select a Project to add

Project Code	Type	Status	Owner	Name	Start	Finish	Other Investigators
space		Approved	galileo	the final frontier			
genetics		Approved	mendel	heritability			
imaging		Approved	rontgen	Medical imaging			
inventions		Approved	leonardo	inventions			
light bulb		Requested	Edison	light bulb			

Remove

4. Once changing *properties* has been completed, press the **Save** button. The item is then saved and shown read-only again.

Calendar Templates Bookings Projects Search Admin Help

admin | Reset password | Settings | Sign out

Login name	Roles	Given name	Other name	Family name	Email address	User Type	Status
nobody	Nothing selected	nobody					Disabled
root	Root, Admin, User	root					Normal
admin	Admin, User, Guest	admin					Normal
paol	User	Paul		Colson	paol@calpendo.com		Normal
clara	Admin, User, Guest	Clara		Blackley	clara@calpendo.com		Normal
leonardo	User	Leonardo		Da Vinci	leon@blah.com	Physics	Password must be reset at next login
hubble	User	Edwin	IP	Hubble	leon@blah.com	Physics	Normal

Edit Delete

Refresh Delete Save Edit View History Create Create copy References Printable view

User 6

Login name	leonardo
Roles	User
Given name	Leonardo
Other name	
Family name	Da Vinci
Email address	blah@blah.com
User Type	Physics
password	
Status	Password must be reset at next login
Requested Project Code(s)	

Projects Groups

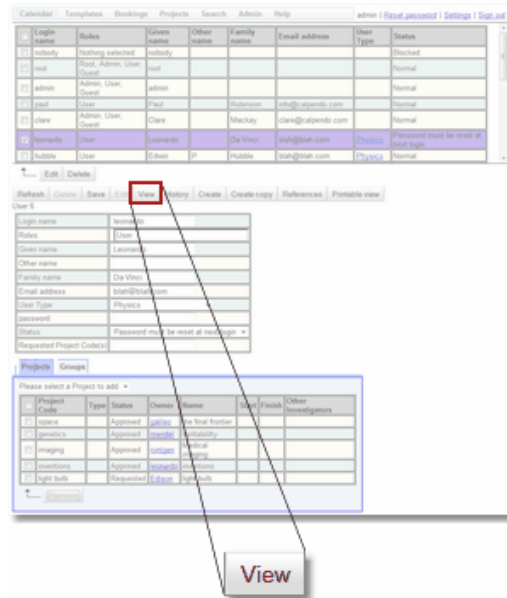
Please select a Project to add

Project Code	Type	Status	Owner	Name	Start	Finish	Other Investigators
space		Approved	galileo	the final frontier			
genetics		Approved	mendel	heritability			
imaging		Approved	rontgen	Medical imaging			
inventions		Approved	leonardo	inventions			
light bulb		Requested	Edison	light bulb			

Remove

Save

5. To cancel the edit while an item is shown in an editable format, either click on the row for a different item or press the **View** button.



Configuring The Properties Displayed

The user can control which *properties* are displayed in the list of items, in the detailed view of a item. This is done in the [Bakery](#)⁵³⁷.

- [Properties Visible In A Biskit List](#)⁵⁴⁷ explains how to change the *properties* that are displayed in a list of items. Change the visibility of each of the *properties* defined on an item.
- [Properties Visible In Biskit Detail](#)⁵⁴⁷ explains how to change the *properties* that are displayed when seeing a detailed view of an item. In this case, it affects both the read-only and editable mode detailed view of an item. Again, change the visibility of each of the *properties* defined on an item.
- [Properties Visible In A Collection Editor](#)⁵⁴⁷ explains how to change the *properties* that are displayed in collection editors. For example, if there is a [Project](#)⁶⁵⁴ *Biskit Type* that contains a list of **Users**, then the user can change which *properties* are shown to represent those users in the *Project's* list of users.

3.11 Frequently Asked Questions

If there are any questions relating to the operation of your **Calpendo** at the facility, check the **Frequently Asked Questions** section of **Calpendo**. This can be found on the **Help** menu, and is called [FAQ](#)⁶⁵³.

The administrator will configure these *FAQs* so that they are appropriate for the facility (as described in the [Configuring Frequently Asked Questions](#)⁶⁰⁶ chapter).

3.12 Web Browser Compatibility

Calpendo supports the most popular web browsers: Firefox, Safari, Google Chrome and Opera. Internet Explorer is also supported from version 9, but is much slower than the other browsers. Older versions of Internet Explorer will mostly work, but will be extremely slow and some behaviour will be incorrect.

Name	Version	Comments
Google Chrome	All	Supported
Firefox	3.0 and later	Supported
Apple Safari	4, and later	Supported
Opera	40 and later	Supported
Internet Explorer	6	Not supported. Very slow and doesn't always behave properly.
	7	Not supported. Very slow and doesn't always behave properly.
	8	Not supported. Very slow and doesn't always behave properly.
	9, 10, 11	Supported, but slower than other browsers.
Edge		Supported

Part

IV

4 Calpendo Administration Guide

The **Calpendo Administration Guide** covers the day-to-day administration that is required for a **Calpendo** installation. This guide should be read in conjunction with the [Calpendo User Guide](#)²⁸ which, as well as describing how to make [bookings](#)⁶⁵² and create or edit [projects](#)⁶⁵⁴, also describes how to search for information.

See the [Calpendo Configuration Guide](#)²¹⁰ for information on how to change the [properties that exist on a project](#)²¹⁸ or the [default value of properties on a project](#)²¹⁸.

4.1 Booking Administration

When a [booking](#)⁶⁵² is created, it will have a [status](#)⁶⁵³ of either [Requested](#)⁶⁵⁵ or [Approved](#)⁶⁵², depending on the [Time Templates](#)⁶⁵⁶ that are in force for the duration of the *booking*. If there are no applicable [Time Templates](#) for a newly created *booking*, then its *status* is determined by a [Global Preferences](#)⁶⁵³ setting. An administrator will need to approve any *booking* requests and possibly make *bookings* for people who need to make *bookings* outside the normal rules.

The administrator may also need to resolve conflicts between people for which they will need to find out when *bookings* were made or modified. Some [properties](#)⁶⁵⁵ on each *booking* will help with this, but also use the [History](#)¹³⁸ page.

See Also:

- [History](#)¹³⁸.
- [Configuring Time Templates](#)²²⁶.
- The [Bookings](#)⁵¹⁵ section of the [Global Preferences](#)⁵⁰⁹.

4.1.1 Booking Properties

The most important [properties](#)⁶⁵⁵ on a [booking](#)⁶⁵² are shown in this table, and these can be seen on the [Bookings Request Page](#)¹⁵¹:

Property Name	Description
Resource	This is the resource ⁶⁵⁵ being booked (the room or equipment etc)
Booker	This is the person that made the booking.
Owner	This is the person that owns the <i>booking</i> . When a <i>booking</i> is created, if the Global Preferences ⁵¹⁵ Owner Defaults to Project Owner is set to True this is automatically set to be the <i>project's</i> owner ⁶⁵⁴ if there is a <i>project</i> , and the booker ⁶⁵² otherwise. Permissions ³¹⁴ determine whether you can change the <i>owner</i> (or indeed any other <i>property</i> of a <i>booking</i>).
Project	The project ⁶⁵⁴ indicates what the <i>resource</i> is being used for. Some <i>resources</i> are configured so that their <i>bookings</i> do not require a <i>project</i> . Some <i>resources</i> can be configured so that the <i>project</i> must have the <i>resource</i> in their Project Resource Settings ⁶⁵⁴ in order to be able to book that <i>resource</i> . This enables Booking Rules to be implemented about the maximum number of <i>bookings</i> as well as get reports on costs.
Project Resource Settings	This is a special <i>property</i> whose value cannot be set by the user. Each <i>project</i> has a <i>Resource Settings property</i> that is a list of settings for each <i>resource</i> . On a <i>booking</i> , the <i>Project Resource Settings</i> property evaluates to the settings defined on the <i>booking's project</i> for the <i>booking's resource</i> . Note that if the <i>project's</i> settings does not include anything for the <i>booking's resource</i> , then the <i>booking's Project Resource Settings</i> property will evaluate to null.
Type	This is the booking type ⁶⁵³
Status	A booking status ⁶⁵³ is one of Requested , Tentative , Approved , Denied and Cancelled .
Date Range	This indicates both the start and the finish of the <i>booking</i> . A Calpendo booking is currently limited to finishing on the same day that it starts. This may be relaxed in a future version, but repeat ⁶⁵⁵ <i>bookings</i> reduce some of the effects of this limitation.
All Day	This records whether the <i>booking</i> is for the whole day. All day <i>bookings</i> do not require a start and finish time, just a date.
Repeat	This indicates whether and how the <i>booking</i> should repeat. Your Permissions ³¹⁴ may be configured to control who can create or modify a <i>repeat booking</i> . See Repeat Bookings ⁵⁷ in the Creating Bookings ⁵³ page for details of <i>repeats</i> .

Find out when a *booking* was modified and by whom by using the [History](#)¹³⁸ page. However, a *booking* also records some dates in its [history](#)⁶⁵³ to help in this process, and by clicking on a *booking* in the [Bookings Calendar](#)³⁹, it will show these dates. Note that this records the dates but not who performed the actions - use the [History](#)¹³⁸ page for that.

Property Name	Description
Created	This records when the <i>booking</i> was created. This <i>property</i> is set automatically and cannot be changed.
Cancelled	This records the last time the <i>booking</i> was modified in such a way that its <i>status</i> changed and its <i>status</i> became Cancelled .
Modified	This records when the <i>booking</i> was last modified. Again, this is set automatically whenever a <i>booking</i> is modified.
Version	The version number of the <i>booking</i> .

Finally, there are some ancillary properties:

Property Name	Description
Description	This is a free-text description of whatever the <i>booker</i> wanted to record. This text is displayed with the <i>booking</i> on the Bookings Calendar ³⁹ .
Cancellation Reason	The reason the booking was canceled. This is a Mapped Int and all the reasons can be changed in the Bakery ⁵³⁷ .
Duration In Minutes	This is a <i>property</i> that is calculated from the <i>booking's</i> Date Range <i>property</i> . This cannot be set directly, but it is useful for reporting.
Warned	This records whether or not the user was shown a warning by a Time Template ⁶⁵⁶ when the <i>booking</i> was made or modified.
Template Approved	This records whether the <i>booking</i> was approved by a Time Template ⁶⁵⁶ , as opposed to being approved manually. This allows you to configure different Permissions ³¹⁴ and Booking Rules ²³⁵ depending on how a <i>booking</i> came to be approved.
Reminder information	This stores information about who should be reminded about a <i>booking</i> and when. It also records whether the reminder has been sent so that it can't be sent twice.

4.1.2 The Booking Approval Process

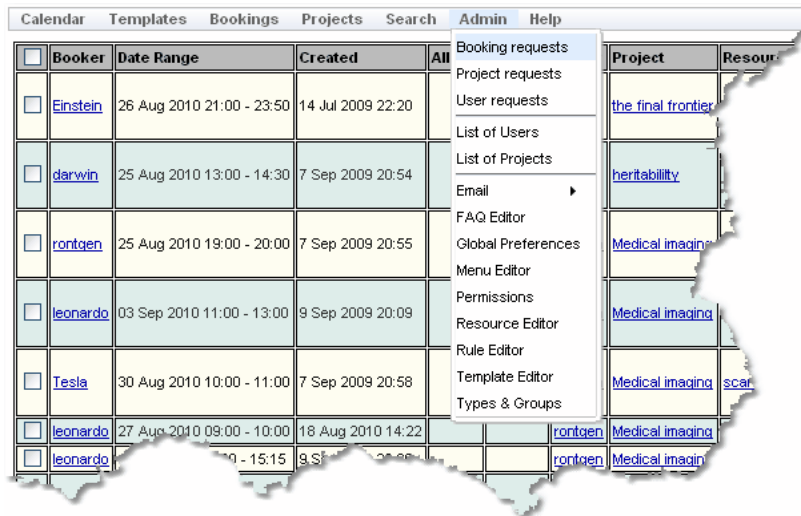
The [booking approval process](#)⁶⁵² is very similar to the [The User Approval Process](#)¹⁶⁷ and is a matter of changing a *booking's status*⁶⁵³ from **Requested or Tentative** to **Approved** if the *booking*⁶⁵² is to be allowed, or else to **Denied**. The only major differences are that:

- *Bookings* may be created pre-approved so that no one needs to manually approve any *bookings*, or perhaps only some *bookings*.
- Delete a *booking* request if required, provided [Permissions](#)³¹⁴ have been suitably configured, however doing so would mean the removal of the *booking* attempt from reports that might be created require.

To approve *bookings*, go to the [The Booking Requests Page](#)¹⁵¹ which shows all the *booking* requests and provides easy ways of approving them. It also offers a way to edit the *bookings*, if required.

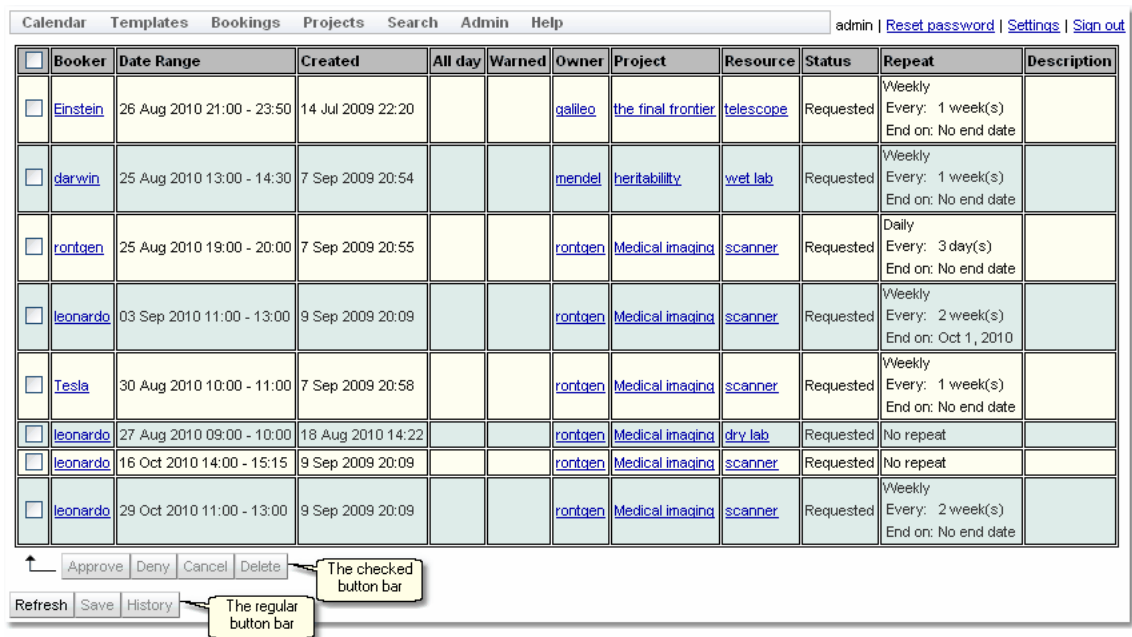
4.1.3 The Booking Requests Page

The **Booking Requests** page shows bookings whose status is **Requested**. By default, the **Booking Requests** page appears on the menu here:



However, the administrator may have configured **Calpendo** so that the menu is different.

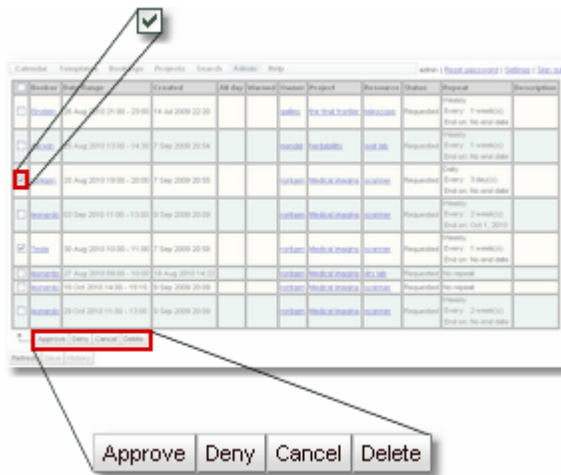
This is what the **Booking Requests** page looks like when there are some [bookings](#) ⁶⁵² awaiting approval:



Bookings will only appear in this page if they are repeatable or are currently in the future. Any request which is now in the past will not be viewed.

Approving Or Denying Bookings

As described in [The Booking Approval Process](#)¹⁵⁰, approving or denying *bookings* means changing their *status*⁶⁵³. The top part of the **Booking Requests** page provides a quick way to do just this. Tick the check box next to each *booking* whose *status* is needed to be changed.



Note that as soon as there are any *bookings* checked, then the checked button bar changes so that it is no longer greyed out.

Now press any of these buttons to do the appropriate function to the ticked *bookings*.

Alternatively, tick the check box in the table's header and every check box will be ticked (or unticked, as appropriate) as a short-cut to ticking all *bookings*.



Changing More Than A Booking's Status

The **Booking Requests** page can be used for more than just changing a *booking's status*. First, click anywhere in the *booking's* row apart from the check box. The *booking's* details will appear at the bottom of the page already in edit mode, and the values can be changed as required.

The screenshot shows the 'Bookings' tab in the Calpendo interface. A table lists several bookings. The booking for 'leonardo' on '03 Sep 2010 11:00 - 13:00' is selected. Below the table, a detailed form for this booking is displayed, showing fields for Booker, Date Range, Created, All day, Warned, Owner, Project, Resource, Status, Repeat, and Description.

Booker	Date Range	Created	All day	Warned	Owner	Project
<input type="checkbox"/> Einstein	26 Aug 2010 21:00 - 23:50	14 Jul 2009 22:20			galileo	the t
<input type="checkbox"/> darwin	25 Aug 2010 13:00 - 14:30	7 Sep 2009 20:54			mendel	herita
<input type="checkbox"/> rontgen	25 Aug 2010 19:00 - 20:00	7 Sep 2009 20:55			röntgen	Med
<input checked="" type="checkbox"/> leonardo	03 Sep 2010 11:00 - 13:00	9 Sep 2009 20:09			röntgen	
<input type="checkbox"/> Tesla	30 Aug 2010 10:00 - 11:00	7 Sep 2009 20:58			röntgen	
<input type="checkbox"/> leonardo	27 Aug 2010 09:00 - 10:00	18 Aug 2010 14:22			röntgen	M
<input type="checkbox"/> leonardo	16 Oct 2010 14:00 - 15:15	9 Sep 2009 20:09			röntgen	
<input type="checkbox"/> leonardo	29 Oct 2010 11:00 - 13:00	9 Sep 2009 20:09			röntgen	M

Buttons: Approve, Deny, Cancel, Delete

Form fields:

- Booker: leonardo (Leonardo Da V)
- Date Range: 03 Sep 2010 | 11:00 - 13:00
- Created: 9 Sep 2009 20:09
- All day: false
- Warned: Not warned
- Owner: röntgen (Wilhelm Röntgen)
- Project: imaging (Medical imaging)
- Resource: scanner
- Status: Requested
- Repeat: Weekly, Every: 2 week(s), End on: Selected date Oct 1, 2010
- Description:

Make changes the *booking properties* ⁶⁵⁵ as required, including changing the status using the status drop-down:

The screenshot shows the same booking details form as before, but with the 'Status' dropdown menu open. The menu lists the following options: Requested, Please select a Status, Requested, Approved, Denied, and Cancelled. The 'Requested' option is currently selected.

Buttons: Approve, Deny, Cancel, Delete

Form fields:

- Booker: leonardo (Leonardo Da V)
- Date Range: 03 Sep 2010 | 11:00 - 13:00
- Created: 9 Sep 2009 20:09
- All day: false
- Warned: Not warned
- Owner: röntgen (Wilhelm Röntgen)
- Project: imaging (Medical imaging)
- Resource: scanner
- Status: Requested
- Repeat: Weekly, Every: 2 week(s), End on: Selected date Oct 1, 2010
- Description:

There is no difference between selecting a *status* here and using the buttons in the checked button bar.

When finished, press the **Save** button. A pop-up will appear briefly in the bottom right corner to indicate the *booking* was saved.

Booker	Date Range	Created	All day	Warned	Owner	Project	Resource	Status	Repeat	Description
<input type="checkbox"/> Einstein	26 Aug 2010 21:00 - 23:50	14 Jul 2009 22:20			galileo	the final frontier	telescope	Requested	Weekly Every: 1 week(s) End on: No end date	
<input type="checkbox"/> darwin	25 Aug 2010 13:00 - 14:30	7 Sep 2009 20:54			mendel	bentability	wet lab	Requested	Weekly Every: 1 week(s) End on: No end date	
<input type="checkbox"/> rontgen	25 Aug 2010 19:00 - 20:00	7 Sep 2009 20:55			rortgen	Medical imaging	scanner	Requested	Daily Every: 3 day(s) End on: No end date	
<input checked="" type="checkbox"/> leonardo (Leonardo Da Vinci)	03 Sep 2010 11:00 - 13:00	9 Sep 2009 20:09			rortgen (Wilhelm Rontgen)	imaging (Medical imaging)	scanner	Approved	Weekly Every: 2 week(s) End on: Oct 1, 2010	
<input type="checkbox"/> Tesla	30 Aug 2010 10:00 - 11:00	7 Sep 2009 20:58			rortgen	Medical imaging	scanner	Requested	Weekly Every: 1 week(s) End on: No end date	
<input type="checkbox"/> leonardo	27 Aug 2010 09:00 - 10:00	18 Aug 2010 14:22			rortgen	Medical imaging	dry lab	Requested	No repeat	
<input type="checkbox"/> leonardo	16 Oct 2010 14:00 - 15:15	9 Sep 2009 20:09			rortgen	Medical imaging	scanner	Requested	No repeat	
<input type="checkbox"/> leonardo	29 Oct 2010 11:00 - 13:00	9 Sep 2009 20:09			rortgen	Medical imaging	scanner	Requested	Weekly Every: 2 week(s) End on: No end date	

Approve Deny Cancel Delete

Refresh Save History

Booker	leonardo (Leonardo Da Vinci)
Date Range	03 Sep 2010 11:00 - 13:00
Created	9 Sep 2009 20:09
All day	false
Warned	Not warned
Owner	rortgen (Wilhelm Rontgen)
Project	imaging (Medical imaging)
Resource	scanner
Status	Approved
Repeat	Weekly Every: 2 week(s) End on: Selected date Oct 1, 2010
Description	

Saved ok

Please note that although we've now changed this *booking's status*, it still appears in the table of *booking requests*. However, the *status* shown in the table has changed. A *booking* will remain in the **Booking Requests** page until the **Refresh** button is pressed, or the user logs out and logs back in again, or refresh's the browser.

4.1.4 Read Only Calendar

There is a [Workflow](#)³³⁴ as part of the standard system which allows the display of a Read Only Calendar to a web browser.

The **Workflow** is disabled by default. The minimum required to get it working is:

- Go to the Workflow Manager
- Find the Workflow, under "Booking" and called "Weekly Calendar"
- Edit it and enable the Workflow, and save it
- Point a web browser at:

```
• https://yourcalpendo/anon/bookings.html?title=Your Title&pks=1,2,3,4
```

where the arguments can be changed in the URL to control the displayed title and the IDs of the resources whose bookings should be displayed.

An example of this can be seen by going to:

```
https://demo.calpendo.com/anon/bookings.html?title=Custom%20Calendar&pks=1,2,3,5,6
```

The displayed data will reflect the permissions of a particular configured user, and that user is by default "admin". This may not be what is required, so if the calendar is to be used, it should be configured to make sure it's not showing information you don't want to show.

4.2 Project Administration

When somebody first creates a [project](#)⁶⁵⁴ (as described by [Creating A New Project](#)⁷⁵), the *project* is created with a [status](#)⁶⁵⁴ of **Requested**. While a *project* has this *status*, nobody can book against it. An administrator must approve the *project* by changing its *status* to **Approved** before it becomes bookable. This and other project-related administration is described in this section.

4.2.1 Project Properties

A [Project](#)⁶⁵⁴ in **Calpendo** has at least the following [properties](#)⁶⁵⁵:

Property Name	Description														
Project Code	This should be a short, unique code for the <i>project</i> . Calpendo will make sure that this is not the same as any other <i>project</i> (unless it's empty, as would be typical of a new <i>project</i> before an administrator assigns a code).														
Type	This is the project's type ⁶⁵⁵ . See Types And Groups ²⁷⁹ to understand how types are used. It is possible that Calpendo has been configured so that there are no <i>project types</i> . If this is the case, then it ought to have been configured so that the <i>project's type</i> property is not visible. See Property Visibility for more details.														
Status	<p>A project status⁶⁵⁴ has one of the following values:</p> <table> <tr> <th>Name</th><th>Description</th></tr> <tr> <td>Requested</td><td>This is the status used when a <i>project</i> is first created. bookings⁶⁵² may not be created for a <i>project</i> that is in this state.</td></tr> <tr> <td>Restricted</td><td>This status means the <i>project</i> is bookable by the Resource Manager and Admins, but not by Project Users or Owner.</td></tr> <tr> <td>Approved</td><td>This status means that the <i>project</i> is bookable.</td></tr> <tr> <td>Denied</td><td>This status indicates that it was decided the <i>project</i> should not be allowed, for whatever reason. <i>Projects</i> in this state are not bookable.</td></tr> <tr> <td>Terminated</td><td>This status indicates that the <i>project</i> has been used for making <i>bookings</i>, but that the <i>project</i> has now come to an end. A terminated <i>project</i> is not bookable.</td></tr> <tr> <td>Unbookable</td><td>This status indicates the <i>project</i> is not bookable for some other reason. Typically, the Project Template²¹⁸ is given this status.</td></tr> </table>	Name	Description	Requested	This is the status used when a <i>project</i> is first created. bookings ⁶⁵² may not be created for a <i>project</i> that is in this state.	Restricted	This status means the <i>project</i> is bookable by the Resource Manager and Admins , but not by Project Users or Owner .	Approved	This status means that the <i>project</i> is bookable.	Denied	This status indicates that it was decided the <i>project</i> should not be allowed, for whatever reason. <i>Projects</i> in this state are not bookable.	Terminated	This status indicates that the <i>project</i> has been used for making <i>bookings</i> , but that the <i>project</i> has now come to an end. A terminated <i>project</i> is not bookable.	Unbookable	This status indicates the <i>project</i> is not bookable for some other reason. Typically, the Project Template ²¹⁸ is given this status.
Name	Description														
Requested	This is the status used when a <i>project</i> is first created. bookings ⁶⁵² may not be created for a <i>project</i> that is in this state.														
Restricted	This status means the <i>project</i> is bookable by the Resource Manager and Admins , but not by Project Users or Owner .														
Approved	This status means that the <i>project</i> is bookable.														
Denied	This status indicates that it was decided the <i>project</i> should not be allowed, for whatever reason. <i>Projects</i> in this state are not bookable.														
Terminated	This status indicates that the <i>project</i> has been used for making <i>bookings</i> , but that the <i>project</i> has now come to an end. A terminated <i>project</i> is not bookable.														
Unbookable	This status indicates the <i>project</i> is not bookable for some other reason. Typically, the Project Template ²¹⁸ is given this status.														
Owner	This is the user that owns the <i>project</i> . Normally, this would be the user that created the <i>project</i> , but an administrator could change the ownership of a <i>project</i> .														
Name	This is the name of the project, and is intended to be a short description of it														
Description	This is a longer description of the <i>project</i> .														
Project Resource Settings	This is a list of settings, for each resource ⁶⁵⁵ , that give per- <i>project</i> and per- <i>resource</i> information. This is described further below.														
Project Service Settings	This is a list of the services available to the project.														

Property Name	Description
Users	This is a list of the users that are associated with the <i>project</i> . Only users associated with a <i>project</i> may book against that <i>project</i> . The exceptions to this rule are: 1) a user with the Admin role ⁶⁵⁶ may book for any <i>project</i> . 2) a user in a Resource Managers group can book for any <i>project</i> (for their resources only)
Project Groups	Which project groups the project is a part of.

These *properties* represent a reasonable minimum that would be required to get started, but any facility that makes good use of *projects* will be likely to want many more *properties*. **Calpendo** may well have additional *properties* configured for its *projects*. See the [Bakery](#)⁶³⁷ for the details of how to configure *properties* on a *project*.

A *Project Resource Settings* has the following properties:

Property Name	Description
Resource	This is the resource this Project Resource Settings ⁶⁵⁴ pertains to
Project	The <i>Project</i> this Resource Settings belongs to
Number of Sessions	This is the total number of <i>bookings</i> that are expected to be used by this <i>project</i> for this <i>resource</i> .
minutesPerSession	This is the expected average duration of each <i>booking</i> for this project on this <i>resource</i> .
costPerSession	This is the expected average cost of each <i>booking</i> for this project on this <i>resource</i> .
costPerHour	This is the cost to be charged for each hour.

Again, as for *projects*, your **Calpendo** may have additional *properties* configured for *Project Resource Settings*. Such extra *properties* could be used in [Booking Rules](#)⁶⁵² or in generating reports.

4.2.2 The Project Approval Process

The process of approving new [projects](#)⁶⁵⁴ is highly configurable in **Calpendo** (see [Configuring The Project Approval Process](#)²²⁰ for details). This should be configured in a manner that makes sense for your facility. The basic options are whether to have a single-step or a multi-step approval process.

Single-Step Project Approval Process

A single-step approval process is appropriate when there is a single person who decides whether a *project* should go ahead, or the decision is made in a single meeting. Under these circumstances, a *project* changes from its **Requested** state to either **Approved** or **Denied** in a single step. The project's status [property](#)⁶⁵⁵ is sufficient to handle this, and as far as **Calpendo** is concerned, the approval process is merely a matter of somebody (probably an administrator) changing the [project's status](#)⁶⁵⁴.

An administrator has a choice of how to approve *projects* using the single-step approval process. The easiest and preferred option is to use [The Project Requests Page](#)¹⁵⁹ which shows you just the *projects* whose status is **Requested**. It also lets you modify those *projects* and provides short-cuts to approving or denying new *project* requests. Alternatively, follow the instructions in [Modifying Projects](#)⁷⁸ and change the *project status* as required, or use the [Data Explorer](#)¹³⁹ to locate the *projects* to be approved.

There may also be an Automatic Email that tells administrators when there's a *project* that needs approving. If there is, then the email could include a direct HTML link that will take the administrator to a page with the *project* already loaded - making approving it very quick.

Restricted

Restricted *projects* can only be booked by users with the admin role or by users in a **Resource Managers** [user group](#)⁶⁵⁶ (and for the resources they manage only). This is useful for initial testing of the project and pilot studies.

Multi-Step Project Approval Process

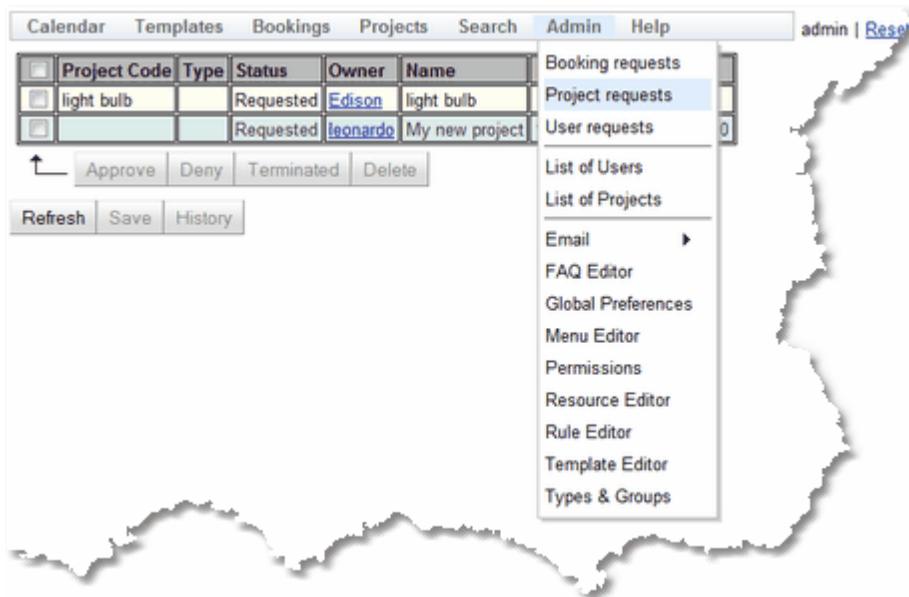
A multi-step approval process is appropriate when there are multiple people who must approve each *project*, and these people make their decisions at different times. To cater for this process, the *project* should be configured with additional *properties* to store the information about whether each person has given their approval. The [Permissions](#)³¹⁴ can be configured to control who can set these approval *properties*, and even the order in which they must be given. [Workflows](#)³³⁴ can be configured to send the next person in the chain an email indicating it's time for them to approve a *project*. The detail of this process is described in [Configuring The Project Approval Process](#)²²⁰.

This all means that approving a *project* can mean quite different things, depending on how **Calpendo** has been configured. However, the multi-step [project approval process](#)⁶⁵⁴ can always be characterised as some people modifying *properties* on the *project* while its status is **Requested** and then finally somebody changing the *project's status* to **Approved** or **Denied**. This means the *project approval process* is a matter of all these approvers editing the *project* in turn, and then somebody edits the *project* to change its status.

[The Project Requests Page](#)¹⁵⁹ is still the preferred way of achieving each of these changes since this page provides a convenient means of locating the requested *projects*, and the page allows modifications to be made to the *projects* in exactly the same manner as the other methods, as described by [Modifying Projects](#)⁷⁸ or using [Data Explorer](#)¹³⁹. However, the multi-stage approval process might send **Workflows** with a nested HTML link directly to the *project* in question. If that is done, then locating the requested *project* does not become an issue and so the email could include a link to the **List of Projects** page or [Data Explorer](#)¹³⁹.

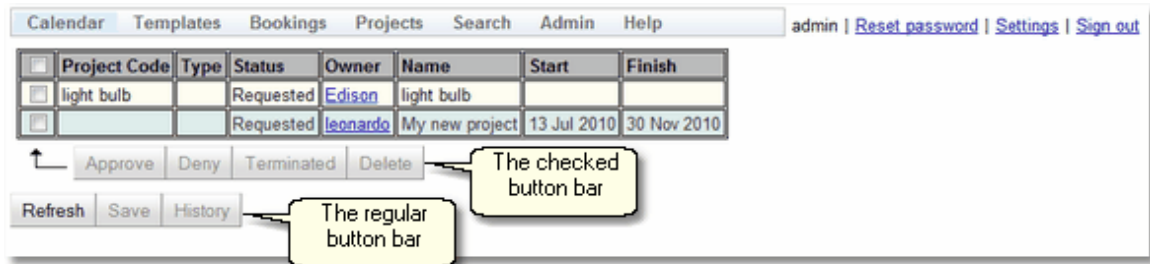
4.2.3 The Project Requests Page

The **Project Requests** page shows you [projects](#)⁶⁵⁴ whose status is **Requested**. By default, the **Project Requests** page appears on the menu here:



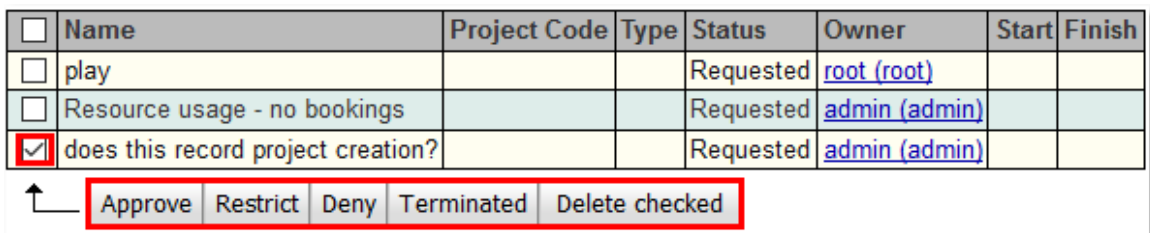
However, the administrator may have configured **Calpendo** so that the menu is different.

This is what the **Project Requests** page looks like when there are some *projects* awaiting approval:



Approving or Denying Projects

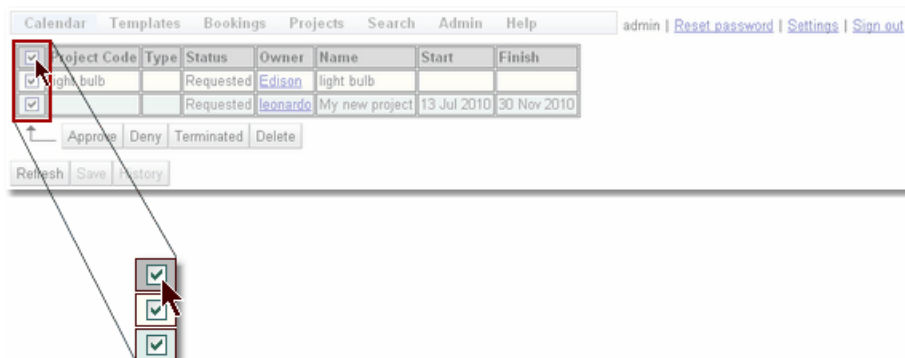
As described in [The Project Approval Process](#)¹⁵⁸, approving or denying new *projects* means changing their status, and possibly changing custom [properties](#)⁶⁵⁵ that have been added to a *project*. If using the [single step project approval process](#)¹⁵⁸, or needing to perform the last step of the [multi-step project approval process](#)¹⁵⁹, then use the top part of the **Project Requests** page to do just this. First, tick the check box next to each *project* whose status requires changing.



Note that as soon as there are any *projects* checked, then the checked button bar changes so that it is no longer greyed out.

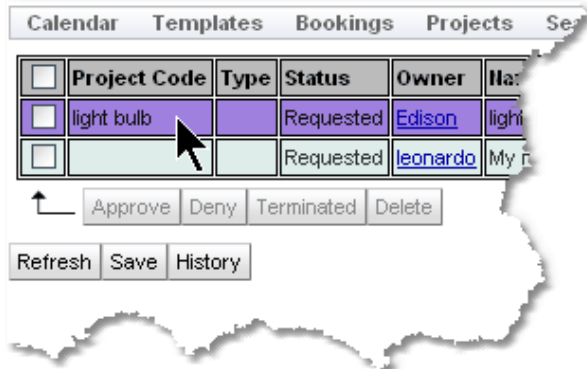
Now press any of the buttons to apply the function to the ticked *projects*.

Alternatively, tick the check box in the table's header and every check box will be ticked (or unticked, as appropriate) as a short-cut to ticking all *projects*.



Changing More Than A Project's Status

The **Project Requests** page can be used for more than just changing a [project's status](#)⁶⁵⁴. First, click anywhere in the *project's* row apart from the check box.

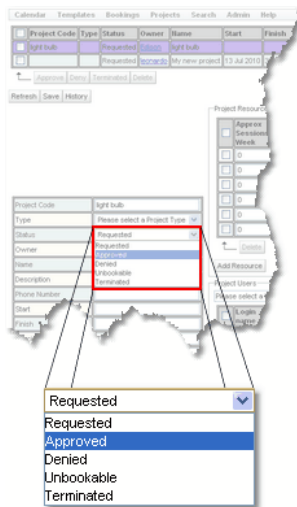


Selecting a single project to edit or view its details

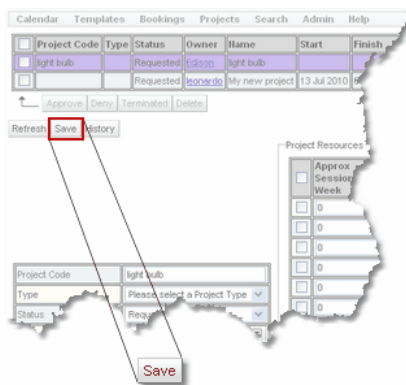
The *project's* details will appear at the bottom of the page already in edit mode, and you can change the values as required. The *projects* may have been configured with different *properties* from those shown in this picture, so may look different.

The screenshot shows the 'Project Details' form. At the top is a table with columns: 'Project Code', 'Type', 'Status', 'Owner', 'Name', 'Start', and 'Finish'. The first row has a checkbox, an empty cell, 'Physics Projects', 'Requested', 'curie (Marie Curie)', 'Silicon chips', and empty cells. Below the table are buttons: 'Approve', 'Deny', 'Terminated', 'Delete', 'Refresh', 'Save', and 'History'. The main form area has a left sidebar with a tree view containing 'General', 'Project Resource Settings', 'Project Service Settings', 'Users', and 'Project Groups'. The 'General' tab is selected, showing a form with the following fields: 'Project Code' (text input), 'Type' (dropdown menu with 'Physics Projects' selected), 'Status' (dropdown menu with 'Requested' selected), 'Owner' (dropdown menu with 'curie (Marie Curie)' selected), 'Name' (text input with 'Silicon chips'), 'Description' (text input), 'Phone Number' (text input), 'Start' (text input), 'Finish' (text input), 'Principal Investigator' (text input), 'Ethics Approval Number' (text input), 'Funding Agency' (text input), and 'Account Number' (text input).

The *project property* values can be changed as required, including changing the status using the status drop-down:

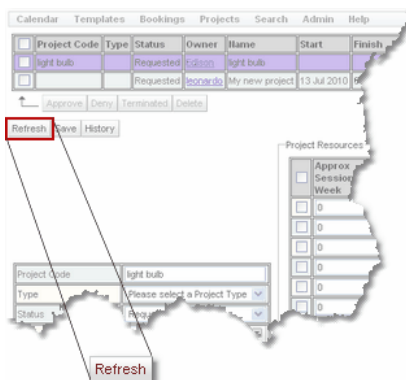


When finished, press the **Save** button.



A pop-up will appear briefly in the bottom right corner to indicate the *project* was saved.

Please notice that although the *project's status* has now changed, it still appears in the table of *project requests*. However, the status shown in the table has changed. A *project* will remain in the **Project Requests** page until the **Refresh** button is pressed, or the user logs out and logs back in again, or refreshes the browser.



4.3 User Administration

When somebody first registers with **Calpendo** (as described by [Getting A User Account](#)²⁸), a new user is created with a status of **Requested**. While a user has this status, they are unable to log in. An administrator must then approve the user by changing their status to **Approved**. This and other user-related administration is described in this section.

4.3.1 User Properties

This chapter describes the properties of a user and what they are for, but not how to change them. See [Modifying Users](#)¹⁷⁷ for a description of how to change the [properties](#)⁶⁵⁵ on a user. The licence for **Calpendo** counts the users with status **Normal** (whose accounts have not expired) and **Password must be reset at next login**.

Identity

Every user has a combination of authentication method and login name which must be unique. The login name may contain letters, numbers, hyphen, underscore or full stop. A login name may be changed as long as the new name adheres to these constraints. We store both a login name and a login identifier. Login identifiers can be long and complex, so the login name can be used in these cases instead of the identifier for logging in and display of the user information.

Roles

There can be up to 32 [roles](#)⁶⁵⁶ defined in the system, including the predefined and special **Root** and **Admin** roles. A user with the **Root** role always has *permission* to do anything in the system. A user with the **Admin** role is allowed to perform administrative tasks. However, **Calpendo** can be configured so that the **Admin** role is not required for administrative tasks. The remaining 30 possible roles can be configured as required, and they are normally associated with setting up *Permissions* as required for your facility. See [User Roles](#)⁵⁵⁴ in the [Bakery](#)⁵³⁷ section of the **Calpendo Configuration Guide** for how add new *roles*.

Users with the **Root** role have the following special properties:

- *Permissions* are not checked. *Permission* is always granted.
- The [booker](#)⁶⁵² of a [booking](#)⁶⁵² can be changed. Normally, the *booker* is always the user that first created the *booking* and only users with the **Root** role can break that.
- They are allowed to log in, even while the licence has expired. This allows you to recover such situations once you have a new licence.
- They can change the [status](#)⁶⁵⁶ of a user to make them active (for example, by changing the status from **Blocked** to **Normal**), even if it means there will be more active users than the licence allows. However, this may not be a good thing to do since only **Root** would then be able to use the system.
- Normally, a user can only allocate other users with roles that they already have. However, if a user has the **Root** role, they can add any role to another user.

Users with the **Admin** role have the following special properties:

- By default, *Permissions* are created that allow a user with the **Admin** role to **create**, **update** and **delete** almost all *Biskit Types*⁶⁵². They can also update the database schema when using the *Bakery*⁵³⁷. However, these *Permissions* are changeable and may be added, removed or modified for any user by a user with **Admin** privileges.
- *Time Templates*⁶⁵⁶ may not apply to users with the **Admin** role, depending on the *global preferences* setting [Templates -> Templates Apply To Admin](#)⁵³⁵
- *Booking Rules* may not apply to users with the **Admin** role, depending on the *global preferences* setting [Booking Rules -> Booking Rules Apply To Admin](#)⁵³¹
- When creating a *booking*, the default status assigned is specified by the *global preferences* setting [Bookings -> Admin Default Booking Status](#)⁵¹⁵ whereas for regular users it is decided by [Bookings -> User Default Booking Status](#)⁵¹⁵.
- They can create *bookings* against any *project*, whereas regular users can only book against *projects* whose status is **Approved** and *projects* for which they appear in the *project's* users selection.
- When the **Calpendo** licence has fewer than 30 days remaining, users with the **Admin** role will receive a warning each time they log in. Regular users only receive a warning in the 7 days prior to expiration. Note that 30 days' grace period is allowed, so that there is always time after expiration before the licence must be renewed.
- When a user with the **Admin** role logs in for the very first time, they are assigned the menu that has been configured for **Admin** users in [Global Preferences -> Menus -> Default Admin Menu](#)⁵²⁷. A user that has neither the **Root** nor the **Admin** role will be assigned the menu specified in [Menus -> Default User Menu](#)⁵²⁷ when they first log in.

The final difference that users with the **Root** or **Admin** role may notice is that each *Biskit Type* can be configured in the *Bakery*⁵³⁷ to be visible to users with the **Root** role, users with the **Admin** role, everybody or nobody. For example this provides a way to reduce the number of *biskit types* that users may search for, for example.

Names And Email

A user's name is split into three parts: their given name (usually their first name), the family name (usually their last name) and their other name (for any middle names). **Calpendo** uses these names for display and in reports, but not for anything else. For example, it is not a requirement that a user's name is unique.

A user's email address is important because **Calpendo** sends emails for various reasons ([Email Workflow Action](#)³⁷³ and [Manual Emails](#)¹⁸⁶).

Type

Calpendo may be configured so that users are asked for a *type*⁶⁵⁶ when they first register. The *type* is a means of segregating users into non-overlapping sets, which may be used by the facility for assigning *Permissions*⁶⁵⁴, *Booking Rules*⁶⁵², and *Time Templates*⁶⁵⁶. The possible values for a *user type* are configured in [Configuring Types And Groups](#)²⁸⁰.

Password

The users password is stored here, but will only be shown as a number of dots for security reasons. If there is no password the box will be empty.

Status

A [user's status](#)⁶⁵⁶ indicates whether they can log in. It can take the following values:

Status	Description
Requested	This is the status given to a user when they first register. A user whose status is Requested cannot log in.
Normal	This is the normal status for a user who can log in.
Password must be reset at next login	A user whose status is Password must be reset at next login can log in, but they will be forced to change their password as soon as they do so. If an administrator needs to reset a user's password, then it's useful also to set their status to this value at the same time.
Blocked	A user whose status is Blocked cannot log in. Typically, use this status for a user that needs to be stopped from logging in for some reason.
Denied	A user whose status is Denied cannot log in. Typically, use this status when not approving a new user request, although there is also have the option of deleting the user.
Expired	A user whose status is Expired cannot log in. Either their expiry date has passed or the administrator has decided they no longer need to use the system.
Lurker	A user whose status is Lurker cannot login but will receive automatic and booking reminder e-mails.
Suspended	A user which doesn't count towards the user limit, they can login, but whilst suspended do not receive any emails. Their status is automatically changed to Normal on login.

The reason that both **Blocked** and **Denied** statuses exist is so that an [Email Workflow Action](#)³⁷³ can be used to send the affected user a message that is suitable for the situation.

Expiry Date

This allows the administrator to set an expiry date for the users login. When this field is edited a calendar will appear, just select the day the login is to expire on.

Last Login and Last Login From

This records the last date and time the user logged in and the IP address of the machine they logged in from.

Groups

A user may belong to any number of [user groups](#)⁶⁵⁶, and group membership can be used for assigning *Permissions*, *Booking Rules* and *Time Templates*. The *groups* that exist are configured in [Configuring Types And Groups](#)²⁸⁰.

4.3.2 The User Approval Process

When users first register, their [status](#)⁶⁵⁶ is set to **Requested**. While a *user's status* is **Requested**, they will not be able to log in. Approving new users, and thereby allowing them to log in, simply means changing their *status* to **Normal**. Deny the user request by changing their *status* to **Denied** or delete the user. It is also useful to define the users [roles](#)⁶⁵⁶ at this point. A new user can be assigned any *role* the person doing the assigning has. User approval can be done in the following places:

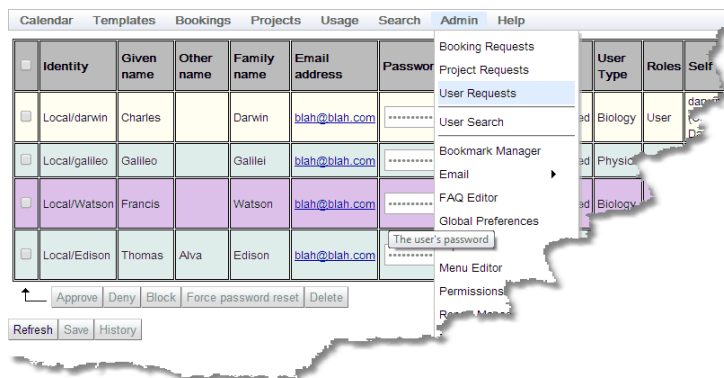
- [The User Requests](#)¹⁶⁷ [page](#)¹⁶⁷ shows just the users whose status is **Requested**. It also provides short-cuts to approving or denying user requests and allows for user detail modification.
- [Modifying Users](#)¹⁷⁷ describes how to use the other options for modifying users. That is, the **User Search** page and the [Data Explorer](#)¹³⁹.

By default, **Calpendo** has an Automatic Email configured that will tell all administrators when there is a new user request. The email received contains links to approve or deny the new user request. Just click on the links in the email to approve or deny new users.

Please note that if a [resource](#)⁶⁵⁵ requires a [project](#)⁶⁵⁴ to be entered when a [booking](#)⁶⁵² is made, a user cannot book that *resource* unless they are assigned to the *project* being used to make the *booking*.

4.3.3 The User Requests Page

The **User Requests** page shows users whose status is **Requested**. By default, the **User Requests** page appears on the menu here:



However, your administrator may have configured the menus differently.

This is what the **User Requests** page looks like when some users are awaiting approval:

Calendar

Templates

Bookings

Projects

Usage

Search

Admin

Help

User Requests

admin

Change Password

Settings

Sign out

<input type="checkbox"/>	Identity	Given name	Other name	Family name	Email address	Password	Status	User Type	Roles	Self	Expiry Date	Version	Created	Updated	Requested Project Code(s)
<input type="checkbox"/>	Local/darwin	Charles		Darwin	blah@blah.com	Requested	Biology	User	darwin (Charles Darwin)		1	5 Jun 2014 14:33	5 Jun 2014 14:33	
<input type="checkbox"/>	Local/galileo	Galileo		Galilei	blah@blah.com	Requested	Physics	User	galileo (Galileo Galilei)		1	5 Jun 2014 14:33	5 Jun 2014 14:33	
<input type="checkbox"/>	Local/Watson	Francis		Watson	blah@blah.com	Requested	Biology	User	Watson (Francis Watson)		1	5 Jun 2014 14:33	5 Jun 2014 14:33	
<input type="checkbox"/>	Local/Edison	Thomas	Alva	Edison	blah@blah.com	Requested	Physics	User	Edison (Thomas Edison)		1	5 Jun 2014 14:33	5 Jun 2014 14:33	

Approve

Deny

Block

Force password reset

Delete

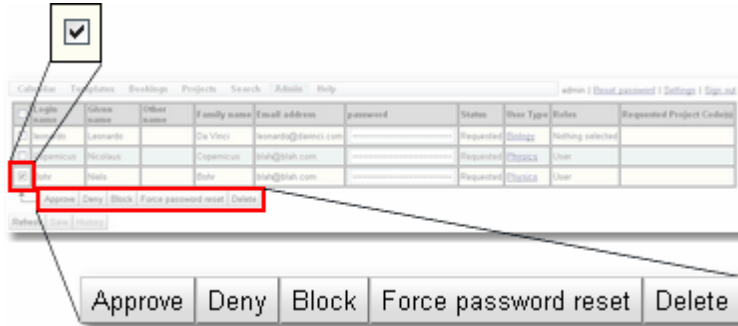
Refresh

Save

History

Approving Or Denying Users

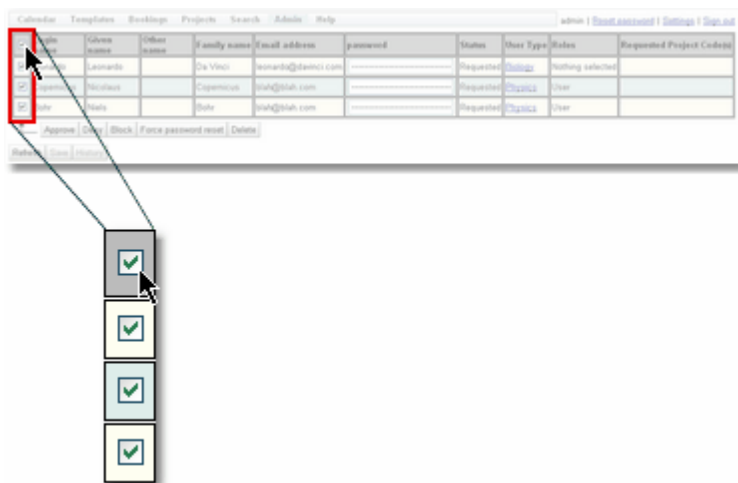
As described in [The User Approval Process](#)¹⁶⁷, approving or denying new users means changing their *status*⁶⁵⁶. The top part of the **User Requests** page provides a quick way to do just this. Tick the check box next to each user whose *status* to be changed.



Note that as soon as there are any users checked, then the checked button bar changes so that it is no longer greyed out.

Now press any of these buttons to **Approve**, **Deny**, **Block**, **Suspend** or **Delete** the ticked users. Press **Force password reset** to allow the users to log in, but force them to change their password.

Alternatively, tick the check box in the table's header and every check box will be ticked (or unticked, as appropriate) as a short-cut to ticking all users.



Changing More Than A User's Status

The **User Requests** page can be used for more than just changing a *user's status*. First, click anywhere in the user's row apart from the check box. The user's details will appear at the bottom of the page already in edit mode, and the values can be changed as required. It is useful to define the [users roles](#)⁶⁵⁶ at this point. A new user can be assigned any *role* the person doing the assigning has. If the assignee has **Admin**, **Guest** then they can assign **Admin** and **Guest** *roles* to another user.

<input type="checkbox"/>	Identity	Given name	Other name	Family name	Email address
<input type="checkbox"/>	Local/darwin	Charles		Darwin	blah@blah.co
<input type="checkbox"/>	Local/galileo	Galileo		Galilei	blah@blah.co
<input checked="" type="checkbox"/>	Local/Watson	Francis		Watson	blah@blah.co
<input type="checkbox"/>	Local/Edison	Thomas	Alva	Edison	blah@blah.co

Given name	Francis
Other name	
Family name	Watson
Email address	blah@blah.com
Password	*****
Status	Requested
User Type	Biology
Roles	User
Expiry Date	
Requested Project Code(s)	

Identity Projects Groups

Local Watson

Choose the *status* you want the user to have from the drop-down.

There is no difference between selecting a *status* here and using the buttons in the checked button bar.

To associate this user with some [projects](#)⁶⁵⁴, select the **Projects** tab at the bottom and select the *projects* you want to add.

Projects Groups

Please select a Project to add

Please select a Project to add

- genetics (heritability)
- imaging (Medical imaging)
- inventions (inventions)
- light bulb (light bulb)
- null (My new project)
- radiation (radiation)
- space (the final frontier)

After *projects* have been added, they are shown in the table.

Project Code	Type	Status	Owner	Name
imaging		Approved	rontgen (Wilhelm Rontgen)	Medical imaging

When finished, press the **Save** button. A pop-up will appear briefly in the bottom right corner to indicate the user was saved.

Refresh Save History

Roles: User

Given name: Galileo

Other name:

Family name: Galilei

Email address: blah@blah.com

User Type: Physics

Password:

Status: Requested

Expiry Date:

Requested Project Code(s):

Save

Saved

Although we have now changed this *user's status*, it still appears in the table of user requests. However, the *status* shown in the table has changed. A user will remain in the **User Requests** page until the **Refresh** button is pressed, or the browser is refreshed.

login name	Given name	Other name	Family name	Email address	password	Status	User Type	Roles	Requested Project Code(s)
blondie	blondie	Da Vinci	blondie	blondie@blondie.com		Requested	Physics		Nothing selected
Chemist	Chemist	Chemist	Chemist	Chemist@chem.com		Requested	Physics	User	
Blah	Blah	Blah	Blah	Blah@blah.com		Requested	Physics	User	

Refresh

Changing The User's Group Membership

To associate the user with some [users groups](#)⁶⁵⁶, first select the user's row, as shown above in [Changing More Than A User's Status](#)¹⁶⁹, and then click the **Groups** tab.

Requested Project Code(s)	
<input type="checkbox"/>	Name
<input checked="" type="checkbox"/>	20th Century
<input type="checkbox"/>	16th Century
<input type="button" value="Apply"/> <input type="button" value="Refresh"/>	

The check boxes indicate what *groups* this user is a member of. New users will not be a member of any *groups*, so no boxes are checked. Tick the check boxes next to the *groups* this user needs to be a member of.

Requested Project Code(s)	
<input type="checkbox"/>	Name
<input checked="" type="checkbox"/>	20th Century
<input type="checkbox"/>	16th Century
<input type="button" value="Apply"/> <input type="button" value="Refresh"/>	

Once all the required *user groups* are chosen, press the **Apply** button. Once done, a pop-up showing the number of user groups that have been changed will appear.

Requested Project Code(s)	
<input type="checkbox"/>	Name
<input checked="" type="checkbox"/>	20th Century
<input type="checkbox"/>	16th Century
<input type="button" value="Apply"/> <input type="button" value="Refresh"/>	

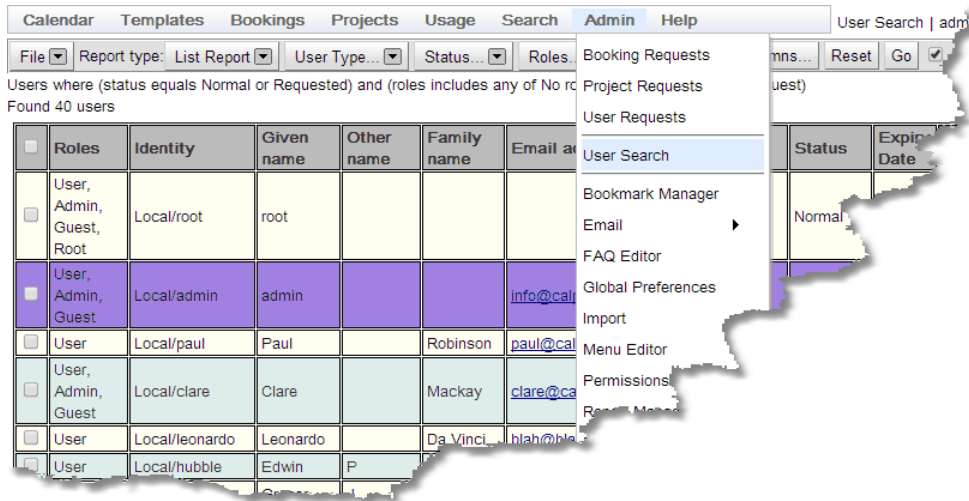
Updated 1 groups

The user does not need to save the user information if only the *group* memberships have changed. That's because *group* memberships are a [property](#)⁶⁵⁵ of the **User Group** and not the **User**, and it's also why there's a separate **Apply** button to save the *group* membership changes.

A number of users can be added or removed from a group at once using the options on the **Checked Menu** called **Add To Group** and **Remove From Group**. Once the users to be added/removed from a group are all checked, click the appropriate button and choose the group for the selected users to be added/removed.

4.3.4 User Search

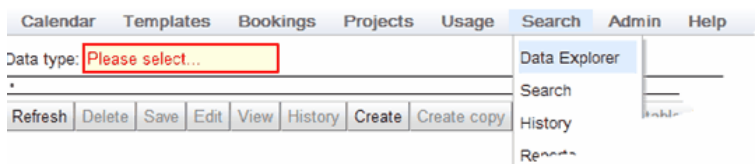
To search for users use the dedicated **User Search** page, which by default appears on the menu here:



The administrator may have configured your **Calpendo** so that the menu is different.

The alternative way to get a list of users is by using the [Data Explorer](#)¹³⁹ and choosing **User** as the [Biskit Type](#)⁶⁵². It doesn't make any difference which method is chosen - these pages behave almost identically apart from **User Search** being dedicated to users, while [Data Explorer](#)¹³⁹ can handle any *Biskit Type*.

By default, [Data Explorer](#)¹³⁹ appears on the menu here:



This is what the **User Search** page looks like when it is first displayed:

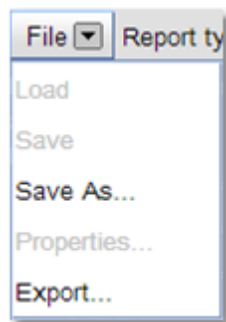
File	Report type: List Report	User Type...	Status...	Roles...	Conditions...	Columns...	Reset	Go	Autorun
------	--------------------------	--------------	-----------	----------	---------------	------------	-------	----	---------

Users where (status equals Normal or Requested) and (roles includes any of No roles, Root, Admin, User or Guest)
Found 18 users

	Roles	Identity	Given name	Other name	Family name	Email address	User Type	Status	Expiry Date	Version	Created	Updated
<input type="checkbox"/>	User, Admin, Guest, Root	Local/root	root					Normal		0		
<input type="checkbox"/>	User, Admin, Guest	Local/admin	admin			info@calpendo.com		Normal		0		
<input type="checkbox"/>	User	Local/paul	Paul		Robinson	paul@calpendo.com		Normal				
<input type="checkbox"/>	User, Admin, Guest	Local/clare	Clare		Mackay	clare@calpendo.com		Normal		0		
<input type="checkbox"/>	User	Local/leonardo	Leonardo		Da Vinci	blah@blah.com	Physics	Normal		0		
<input type="checkbox"/>	User	Local/hubble	Edwin	P	Hubble	blah@blah.com	Physics	Normal		0		
<input type="checkbox"/>	User	Local/mendel	Gregor	J	Mendel	blah@blah.com	Biology	Normal		0		
<input type="checkbox"/>	User	Local/roentgen	Wilhelm	K	Rontgen	blah@blah.com	Physics	Normal		0		
<input type="checkbox"/>	User	Local/newton	Isaac		Newton	blah@blah.com	Physics	Normal		0		
<input type="checkbox"/>	User	Local/darwin	Charles		Darwin	blah@blah.com	Biology	Requested		1	5 Jun 2014 14:33	5 Jun 2014 14:33
<input type="checkbox"/>	User	Local/curie	Marie		Curie	blah@blah.com	Physics	Normal		0		
<input type="checkbox"/>	User	Local/galileo	Galileo		Galilei	blah@blah.com	Physics	Requested		3	5 Jun 2014 14:33	5 Jun 2014 16:11
<input type="checkbox"/>	User	Local/Crick	Francis		Crick	blah@blah.com	Biology	Normal		0		
<input type="checkbox"/>	User	Local/Watson	Francis		Watson	blah@blah.com	Biology	Requested		2	5 Jun 2014 14:33	5 Jun 2014 16:02
<input type="checkbox"/>	User	Local/Einstein	Albert		Einstein	blah@blah.com	Physics	Normal		0		
<input type="checkbox"/>	User	Local/Edison	Thomas	Alva	Edison	blah@blah.com	Physics	Requested		1	5 Jun 2014 14:33	5 Jun 2014 14:33
<input type="checkbox"/>	User	Local/Tesla	Nikola		Tesla	blah@blah.com	Physics	Normal		0		
<input type="checkbox"/>	User	Local/Planck	Max		Planck	blah@blah.com	Physics	Normal		0		

↑ Edit Delete Approve Deny Block Force password reset

The following section goes through the options available after the search has been completed, enabling the search to be changed, saved or the data exported, as well as change the information viewed by the search. For more information on any of the features shown read the [Search](#)¹¹⁸ chapter.



Save As: Allows saving of the report for later use.

Export: Allows export of the information found in the search to a file.

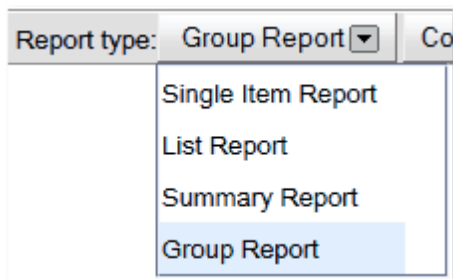
This button determines what type of report will be seen:

[Single Item Report](#)⁶⁵⁵: Displays a single record that matches the search.

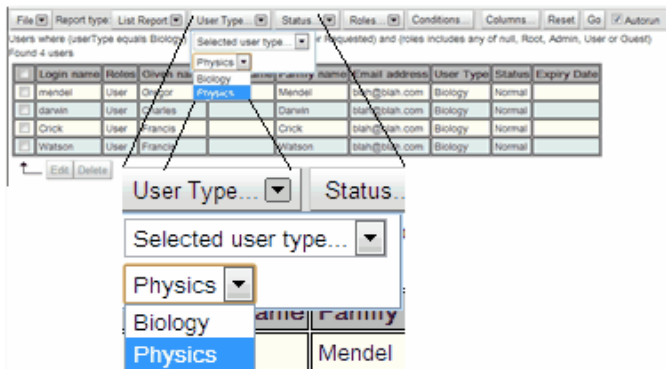
[List Report](#)⁶⁵³: Lists each record found that matches the search one under the other.

[Summary Report](#)⁶⁵⁶: Allows the information found to be summarised.

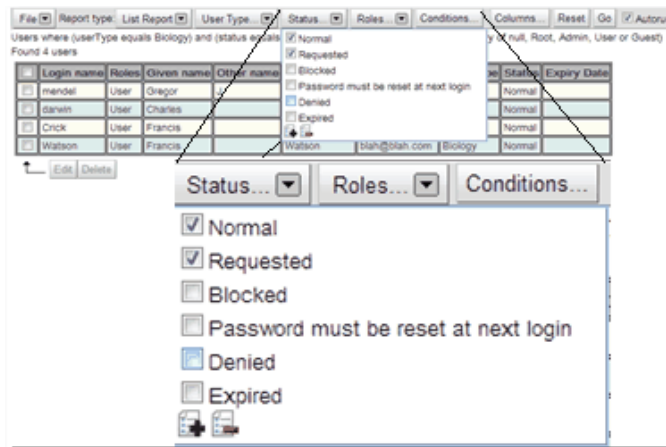
[Group Report](#)⁶⁵³: Allows the report to be shown by groupings with a count of the number of records that fit each group.



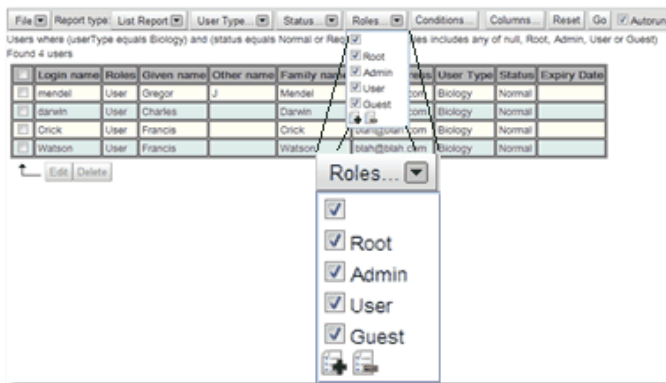
To change the search parameters



Choose **Selected User Type** then select a user type⁶⁵⁶ from the drop down options.

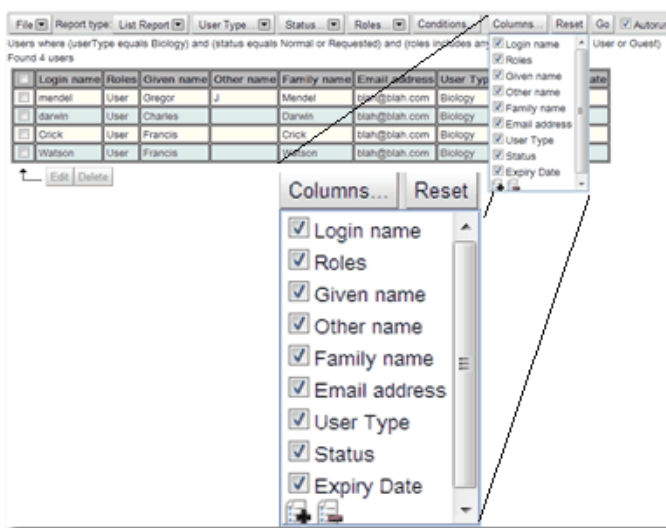
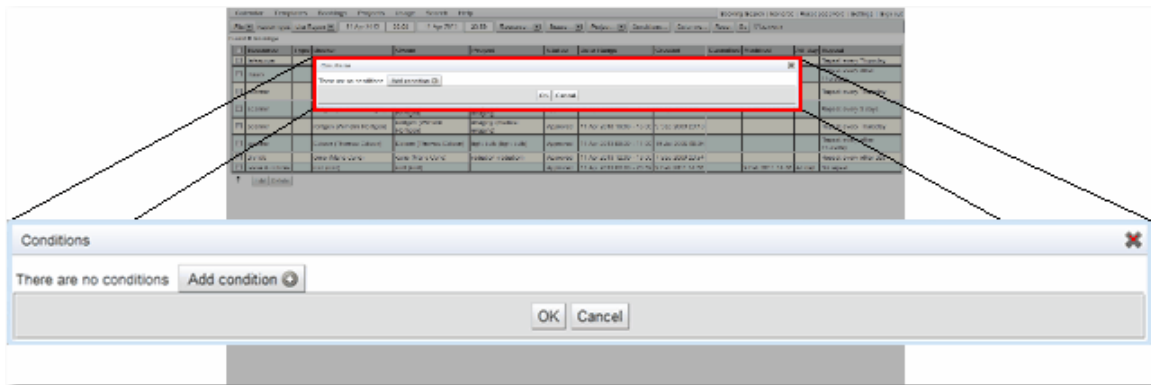


Then select the user status⁶⁵⁶ required.



And select the user roles⁶⁵⁶.

Set up additional [conditions](#)⁶⁵³ for the search. For more information on how to set up *conditions* read the information in the [Search](#)¹¹⁸ page.



And change the columns displayed in the output.

Reset: Resets the search *conditions* to the defaults set up on starting the search.

Go: Runs the search with the current *conditions*



Autorun: If this button is ticked then searches will run as soon as any *conditions* change. If the user will be changing a number of the *conditions* of the search, or searches will return large amounts of information it is more efficient if this is not ticked.

4.3.5 Creating Users

To add a new user that has not been requested go to the **User Search** page. There will be a list of the current users and underneath a menu bar with **Create** button enabled.

File Report type: List Report User Type... Status... Roles... Conditions... Columns... Reset Go ☒ Autorun

Users where (status equals Normal or Requested) and (roles includes any of No roles, Root, Admin, User or Guest)

Found 5 users

<input type="checkbox"/>	Roles	Identity	Given name	Other name	Family name	Email address	User Type	Status	Expiry Date	Version	Created	Updated
<input type="checkbox"/>	User, Admin, Guest, Root	Local/root	root					Normal		0		
<input type="checkbox"/>	User, Admin, Guest	Local/admin	admin			info@calpendo.com		Normal		0		
<input type="checkbox"/>	User	Local/leonardo	Leonardo		Da Vinci	blah@blah.com	Physics	Normal		0		
<input type="checkbox"/>	User	Local/hubble	Edwin	P	Hubble	blah@blah.com	Physics	Normal		0		
<input type="checkbox"/>	User	Local/mendel	Gregor	J	Mendel	blah@blah.com	Biology	Normal		0		

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

In order to understand more about the [properties](#)⁶⁵⁵ used for a user read the section on [User Properties](#)¹⁶³. The login name to be used will be added under identity after choosing the appropriate authentication method. Also in order to understand more about setting up [projects](#)⁶⁵⁴ and [groups](#)⁶⁵⁶ for a user read the section on [The User Requests Page](#)¹⁶⁷ and [Modifying Users](#)¹⁷⁷.

Cancel Save Apply

Roles	No roles
Given name	<input type="text"/>
Other name	<input type="text"/>
Family name	<input type="text"/>
Email address	<input type="text"/>
User Type	Please select a User Type ▼
Password	<input type="text"/>
Status	Please select a Status ▼
Expiry Date	<input type="text"/>
Requested Project Code(s)	<input type="text"/>

Login Name to be added here

4.3.6 Modifying Users

To see, change or delete users, go to a list of users. Either use the dedicated **User Search** page, or use [Data Explorer](#)¹³⁹ as explained in the [User Search](#)¹⁷² chapter to get the list.

Once the list of users has been returned they can be modified. Depending on which page was used to get the list of users either go to the [Data Explorer](#)¹³⁹ or [Search](#)¹¹⁸ pages for more details on editing and saving. Described below is how to specifically change a [user's project associations](#)⁶⁵⁶ and [user groups](#)⁶⁵⁶.

Changing A User's Password

Once the list of users is displayed, click on the user whose password needs to be changed.

<input type="checkbox"/>	Roles	Identity	Given name	Other name	Family name	Email address
<input type="checkbox"/>	User, Admin, Guest, Root	Local/root	root			
<input type="checkbox"/>	User, Admin, Guest	Local/admin	admin			info@calpendo.com
<input checked="" type="checkbox"/>	User	Local/leonardo	Leonardo		Da Vinci	blah@blah.com
<input type="checkbox"/>	User	Local/hubble	Edwin	P	Hubble	blah@blah.com
<input type="checkbox"/>	User	Local/mendel	Gregor	J	Mendel	blah@blah.com

[↑](#)
[Edit](#)
[Delete](#)
[Approve](#)
[Deny](#)
[Block](#)
[Force password reset](#)

[Edit](#)
[Create](#)
[Create copy](#)
[Delete](#)
[References](#)
[History](#)

Roles	User
Self	leonardo (Leonardo Da Vinci)
Given name	Leonardo
Other name	
Family name	Da Vinci
Email address	blah@blah.com
User Type	Physics
Password
Status	Normal
Expiry Date	
Version	0
Created	
Updated	
Requested Project Code(s)	

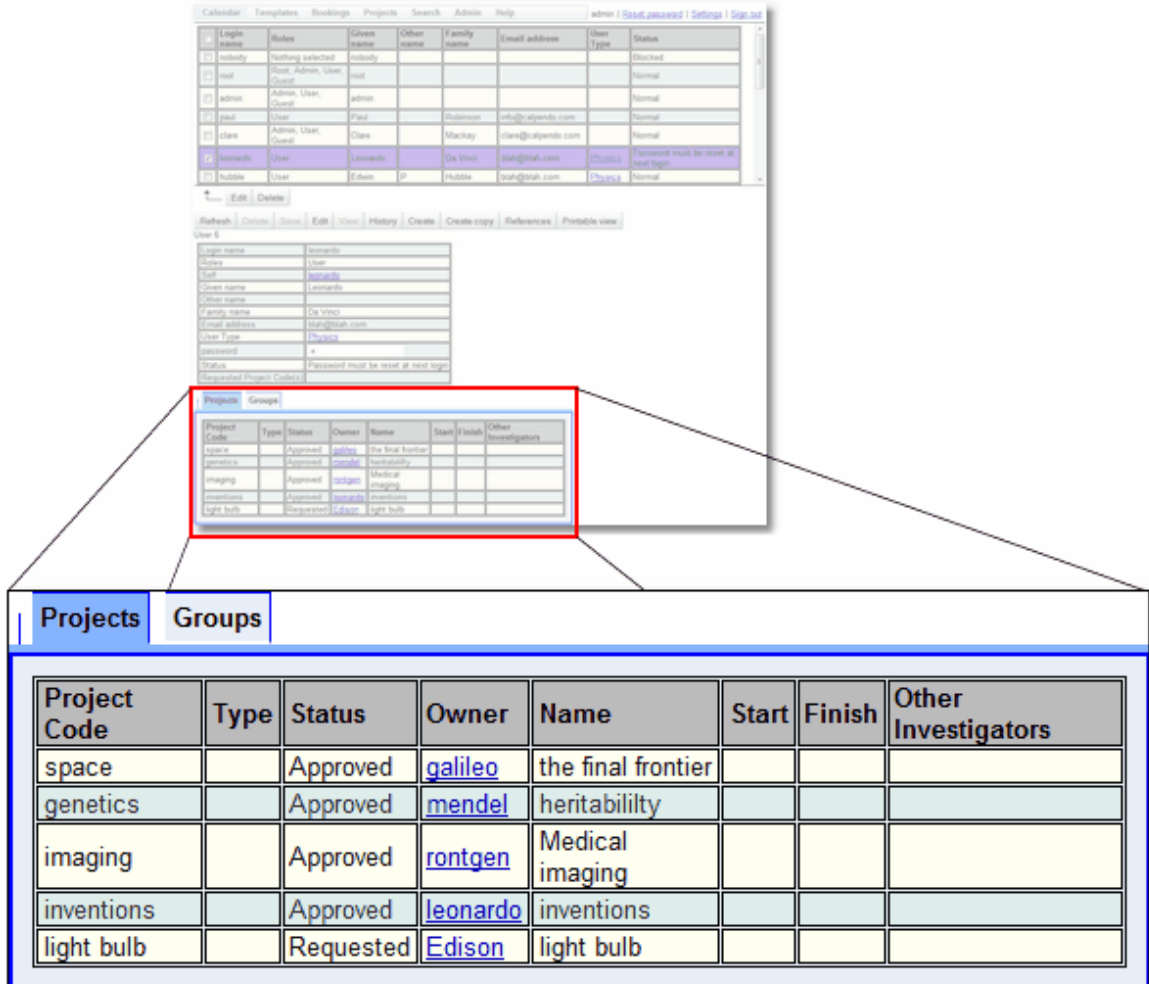
[Identity](#)
[Projects](#)
[Groups](#)

Local/leonardo

Click on the **Edit** button, once in edit mode any of the users details may be changed. If changing a users password it may be an idea to change the **Status** to **Password must be reset at next login**, in order that the user must change their password again when they log in. Once the details are changed click **Save** (this will have replaced the **Edit** button).

Changing A User's Project Associations

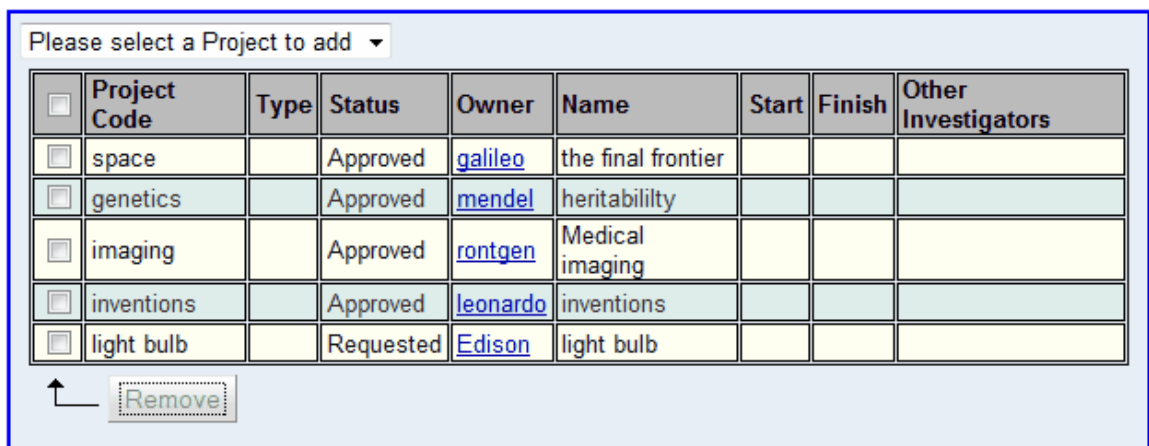
With a user's detail showing, as described above, the *user's project associations* are shown in the **Projects** tab at the bottom of the page.



The screenshot shows the Calpendo user interface. At the top, there is a navigation bar with tabs: Calendar, Templates, Bookings, Projects, Search, Admin, and Help. Below this, there is a table of users. The user 'Leonardo' is selected, and his details are shown in a form below the table. The 'Projects' tab is selected, and a table of projects associated with the user is displayed. A red box highlights the 'Projects' tab and the project table.

Project Code	Type	Status	Owner	Name	Start	Finish	Other Investigators
space		Approved	galileo	the final frontier			
genetics		Approved	mendel	heritability			
imaging		Approved	rontgen	Medical imaging			
inventions		Approved	leonardo	inventions			
light bulb		Requested	Edison	light bulb			

In edit mode, the **Projects** tab changes to this:



The screenshot shows the Calpendo user interface in edit mode for the 'Projects' tab. The 'Projects' tab is selected, and a table of projects is displayed. A 'Remove' button is visible at the bottom left.

Please select a Project to add ▼

<input type="checkbox"/>	Project Code	Type	Status	Owner	Name	Start	Finish	Other Investigators
<input type="checkbox"/>	space		Approved	galileo	the final frontier			
<input type="checkbox"/>	genetics		Approved	mendel	heritability			
<input type="checkbox"/>	imaging		Approved	rontgen	Medical imaging			
<input type="checkbox"/>	inventions		Approved	leonardo	inventions			
<input type="checkbox"/>	light bulb		Requested	Edison	light bulb			

Remove

This has its own checked button bar:

1. Tick the check boxes of any [project](#)⁶⁵⁴ to be removed.
2. The **Remove** button will change so that it is not greyed out once a check box is ticked.
3. Press the **Remove** button to remove the ticked projects from the user's list.

To add a new *project*, select it from the drop down. Note that the way the drop-down is formatted for adding new *projects* will change depending on the number of *projects* that exist. For a small number of *projects*, you get a simple drop down. When the number grows, then you get both a drop-down menu with all *projects*, but also a text box in which to type the *project* name. As the name is typed, it shows a selection of *projects* that match.

The *user's association of projects* is only changed once the **Save** button is pressed, in the regular button bar. Note that the *projects* themselves are also changed because they know which users are associated with them.

Changing A User's User Groups

To see which [user groups](#)⁶⁵⁶ a user is a member of select the user from the list of users, as described by [How To Edit A Single User](#)¹⁴² above. The tab panel at the bottom of the page shows the user's *projects* and *group* memberships. Select the **Groups** tab and a list of all *user groups* is seen, with a check box next to each group the user is a member of.

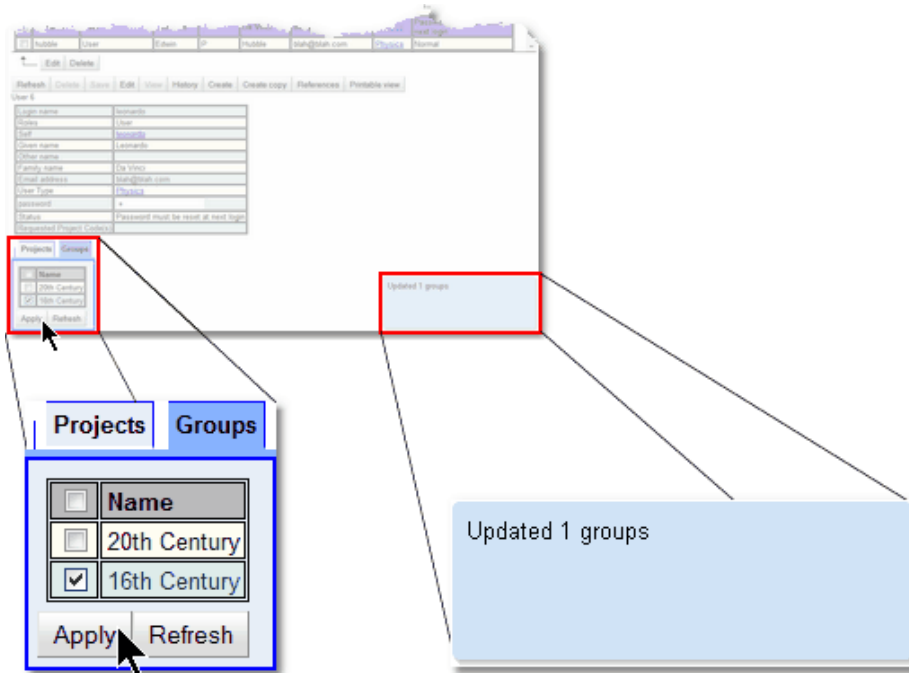
The screenshot shows a web interface for managing user groups. At the top, there is a text input field labeled 'Requested Project Code(s)' with a small dropdown arrow. Below this is a tabbed interface with two tabs: 'Projects' and 'Groups'. The 'Groups' tab is currently selected. Under the 'Groups' tab, there is a table with two columns: a checkbox column and a 'Name' column. The table contains two rows of data: '20th Century' and '16th Century'. Below the table are two buttons: 'Apply' and 'Refresh'.

	Name
<input type="checkbox"/>	20th Century
<input type="checkbox"/>	16th Century

Apply Refresh

Note: the user's details do not need to be in edit mode in order to change the *user group* membership. That's because the *user group* membership is not a *property* of the user itself, but of the *user groups*. To modify the *group* membership for the selected user:

1. Tick or untick the groups. The *groups* ticked are the ones the user is a member of. Tick the check box in the header row if all the check boxes are to be ticked or unticked.
2. Press the **Apply** button when finished.



Configuring The Properties Displayed

The user can control which [properties](#)⁶⁵⁵ are displayed in the list of items, in the detailed view of a item. This is done in the [Bakery](#)⁶³⁷.

- [Properties Visible In A Biskit List](#)⁶⁴⁷ explains how to change the *properties* that are displayed in a list of items. Change the visibility of each of the *properties* defined on an item.
- [Properties Visible In Biskit Detail](#)⁶⁴⁷ explains how to change the *properties* that are displayed when seeing a detailed view of an item. In this case, it affects both the read-only and editable mode detailed view of an item. Again, change the visibility of each of the *properties* defined on an item.
- [Properties Visible In A Collection Editor](#)⁶⁴⁷ explains how to change the *properties* that are displayed in collection editors. For example, if there is a *Project Biskit Type*⁶⁵² that contains a list of users, then the user can change which *properties* are shown to represent those users in the *Project's* list of users.

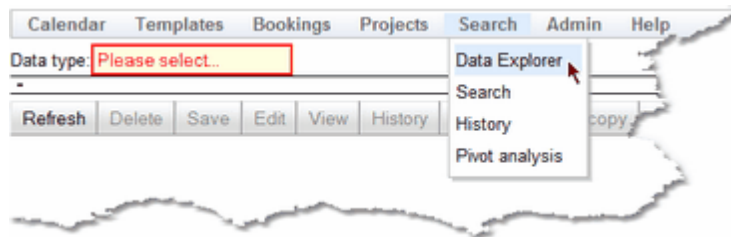
4.3.7 Changing A User's Settings

A user can change their own [settings](#)⁶⁵⁶, as described in [User Settings](#)³⁴ in the **Calpendo User Guide**. Sometimes, however, the administrator may want to edit another *user's settings*. This can be desirable if some users do not have [permission](#)⁶⁵⁴ to modify all of their *settings*. For example, **Calpendo** may have been configured so that a regular user cannot choose which menu they use, whereas an administrator may have *permission* to do so. The menu is one of the [properties](#)⁶⁵⁵ of a *user's settings*.



Page For Editing Another User's Settings

Currently, there is no page dedicated to editing other *user's settings*. To do this, use [Data Explorer](#)¹³⁹ and set the [Biskit Type](#)⁶⁵² to **Calpendo User Settings**. By default, [Data Explorer](#)¹³⁹ appears on the menu here:



4.3.8 Special User

There is only one special user, and that is **nobody**. This is a user whose [status](#)⁶⁵⁶ should be set to **Blocked** and never used. This user is used in the following circumstance:

Sending automated emails with embedded data:

When an [Email Workflow Action](#)³⁷³ is run to send an email, it can include data taken from some [Biskits](#)⁶⁵². When this happens, to make sure that it is safe to include such information, make sure that the information is only readable by whomever the email goes to, or whomever initiated the action that caused the email to be sent, as it is possible that anybody could read the email. It is therefore better to ensure that emails only include information that is readable by the special user **nobody**. If the special user **nobody** can read the data, then it's safe to put it into an email. Your administrator should have set up which data **nobody** has [Permission](#)³¹⁴ to read.

If a [Permission](#)³¹⁴ is set up to stop users from running reports, then **nobody** must be in the **Does Not Apply To** section in order for scheduled reports to run.

Please note that, while there are users created by default that are called **Root** and **Admin**, these are **not** special users. They have a [user role](#)⁶⁵⁶ of **Root** and **Admin** respectively, and their *roles* are special, but the users are not. As many users with the **Root** role may be created as required. Any user with the **Root** role is not subjected to [permissions](#)⁶⁵⁴ checks, regardless of whether it is the user **Root** or not; only the *roles* matter.

4.4 Bookmark Manager

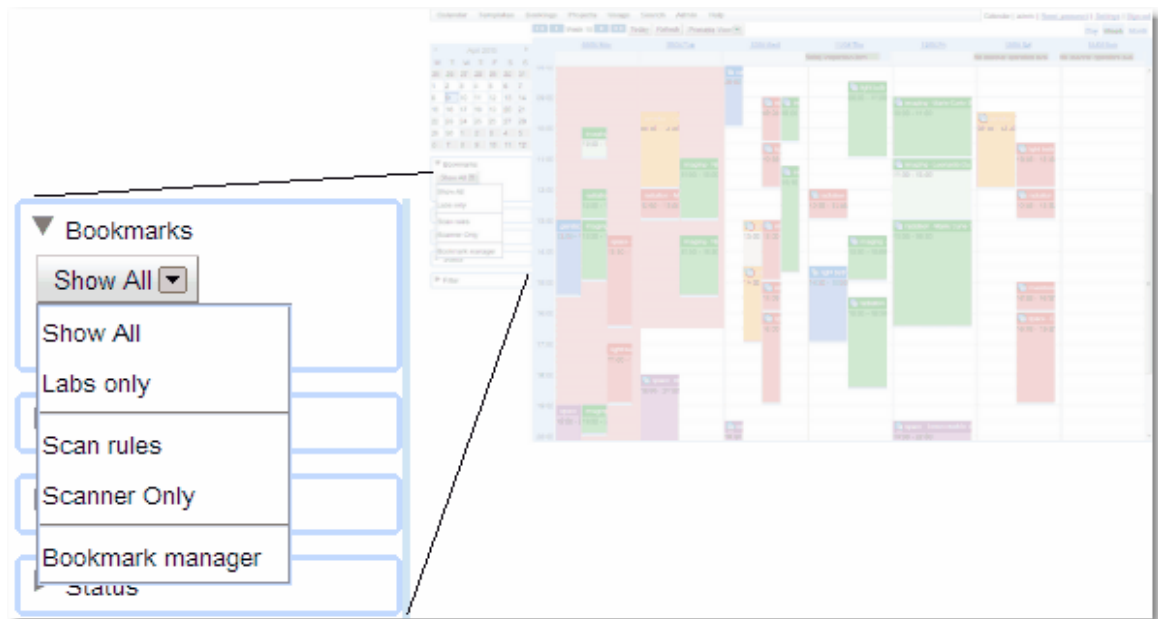
The **Bookmark Manager** allows [bookmarks](#)⁶⁵³ to be created/edited for use in the [Bookings Calendar](#)⁶⁵². The manager also keeps track of which *bookmarks* apply to which users and which *bookmarks* are system wide.

A *bookmark* holds a list of [resources](#)⁶⁵⁵ that a user would like to see on the *calendar* at the same time. This is a quick way for the user to set up their *calendar* to view the required *resources*. A user may have many *bookmarks* to enable different views of the *calendar*. A user may see their own *bookmarks* and all system wide *bookmarks*. When a user creates a *bookmark* they can make them system wide for other users to use.

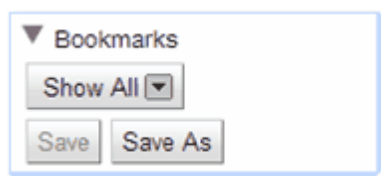
All users start with the default **System Bookmarks->Show All** bookmark by default. It is important that if you create any [new resources](#)²⁸⁴ you add them to that *bookmark* so all users can see them initially.

Calendar Page Bookmarks

This is what a standard *bookmark* list would look like in the *calendar* view. The top two *bookmarks* are system wide, the bottom two are the users specific *bookmarks*, the final option opens up the **Bookmark Manager** for the user:



A user with the correct [permission](#)⁶⁵⁴ may save a current *resource* configuration as a bookmark. Set up the *resources* required in the *bookmark* as those that can be seen on the *calendar*, under the *bookmark* pull down, click on the **Save As** button.



Then save the *bookmark*, giving it a name, saying whether anyone can see it (**System Wide**). The **Public** [property](#)⁶⁵⁵ is not currently used (any user can edit this *bookmark*).

▼ Bookmarks	
Show All ▼	
Save Save As	
Name	My bookmark
System Wide	<input type="checkbox"/>
Public	<input checked="" type="checkbox"/>
Cancel Save	

User Bookmark Permissions

By default non-admin users cannot create *bookmarks* or access the **Bookmark Manager**, either from the **Admin->Bookmark Manager** menu option or the **Bookmark Manager** option on the bottom of their *bookmark* list in the *calendar* view. In order to change this *Permission* **Create->Bookmark->Anybody can create non-system bookmarks** would need to be enabled. If you have given users *permission* to create *bookmarks* you will need to give them *permission* to edit *bookmarks* by enabling **Update->Bookmark->Anybody can update their own non-system bookmarks**. Please read the [Permissions](#)⁶⁵ chapter for more information on setting and using *Permissions*.

Bookmark Manager

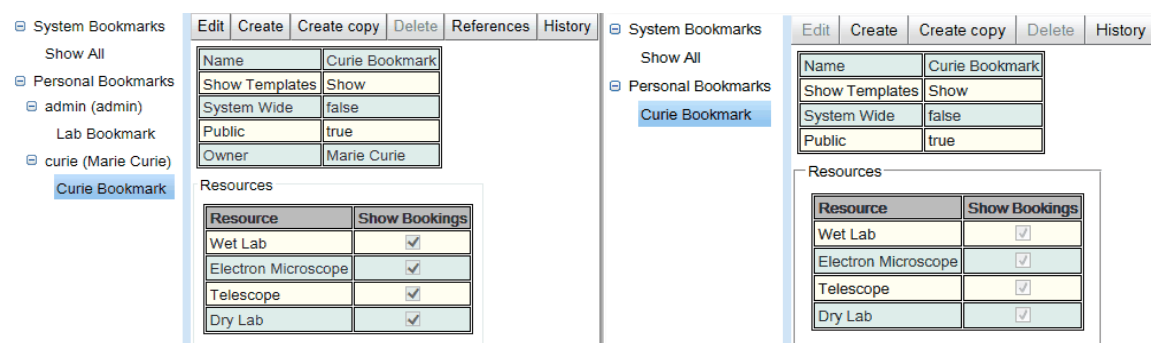
To open up the **Bookmark Manager** the **Admin** user can use the **Admin->Bookmark Manager** item. A user would have to use the option on the pop up from the *Bookmarks* section of the *Calendar* page (see **Calendar Page Bookmarks**).

The **Bookmark Manager** allows users to see their own and system wide *bookmarks*. If they have the permissions the user may edit and delete these *bookmarks*. An **Admin** user can see all users personal *bookmarks* as well as all the system wide *bookmarks*, and may edit or delete any of them. The exception is no-one can delete a *bookmark* that is currently in use.

A *bookmark* has a number of basic [properties](#)⁶⁵⁵:

Property	Description
Name	The name of the <i>bookmark</i> as assigned by the user.
Show Templates	If you are in the <i>calendar</i> view which Time Templates ⁶⁵⁶ will be shown. There are two choices: Show: Show any <i>Time Template</i> information in the <i>Bookings Calendar</i> . Hide: Hide any <i>Time Template</i> information in the <i>Bookings Calendar</i> .
System Wide	Can everyone see the report or is it a personal one. If this is true the report will appear under System Bookmarks rather than Personal Bookmarks .
Public	If a system wide <i>bookmark</i> can anyone modify it. (This is currently not being used).
Owner	The user who created the <i>bookmark</i> and if Public is false, the only person who can edit a <i>bookmark</i> if it is also System Wide .

The page is split into two. On the left is a list of the **System Bookmarks** (those that anyone can see) and the users **Personal Bookmarks**. On the right is the area where *bookmarks* can edited, created and deleted. Where the page splits can be moved to change the space available for each of the sides.



Admin view of the Bookmark Manager

User View of the Bookmark Manager

The main difference between the two views is that the **Admin** user can see **Personal Bookmarks** for all users. Also the **Admin** user can see the [owner](#)⁶⁵⁴ of a *bookmark* and also get a list of [references](#)⁶⁵⁵ to the *bookmark*.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Editing Bookmarks

Once the choice to edit a *bookmark* or create a new one has been made the view will be in edit mode.

Cancel		Save	
Name	Curie Bookmark		
Show Templates	Show	▼	
System Wide	false	▼	
Public	true	▼	
Owner	curie (Marie Curie) ▼		
Resources			
Choose resources...			
	Resource	Show Bookings	
[drag]	Wet Lab	<input checked="" type="checkbox"/>	
[drag]	Electron Microscope	<input checked="" type="checkbox"/>	
[drag]	Telescope	<input checked="" type="checkbox"/>	
[drag]	Dry Lab	<input checked="" type="checkbox"/>	

In the top half of the view there are the *properties* that can be edited. When finished either **Save** to complete or **Cancel** to reject the changes.

In the bottom half of the view the *resources* that will be shown by using this *bookmark* can be chosen. To add or delete resources available in this list use the **Choose Resources...** button. This will open up a resource selector. To learn more about how this works read the section on the Resource Selector. Once the *resources* required are in this list, check/uncheck the *resources* required to view in the *Bookings Calendar* for this *bookmark*.

Click on the first column *[drag]* to drag and drop *resources* to reorder the list.

4.5 Using Calpendo To Send Emails

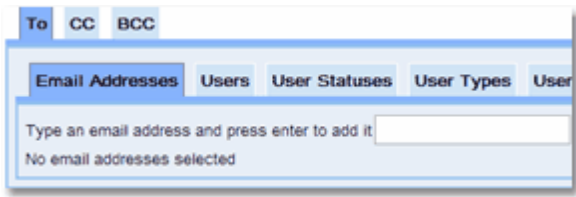

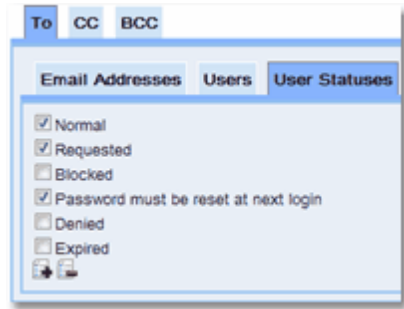
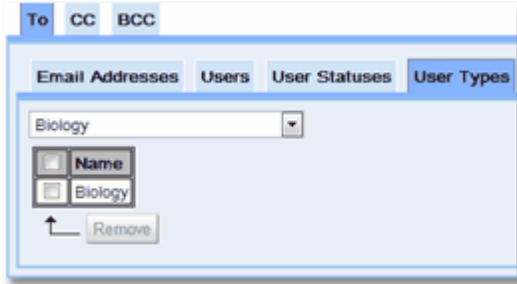
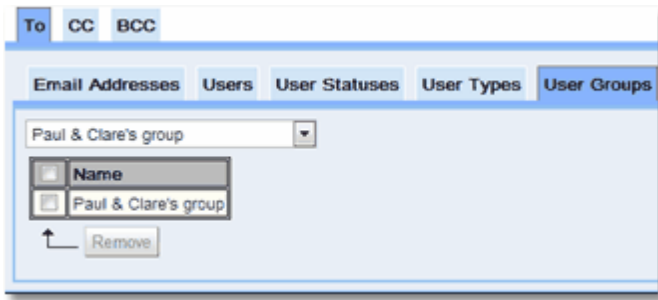
Calpendo can be used to send emails to anybody that is registered with **Calpendo**, since all users must provide an email address. Emails can also be sent to any email address as required. This can be used as a means of general communication so that the administrator doesn't necessarily need to maintain email distribution lists. This works because if you want to send emails to people who use your [resources](#)⁶⁵⁵, then they are probably also **Calpendo** users. If you are running **Calpendo** on your own server, **Calpendo** can make a secure connection to the mail server for sending email see Global Preferences Email on how to set this up..

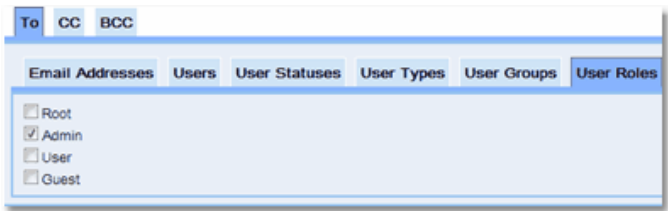
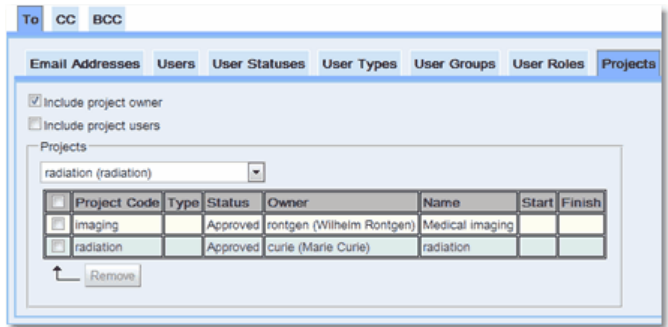
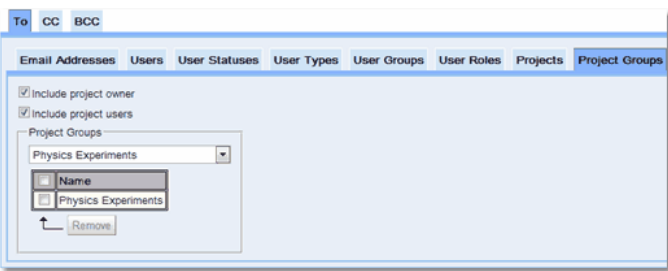
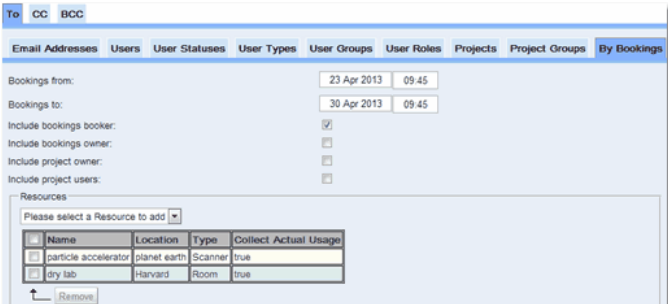
Calpendo can also be used to send email to people who have a [booking](#)⁶⁵² over the next few days, or whatever period required. This is useful when there's a problem with a *resource* that needs to be brought to the attention of those that are due to use it.

Send emails using the **Send Email** page, which is placed by default onto the menu as **Admin-->Send Email**.

The page sets up the email subject, email message (which can be text or HTML) and priority, as though using any normal email client. There are then three tabs, named **To**, **CC** and **BCC**, to specify who the email is sent to, copied to and blindly sent to. As long as someone is in the **CC** or **BCC** list then there doesn't need to be anything in the **To** list.

Each of the **To**, **CC** and **BCC** tabs have the same content, with tabs as follows:

Tab Name	Description	
Email Addresses	This adds any number of specific email addresses. This means emails can be sent to people who are not known by Calpendo .	
Users	This chooses individual users to send the email to. Use the drop down to select the users required. Use the tick box to choose which users you wish to remove and then click the Remove button.	
User Statuses	This chooses the statuses ⁶⁵⁶ of users that are allowed to received the email. Note that the recipients will be those specified by any of the other tabs that are not also prohibited by this User Statuses tab.	
User Types	If Calpendo is configured so that users are classified by their user type ⁶⁵⁶ , then select one or more <i>user types</i> so that all users with those <i>types</i> should receive an email.	
User Groups	Select any number of user groups ⁶⁵⁶ to send email to. Use the drop down to select the <i>user groups</i> required. Use the tick box to choose which <i>user groups</i> to remove and then click the Remove button.	

Tab Name	Description	
User Roles	Add any user to the recipient list that has the selected roles ⁶⁵⁶ .	
Projects	Select any number of projects ⁶⁵⁴ , and choose whether the email should be sent to the <i>projects'</i> owners, their users or both. Use the drop down to select the <i>projects</i> required. Use the tick box to choose which projects to remove and then click the Remove button.	
Project Groups	Similar to the Projects tab, this chooses project groups ⁶⁵⁴ and whether the email should be sent to the <i>projects'</i> owners, their users or both.	
By Bookings	This specifies a selection of <i>bookings</i> by the <i>resource</i> and time period, and then to choose whether the email should go to the person that made each <i>booking</i> , the person that owns it, and for any <i>project</i> associated with the <i>booking</i> to choose whether to send the email to the <i>project</i> owner or the <i>project</i> users.	

When the email and its recipients are ready, then use the buttons in the button bar at the bottom:

Button	Description
Send	This sends the email
Show Recipients	Calpendo calculates who the email would be sent to and shows a list of those users without actually sending the email. Use this to verify that the email will target the expected users. If an email address appears more than once in any of the tabs then only one email is sent with To overriding CC overriding BCC to decide which address line is used.
Reset	After using the Send Email page, press the Reset button to reset the page back to its original content, ready to compose a new email.

4.6 Import

Calpendo supports the import of data using comma-separated values (csv). This is found on the **Admin->Import** menu.

In order to do this an import file needs to be created. Only information for a single [Biskit Type](#)⁶⁵² can be imported at a time. For example if importing [projects](#)⁶⁵⁴, import the *project* and then separately the [project resource settings](#)⁶⁵⁴ for each of the *projects*. The exception to this is, if importing data belonging to a **Slave Biskit Type**, that needs to be imported with the **Master Biskit Type** data as if the **Slave** [properties](#)⁶⁵⁵ actually belonged to the **Master** rather than the **Slave**.

Once the import files have been created they can be used in the **Calpendo** import page to import your information.

4.6.1 Example Import Files

The file should have a header row which defines the [properties](#)⁶⁵⁵ that are going to be imported, and then one row for each object of the [Biskit Type](#)⁶⁵². Use the [Search](#)¹¹⁸ page or the [Bakery](#)⁵³⁷ to find the names for the *properties*. All the fields need to be comma separated.

To modify the value of a *property* that is from a component [biskit](#)⁶⁵², values to all the *properties* on that component must be provided. Any *properties* not mentioned will be attempted to be set to null. If this fails then an error will occur.

Eg, When importing users and setting a new value for **userIdentity.authenticationMethod**, then a value for **userIdentity.loginName** must also be set.

If a new value for **userIdentity.authenticationMethod** is set and a new value for **userIdentity.loginName** is not set, then the **userIdentity.loginName** will be set to null.

A *property* that is used to identify the *biskit* to be updated can itself be updated. For example, when importing users, users could be identified by their **email property** (if it's unique) and the import could also modify that **email property**. To do this, two **email** columns in the CSV file are required.

i.e.

email,email

[tim@conaptic.com,timb@conaptic.com](#)

The first column would then be the key *property* to identify the user to be changed and the second column would be the new value of the **email property**.

Use the following options **Key:** and **Ignore:** (typed before the property name) to pre select the properties that will be key fields or fields to be ignored.

Example File For Importing A User

```
import user.txt - Notepad
File Edit Format View Help
userIdentity.loginName,userIdentity.authenticationMethod,givenName,familyname,status,roles,email,password
Timb,Local,Tim,Bilder,Blocked,USER,ben@conaptic.com,timbil
```

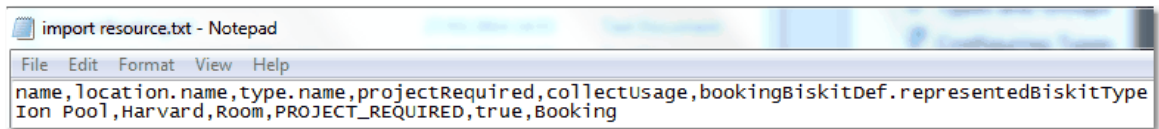
The **User** record that will be created from this file:

Login name	Timb
Roles	User
Self	Timb (Tim Bilderbeck)
Given name	Tim
Other name	
Family name	Bilderbeck
Email address	blah@conaptic.com
User Type	Physics
Password	*****
Status	Normal
Expiry Date	
Requested Project Code(s)	

Name	Value	Comments
userIdentity.loginName	Timb	String
userIdentity.authenticationMethod	Local	Needs to match available authentication method exactly.
roles	4	Set by bits 1 = Root 2 = Admin 4 = User 8 = Guest See User Roles ⁵⁵⁴ for more information.
givenName	Tim	String
familyName	Bilderbeck	String
email	blah@conaptic.com	String
userType.name	Physics	Needs to match name exactly
password	timbil	Depends on system configuration
status	Normal	Requested, Normal, Blocked

Notice the order of the *properties* does not matter. Depending on how the system is configured password may be a compulsory *property* and may have a minimum number of characters and/or a combination of alphanumerics.

Example File For Importing A Resource



```
import_resource.txt - Notepad
File Edit Format View Help
name,location.name,type.name,projectRequired,collectUsage,bookingBiskitDef.representedBiskitType,lon Pool,Harvard,Room,PROJECT_REQUIRED,true,Booking
```

The **Resource** record that will be created from this file:

Name	lon Pool
Location	Harvard
Type	Room
Project Required	Any Project Required
Booking Subtype	Booking
Require Reason for Cancellations	<input type="checkbox"/>
Allocate Calendar Column	<input checked="" type="checkbox"/>
Collect Actual Usage	<input checked="" type="checkbox"/>
Allow Old Changes	<input type="checkbox"/>
Enable pre-defined time slots	<input type="checkbox"/>
Version	3
Created	19 May 2014 09:26
Updated	21 May 2014 10:12

Name	Value	Comments
name	lon Pool	String
location.name	Harvard	Needs to match name exactly
type.name	Room	Needs to match name exactly
projectRequired	PROJECT_REQUIRED	PROJECT_REQUIRED or PROJECT_NOT_REQUIRED or PROJECT_WITH_RESOURCE_ENTRY
collectUsage	true	true or false
bookingBiskitDef.representedBiskitType	Booking	The Biskit Type to be used for bookings of this resource

Example File For Importing A Project

import project.txt - Notepad

```
File Edit Format View Help
projectCode,type.name,status,owner.userId,entity.loginName,name,start,finish
SunSpots,Physics phantoms,Approved,curie,Sun Spots,25 May 2013,25 Dec 2014
```

The **Project** record that will be created from this file:

General	Project Code	SunSpots
Project Resource Settings	Type	Physics Projects
Users	Status	Approved
Project Groups	Owner	curie (Marie Curie)
	Name	Sun Spots
	Description	
	Phone Number	
	Start	25 May 2013
	Finish	25 Dec 2014
	Principal Investigator	
	Ethics Approval Number	
	Funding Agency	
	Account Number	

Name	Value	Comments
projectCode	SunSpots	String
type.name	Physics Project	Needs to match name exactly
status	Approved	Approved, Requested, Denied, Terminated
owner.userId entity.loginName	curie	Login name
name	Sun Spots	String
start	25 May 2013	Date
finish	25 Dec 2014	Date

Projects have other information attached to them but they are stored in different *Biskit Types*. These are used to store the [Project Resource Settings](#)⁶⁵⁴, **Users** and [Project Groups](#)⁶⁵⁴ *properties*. In order to import this information, first import the *project* information and then import each of the other three *Biskit Types* one after another.

For example importing the *Project Resource Settings* for the above project:

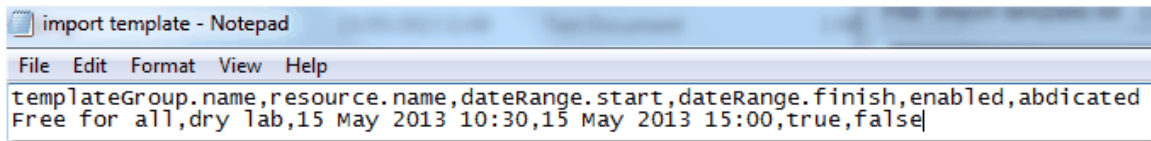
import project resource settings - Notepad

```
File Edit Format View Help
project.name,resource.name,numberofSessions,minutesPerSession,costPerHour
Sun Spots,dry lab,10,60,400
```

Project	SunSpots (Sun Spots)
Resource	dry lab
Number of Sessions	10
Minutes per Session	60
Cost Per Hour	400

Name	Value	Comments
project.name	SunSpots	Needs to match name exactly
resource.name	Physics Project	Needs to match name exactly
numberOfSessions	10	Integer
minutesPerSession	60	Integer
costPerHour	400	Double

Example File For Importing A Time Template



```
import template - Notepad
File Edit Format View Help
templateGroup.name,resource.name,dateRange.start,dateRange.finish,enabled,abdicated
Free for all,dry lab,15 May 2013 10:30,15 May 2013 15:00,true,false
```

The **Template** record that will be created from this file:

Date Range	15 May 2013 10:30 - 15 May 2013 15:00	Name	Value	Comments
Resource	dry lab	date Range.start	15 May 2013 10:30	Date and Time
TAM Group	Free for all	dateRange.finish	15 May 2013 15:00	Date and Time
Near Term TAM Group		templateGroup	Free for All	string
Near Term		Enabled	true	true or false
Enabled	true	Abdicated	false	true or false
Far Term				
Far Term TAM Group				
Abdicated	false			
Abdicated Date				
Abdicated By				

To import repeat *Time Templates* see below for additional information about importing *repeat bookings*.

Example File For Importing Two Bookings

```
import Booking.txt - Notepad
File Edit Format View Help
booker.userIdentity,loginName,owner.userIdentity,loginName,status,dateRange.start,dateRange.finish,resource.name,project.projectCode
admin,admin,Approved,18 May 2013 08:00,18 May 2013 08:30,dry lab,inventions
admin,admin,Approved,18 May 2013 09:00,18 May 2013 09:30,dry lab,helper
```

The **Booking** record that will be created from the first import line in this file is:

Resource	dry lab
Type	
Booker	admin (admin)
Project Resource Settings	dry lab 0 scans of 60 mins for £300.0
Owner	admin (admin)
Project	inventions (inventions)
Status	Approved
Approved by template	false
Warned	
Date Range	18 May 2013 08:00 - 08:30
Created	16 May 2013 10:23
Cancelled	
Modified	
All day	
Description	
Reminder Trigger Time	
Reminder Sent	false
Remind Booker	false
Remind Owner	false
Remind Project Owner	false
Remind Project Users	false

Repeat

Not set

Name	Value	Comments
booker.userIdentity.loginName	admin	Needs to match login name exactly
owner.userIdentity.loginName	admin	Needs to match login name exactly
status	Approved	Approved, Requested, Cancelled, Denied
dateRange.start	18 May 2013 08:00	Date and Time
dateRange.finish	18 May 2013 08:30	Date and Time
resource.name	dry lab	Needs to match name exactly
project.Project Code	inventions	Needs to match project code exactly

When importing [repeat](#)⁶⁵⁵ bookings the repeat sub *Biskit Type* for the *repeat booking* needs to be defined as well as all the *properties* defined for that *Biskit Type* and all the properties of the parent *Biskit Type Repeat*. Any *property* not defined will have its value set to null.

Biskit Type	Property	Type	Example Value
Repeat	repeat.biskitType (must always be present)	Biskit	AnnualRepeat DailyRepeat MonthlyRepeatByDate MonthlyRepeatByDay WeeklyRepeat
	repeat.start	Date	12 January 2015 13:00
	repeat.finish	Date	12 January 2999 14:00 2999 for never ending repeats
	repeat.repeatEvery	int	1
	repeat.repeatType (must match repeat.biskitType)	JavaEnum	Annually Daily Monthly by date Monthly by day Weekly
Annual Repeat	repeat.dayOfMonth	Int	1-31
	repeat.monthOfYear	int	0-11 with 0 = January
Daily Repeat	None		
Monthly Repeat By Date	repeat.dayOfMonth	int	1-31
Monthly Repeat By Day	repeat.dayOfWeek	int	1-7 with 1 = Sunday
	repeat.weekOfMonth	int	1-5
Weekly Repeat	repeat.sunday	Boolean	True or False
	repeat.monday	Boolean	True or False
	repeat.tuesday	Boolean	True or False
	repeat.wednesday	Boolean	True or False
	repeat.thursday	Boolean	True or False
	repeat.friday	Boolean	True or False
	repeat.saturday	Boolean	True or False

4.6.2 The Import Page

When importing data you can specify whether data is imported if there are any errors, or that no data is imported from the file if any errors occur, errors include [Time Templates](#)⁶⁵⁶, [Booking Rules](#)⁶⁵² and [Permissions](#)⁶⁵⁴ failing to allow creation or updating of records. Examples of how different error types are handled can be found in the section on [Handling Import File Errors](#)²⁰⁰.

On entering the **Import** page it will look like:

A button labeled 'Browse...' followed by the text 'No file selected.'

Press **Browse...** (or **Choose File**) to select the file to import. At this point **Calpendo** will attempt to decide which [Biskit Type](#)⁶⁵² is being imported from the header information. If it cannot determine the *Biskit Type*, the user will need to select it.

import user.txt Choose new file Biskit type: Select import biskit type Update Existing Data Preview Import

Once the **Import** page knows which *Biskit Type* it is importing it will look like this:

import user.txt Choose new file Biskit type: User Update Existing Data Preview Import

At this point decide whether to choose to **Update Existing Data** or **Insert New data**. If updating data then import needs to know which columns to use as the key fields to find the appropriate records to update. Click on the **Key** option below the column headings of those [properties](#)⁶⁵⁵ that are keys fields or put **Key:** before the *property* name in the file.

In this example for importing **User** data, the key is specified to be the combination of **userIdentity.loginName** and **userIdentity.authenticationMethod.name**, as the combination of these two will be unique.

	1 (A) userIdentity.loginName	2 (B) userIdentity.authenticationMethod.name	3 (C) givenName	4 (D) familyName	5 (E) status	6 (F) roles	7 (G) email	8 (H) password
	<input type="radio"/> Normal	<input checked="" type="radio"/> Normal	<input checked="" type="radio"/> Normal	<input checked="" type="radio"/> Normal	<input checked="" type="radio"/> Normal	<input type="radio"/> Normal	<input type="radio"/> Normal	<input type="radio"/> Normal
	<input type="radio"/> Ignore	<input type="radio"/> Ignore	<input type="radio"/> Ignore	<input type="radio"/> Ignore	<input type="radio"/> Ignore	<input type="radio"/> Ignore	<input type="radio"/> Ignore	<input type="radio"/> Ignore
	<input checked="" type="radio"/> Key	<input type="radio"/> Key	<input type="radio"/> Key	<input type="radio"/> Key	<input type="radio"/> Key	<input type="radio"/> Key	<input type="radio"/> Key	<input type="radio"/> Key

In this example for importing **User** data, **Insert New Data** has been chosen, the **Key** option has disappeared as no longer required.

import user.txt	Choose new file	Biskit type: User	Insert New Data	Preview	Import		
1 (A) userIdentity.loginName	2 (B) userIdentity.authenticationMethod.name	3 (C) givenName	4 (D) familyName	5 (E) status	6 (F) roles	7 (G) email	8 (H) password
<input checked="" type="radio"/> Normal <input type="radio"/> Ignore	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore

Choose the columns to be ignored during the import by clicking the **Ignore** option below the *property* name or by putting **Ignore:** before the *property* name in the file, and then press the **Preview** button to see the data before importing, at least one **Preview** must be done before attempting to **Import**..

If previewing once the data has been loaded into the top pane, select an individual record and see the record to be created/updated in the bottom left pane, and if doing an update the original record is in the bottom right pane.

import user.txt	Choose new file	Biskit type: User	Update Existing Data	Preview	Import		
1 (A) userIdentity.loginName	2 (B) userIdentity.authenticationMethod.name	3 (C) givenName	4 (D) familyName	5 (E) status	6 (F) roles	7 (G) email	8 (H) password
<input checked="" type="radio"/> Normal <input type="radio"/> Ignore <input type="radio"/> Key	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore <input type="radio"/> Key	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore <input type="radio"/> Key	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore <input type="radio"/> Key	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore <input type="radio"/> Key	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore <input type="radio"/> Key	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore <input type="radio"/> Key	<input checked="" type="radio"/> Normal <input type="radio"/> Ignore <input type="radio"/> Key
No problems	No problems	No problems	No problems	No problems	No problems	No problems	No problems
2	Timb	Local	Tim	Bilder	Blocked	USER	tim@calpendo.com

Roles	No roles
Self	Timb (Tim Bilder)
Given name	Tim
Other name	
Family name	Bilder
Email address	tim@calpendo.com
User Type	
Password	
Status	Blocked
Expiry Date	
Version	1
Created	12 Mar 2015 09:23
Updated	12 Mar 2015 09:23
Requested Project Code(s)	

Identity Projects Groups

Login name	Timb
Authentication Method	Local

Roles	No roles
Self	Timb (Tim Bilder)
Given name	Tim
Other name	
Family name	Bilder
Email address	tim@calpendo.com
User Type	
Password	
Status	Blocked
Expiry Date	
Version	1
Created	12 Mar 2015 09:23
Updated	12 Mar 2015 09:23
Requested Project Code(s)	

Identity Projects Groups

Login name	Timb
Authentication Method	Local

Press the **Import** button to import the data into **Calpendo**. While the import is being run a progress bar will be displayed which will show the time taken, % amount completed and any errors. There is also a **Cancel** button to cancel the import. When the import completes a pop up will appear in the bottom right hand corner of the screen saying **Import Completed**.

Import Progress

Time taken 00:00:00

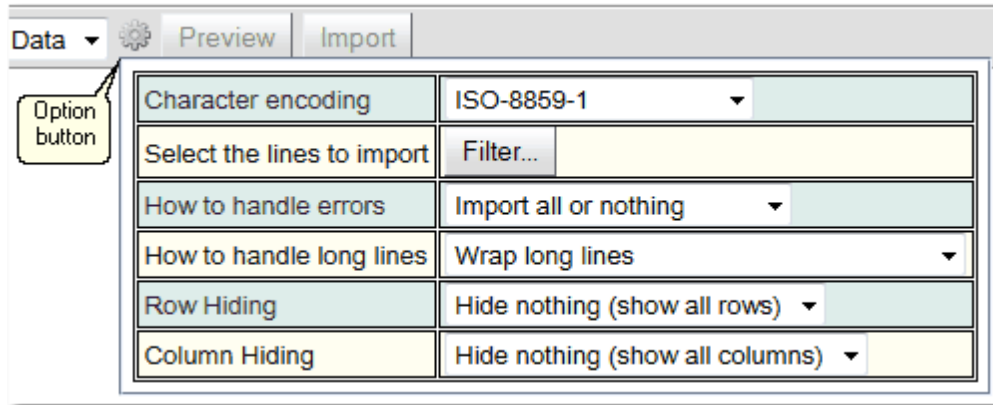
Amount completed 100%

Time remaining

Cancel

Importing Options

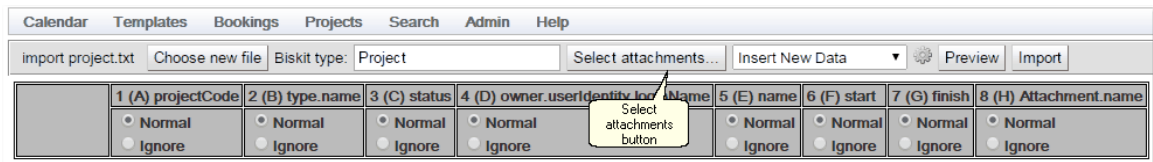
Click on the options button to see all the options available for viewing and importing the data file



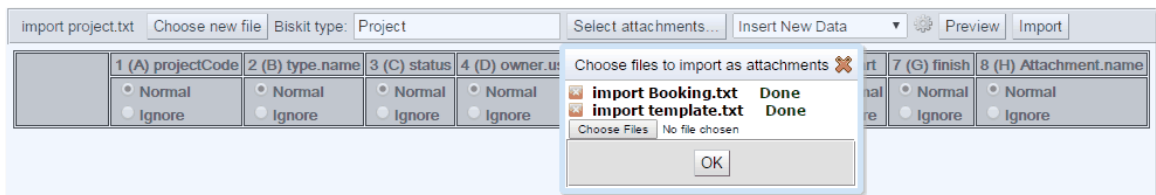
Option	Description
Character encoding	While importing Calpendo guesses at the character encoding used for the file being imported, and allows the administrator the chance to choose the one to use, allowing non-ASCII files to be imported.
Select the lines to import	Use the Filter... button to set up conditions ⁶⁵³ to define which rows will be imported.
How to handle errors	Choose Import any that do not fail which will import all records that do not have errors, or Import all or nothing , which will only import the records if they would all succeed.
How to handle long lines	Whether lines should be wrapped making the box deeper, Wrap long lines , or whether to make the table wider to deal with long lines. Do not wrap long lines; make the table wider.
Row Hiding	Whether to hide those rows that are being ignored for import, Hide ignored rows , or to hide all rows without an error, Hide error-free rows . Press the Preview button to see the results of any changes to this setting.
Column Hiding	Whether to hide those columns that are being ignored for import, Hide ignored columns , or to hide all columns without an error, Hide error-free columns . Press the Preview button to see the results of any changes to this setting.

Importing File Attachments

If any of the *properties* to be imported include the **Attachment** [BiskitDef](#)⁶⁵², then the user is supplied with an additional button to select the list of attachments to be imported. This needs to be done before the **Preview** or **Import** buttons are pressed.



Once the attachment pop up is present, use the **Browse...** (or **Choose File**) button to find all the attachments required for the import. Once selected click OK and move on to **Preview** and **Import** the data.



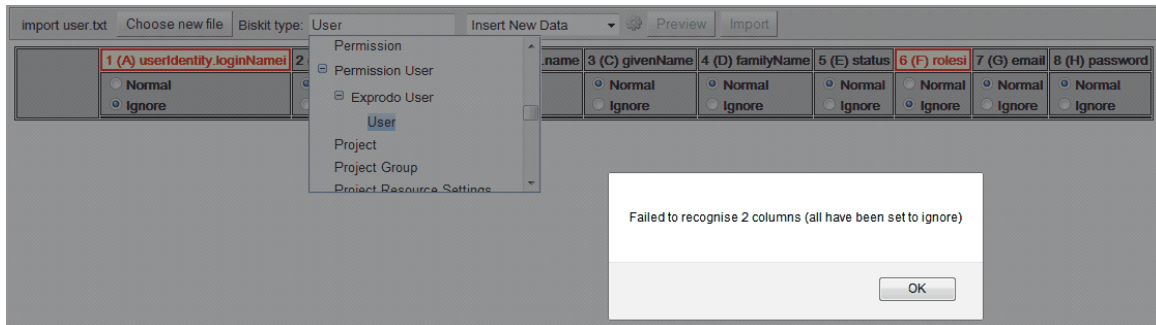
Handling Import File Errors

During the import process there are three main types of errors.

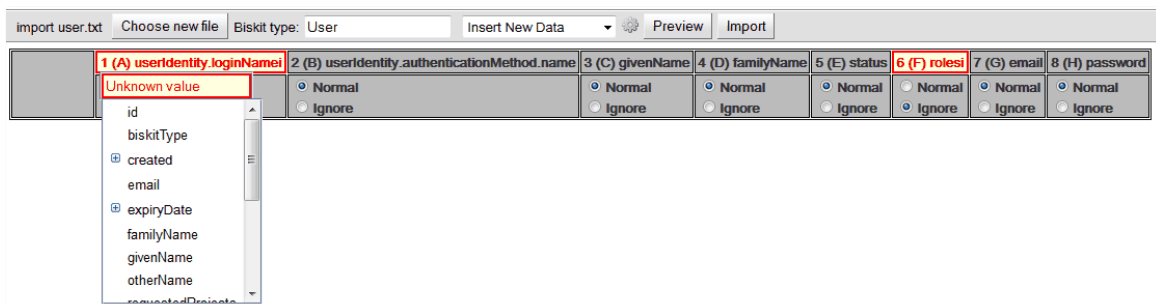
1. Incorrect *property* names or labels in the header of the import file. (Header errors)
2. Incorrect data in a record, stopping **Import** from resolving property values. (Data Errors)
3. Failure to import due to internal checking. (Violation errors)
 - a. Incorrect data in a record, causing *Booking Rules* or *Permissions* to fail to allow creation of records(e.g. password too short).
 - b. *Time Templates*, *Booking Rules* and *Permissions* failing to allow creation or updating of records.

Header Errors

Header errors are found when the import file is first loaded using the **Send** button. Once **Import** has decided on the *Biskit Type* being imported or the user has selected the correct *Biskit Type*, the header information is loaded. If any of the headers cannot be resolved to *property* names or *property* labels (in a case insensitive manner if possible) then those headers are marked in red. A pop up will inform the user how many column headings have not been recognised. Import then updates those columns from **Normal** to **Ignore**, so the user can proceed with the import without those columns.



These errors can be fixed in the Import page. Just click on any red header and you will get a drop down box with a complete lists of this *Biskit Types properties*. Select the correct one, and the error will be removed. Once done the column will be moved from **Ignore** to automatically.



Data Errors

Data errors are found either during the preview of the data, or if a preview is not done, during import.

Records with errors highlighted in red. If there are only warnings they will be highlighted in blue.

Number of problems in a column. Click to get a list of problems for this column.

Properties with errors highlighted in red, warnings highlighted in blue.

Initial pop up used to show log.

If there are problems a pop up will appear use the **Show Log** button to show a complete list of the problems. Records with an error will have their record number highlighted in red as will those *property* values which are causing an error. If a record has only warnings it will be highlighted in blue, as will those *property* values which are just warnings. Moving the cursor over a *property* with an error/warning will result in a tool tip describing the error/warning. Also for each column you can see the number of problems. If there are any problems in a column clicking on the number of problem text will produce a pop up which will show the problems for just that column.

Line in file with error

Number of property column with error

Error message

The problem log will list the problems, showing the line and column the problem occurs on as well as the problem message. Clicking on the an errors column will scroll the file to the error, and it will be coloured blue to stand out.

In order to fix these types of problems go back to the original import document and fix them there before loading it back into the **Import** page. Remember once the file to be imported has been updated, save it before reloading the file into **Calpendo Import**.

Violation Errors

Violation errors are found during the import of the data into the database. These are errors that occur when the **Calpendo** is trying to create or update the database. During this process all the *Time Templates*, *Booking Rules* and *Permissions* are checked.

	1 (A) userIdentity.loginName	2 (B) userIdentity.authenticationMethod.name	3 (C) givenName	4 (D) familyName	5 (E) status	6 (F) userType.name	7 (G) roles	8 (H) email	9 (I) password
2 problems	<input type="radio"/> Normal <input type="radio"/> Ignore	<input type="radio"/> Normal <input type="radio"/> Ignore	<input type="radio"/> Normal <input type="radio"/> Ignore	<input type="radio"/> Normal <input type="radio"/> Ignore	<input type="radio"/> Normal <input type="radio"/> Ignore	<input type="radio"/> Normal <input type="radio"/> Ignore	<input type="radio"/> Normal <input type="radio"/> Ignore	<input type="radio"/> Normal <input type="radio"/> Ignore	<input type="radio"/> Normal <input type="radio"/> Ignore
	No problems	No problems	No problems	No problems	No problems	No problems	No problems	No problems	No problems
2	Timb	local	Tim	Bilder	Normal	Physicist	4	timb@conaptic.com	timbl
3	Tima	local	Tim	Arnold	Normal	Physicist	4	tima@conaptic.com	tim

Import Result Detail

Level	Line	Column	Message
Info	-1	-1	Import not saved: found a total of 2 problems
Error	2	-1	Could not create user: Login name 'Timb' is not available
Error	3	-1	Could not create user: Your password must be at least 6 characters long

OK

Import Result

Import not saved: found a total of 2 problems

OK Show Log

If there are errors a pop up will appear, use the **Show Log** button to show a complete list of the errors. Records with an error will have their record number highlighted in red. Moving the cursor over a record number with an error will result in a tool tip coming up describing the error. The error log just shows the records that failed to import and a reason for that failure. If there are two violation reasons that will cause a record to fail, only one is found, and when fixed, the second reason will cause a failure next time around. This is because the *Time Templates*, *Booking Rules* and *Permissions* are run until one fails and the rest are then ignored.

In the example above the third record has both a password that is too short and an illegal email address, but only the short password has been found, the illegal email address has been found for the second record but not the third. So after fixing the password for the third record and the email address for the second record, if an attempt was made to import the file again with no other changes, another error would occur this time an invalid email for the third record.

In order to fix these types of errors go back to the original import document and fix them there before loading it back into the **Import** page. Remember once the file to be imported has been updated, save it before reloading the file into **Calpendo Import**.

4.7 System Events

The **System Events** page provides a convenient way to search for and display **Calpendo's** record of [events](#)⁶⁵⁶ such as people logging in or emails being sent. It is recommended that only those who administer **Calpendo** should have this added to their menu. To access the **Systems Events** page it is found on the **Admin** menu. However, the administrator may have configured **Calpendo** so that the menu is different.

The **Systems Events** page is very similar to the [Search](#)¹¹⁸ page. The *events* returned can be filtered by time, **Type**, **Source**, **Category**, **Affected Type**, **Affected ID**, **Thread** and **Process ID**. Unlike other [Search](#)¹¹⁸ pages there is no auto run option. When the page opens up it looks like the following:

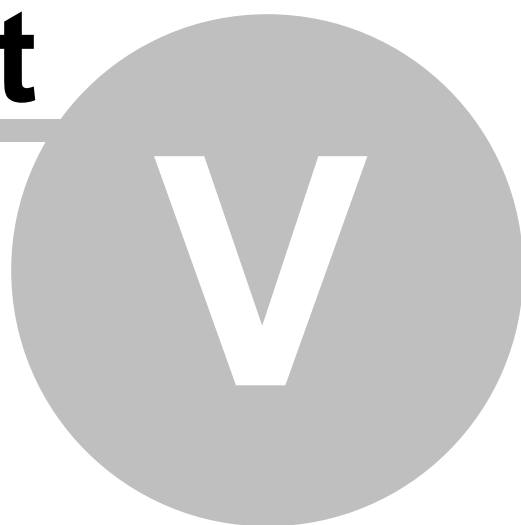
Once the search has run the results will look similar to below, clicking on an *event* will bring up the expanded view for that *event* below the list view.

When	Source	Category	Thread	Type	User	Affected Type	Affected ID	Process ID	Compressed Data Size
20 Feb 2018 00:05:35.000	RepeatableHandler	Update	com.springolutions.exprodo.core.server.ExprodoBootServlet@1ab2bdf5	Info		RepeatableHandler Timer			390
20 Feb 2018 04:30:00.000	Reminder Manager	Reminder unsent	/calpendo-demo/ExprodoReminderManager	Error		WetLabBookings	18833		0
20 Feb 2018 05:35:00.000	Timer Manager	Firing triggerable	/calpendo-demo/TimerManager	Info					338
20 Feb 2018 05:35:00.000	WorkflowManager	Run event	/calpendo-demo/WorkflowManager Worker DedicatedMsgBus Reader #2	Info		RelativeTimeWorkflowEvent	2754		328
20 Feb 2018 05:35:00.000	WorkflowManager	Run action	/calpendo-demo/WorkflowManager Worker DedicatedMsgBus Reader #2	Info		SimpleWorkflowAction	2755	365	151
20 Feb 2018 05:35:00.000	WorkflowManager	Action succeeded	/calpendo-demo/WorkflowManager Worker DedicatedMsgBus Reader #2	Info		SimpleWorkflowAction	2755	365	377
20 Feb 2018 05:35:00.000	WorkflowManager	Event succeeded	/calpendo-demo/WorkflowManager Worker DedicatedMsgBus Reader #2	Info		RelativeTimeWorkflowEvent	2754		347
20 Feb 2018 05:35:00.000	Timer Manager	Firing triggerable	/calpendo-demo/TimerManager	Info					352
20 Feb 2018 05:35:35.000	RepeatableHandler	Update	com.springolutions.exprodo.core.server.ExprodoBootServlet@1ab2bdf5	Info		RepeatableHandler Timer			294
20 Feb 2018 08:05:35.000	Timer Manager	Firing triggerable	/calpendo-demo/TimerManager	Info					353
20 Feb 2018 08:05:35.000	RepeatableHandler	Update	com.springolutions.exprodo.core.server.ExprodoBootServlet@1ab2bdf5	Info		RepeatableHandler Timer			393
20 Feb 2018 08:20:53.000	Forgotten Password Handler	Forgotten password reset initiated	http-nio-10001-exec-5	Info	Calpendo.CalpendoUser1#8705388				295
20 Feb 2018 08:22:42.000	Forgotten Password Handler	Forgotten password reset initiated	http-nio-10001-exec-10	Info	Calpendo.CalpendoUser12#8705390				293

When	20/02/2018 08:20 AM
Source	Forgotten Password Handler
Category	Forgotten password reset initiated
Thread	http-nio-10001-exec-5
Type	Info
User	Calpendo.CalpendoUser1#8707121
Affected Type	
Affected ID	
Message	Forgotten password reset initiated for nobody (Localnobody)
Process ID	
Compressed Data Size	295
Client ID	null
Forwarded For	212.69.62.173
Forwarded Host	demo.calpendo.com
Forwarded Server	demo.calpendo.com
Geometry	null
Ipaddress	212.69.62.173
Remote Address	127.0.0.1
User Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.167 Safari/63.6

If there are any questions about *system events* please get in touch with **Calpendo** customer support (info@calpendo.com).

Part



5 Calpendo Quick Start Configuration Guide

This guide will show a new administrator how to set up **Calpendo** quickly, getting it running with basic functionality.

Resources

Start off by creating each of the bookable [resources](#)⁶⁵⁵ required, such as equipment and rooms etc. To do this go to **Admin->Resource Editor** and click on **Resource**. Then, for each *resource* to be created:

1. Click the **Create** button.
2. Give the new *resource* a name.
3. Leave the location as **Please select a location**. Locations can be added later.
4. Use one of the existing [resource types](#)⁶⁵⁵ even if they don't quite match what's needed. New *resource types* can be added later.
5. Specify whether a [project](#)⁶⁵⁴ is required to book the *resource*.
6. Leave everything else as default values.
7. Click the **Save** button to save your *resource*.

See Configuring Resources and Locations in the [Calpendo documentation](#).

Once all the required *resources* have been created, they need to be added to the **Show All bookmark**⁶⁵³ so that the *resources* will appear in the [calendar](#)⁶⁵². Go to **Admin->Bookmark Manager** and:

1. Under **System Bookmarks** click the **Show All bookmark**.
2. Click the **Edit** button to go into edit mode.
3. Click the **Choose Resources** button.
4. Move all the resources from **Available Resources** to **Selected Resources** by selecting each one and clicking the right arrow.
5. When finished click **OK**.
6. Click the **Save** button.

See the Bookmark Manager chapter in the [Calpendo documentation](#).

Projects

Go to **Projects->Create Project** and create a few *projects*. Information stored in a *project* may need to be changed by adding your own [properties](#)⁶⁵⁵, but just use the defaults for now. Then for each *project* to be created:

1. Give each project a [Project Code](#)⁶⁵⁴ and a **Name**, and make sure to change the [Status](#)⁶⁵⁴ to **Approved** so that non-administrators can use the project for [booking](#)⁶⁵².
2. Go to the **Project Resource Settings** tab.
3. Click the **Choose Resources** button.
4. Move the *resources* that can be booked for this *project* from **Available Resources** to **Selected Resources** by selecting each one and clicking the right arrow.

5. When finished, click **OK**.
6. Click the **Submit project request** button.

See Creating Projects chapter in the [Calpendo documentation](#).

Email

If **Calpendo** is hosted by **Exprodo Software**, then the email will already be configured. Otherwise, go to **Admin->Global Preferences** and configure the email tab as described in [Global Preferences](#)⁶⁵³ Email chapter in the [Calpendo documentation](#).

Users

Any new users should be sent a link to the **Calpendo Login** page, and told to use the **Register New User** button to make a user request.

If email is set up correctly, the administrator should get an email telling them about each new user request. To approve a user, go to **Admin->User Request** and:

1. Click on the **Login Name**.
2. Set **Status** to **Normal**.
3. Set the **Roles** to be assigned to the user.
4. In the **Projects** tab at the bottom, click on **Please Select a Project to add** and select the *projects* to be added to this user. (A user needs to be associated with a *project* before they can use it to make a *booking*.)
5. Click the **Save** button.

See the [The User Approval Process](#)¹⁶⁷ and [The User Requests Page](#)¹⁶⁷ chapters in the [Calpendo documentation](#).

If the users are going to be created by the administrator, go to **User->Search** and click the **Create** button to create a new user. Remember to allocate any *projects* to that user.

At least one user should be set up with the **Admin role**⁶⁵⁶. It is advisable to leave the initial **Admin** and **Root** users alone as this allows **Exprodo Software** access to your **Calpendo** in the event of any problems. For more help read the [Modifying Users](#)¹⁷⁷ chapter.

Next Steps

Once the above configuration is done, **Calpendo** will be ready for use.

Calpendo is now working, but many additional features and configuration options are available. To see what else could be configured, see the Before Going Live section of the Initial Configuration chapter in the [Calpendo documentation](#).

It would also be advisable to read through the **Calpendo Users Guide** chapters on the Bookings Calendar in order to get a full understanding on how the [calendar](#)⁶⁵² works and the chapters on [The Booking Approval Process](#)¹⁵⁰ and [The Booking Request Page](#)¹⁵¹ in the [Calpendo documentation](#) in order to understand how to approve *bookings*.

If there are any questions, please email support@exprodo.com.

Part

VI

6 Calpendo Configuration Guide

Calpendo was designed from the very beginning to be flexible, so that it could meet the requirements of many different types of facilities. It's targeted at people who have shared [resources](#)⁶⁵⁵ that need [booking](#)⁶⁵², but within that, there are many different ways you might like a *booking* system to work.

The **Calpendo Configuration Guide** describes how to change your **Calpendo** so that works the way you want it to. This does not normally need to be read by non-administrators, nor by anybody who only needs to do the day-to-day administration. This guide should be read in conjunction with the [Calpendo User Guide](#)²⁸ and [Calpendo Administration Guide](#)¹⁴⁸.

6.1 Configuration Questions

Calpendo was designed from the very beginning to be flexible, so that it could meet the requirements of many different types of facilities. It's targeted at people who have shared [resources](#)⁶⁵⁵ that need [booking](#)⁶⁵², but within that, there are many different ways you might like a *booking* system to work. Consequently, there are many options for how **Calpendo** can be configured.

When first using **Calpendo**, it is unusual to know exactly (or even approximately) how it is required to be configured for the facility.

This section poses some of the questions to ask yourself as the configurer, but understand that most of them will not be answerable initially. The options are numerous enough that it is expected that the system will be configured over a period of time as a better feel for what can be done is acquired and more information on how it needs to work for the facility is realised. Also, these questions only give a taste of some of the things that can be done.

Please understand that things can be made as simple or as complicated as required. If the facility is relatively new, and the *resources* are not yet fully utilised, then some of the things that **Calpendo** allows may not be required.

Resources

- What *resources* are there that should be bookable?
- Is there going to be a single *resource* to record annual leave?
- Is there going to be a *resource* to represent individual people whose time can be booked?
- Do bookers of a *resource* have to have an allocated [project](#)⁶⁵⁴.

Booking Requests

- Are users going to create requests that are then approved by an administrator?
- Or do users create pre-approved *bookings*?
- Are *bookings* to be pre-approved for some people and/or some of the time?
- Who can make *bookings* for which *resource*?
- What if any information is required for a *booking* to be made for a particular *resource*?

Booking Restrictions

- Is it required to put users into separate categories for *booking* restrictions, and how might that to work?
- How might the [Time Templates](#)⁶⁵⁶ work? Are *bookings* going to be prohibited in other people's time? Is that behaviour going to change when it gets close to the time? So a slot left free with a day or a week to go becomes up for grabs to everyone?
- Can everyone book as far into the future as they want?
- Are double *bookings* allowed? Perhaps for some *resources* it might be allowed (e.g. a computer room with numerous machines)
- Are pre-requisite *bookings* going to be enforced? That is, that a user must book *resource X* before they are allowed to use *resource Y*.
- Is there a limit on how much time a *resource* can be booked for in a given period of time? If so, who should be allowed to book how much time?

Booking Modifications

- Do users have to downgrade approved *bookings* to a request so they can then make changes to it that need to be re-approved?
- Or are users to be allowed to make some changes to [approved bookings](#)⁶⁵²?
- Some users may fear cancelling *bookings* because the statistics are collected. They may try to downgrade to a request and then change the time or day (by many weeks) to avoid being seen to cancel. So will users be allowed to change things like the time of day of an *approved booking*, but not allowed the date to be changed.

Administration

- Who is going to administrate **Calpendo**?
- Who will install and upgrade **Calpendo**?
- Are there going to be separate administrators so some people only see the most common and easier admin options? Should a menu be configured so that a simple **Admin** menu offers the ability to handle *booking* requests, new user requests and *project* requests, but not more advanced administration option, and an advanced **Admin** menu that also gives access to automated emails, *Permissions*, *Time Templates* and *Booking Rules*.

Projects

- What process are required to be put in place for creating new *projects*?
- Will there be a single-step approval process, or a multi-step approval process in which many people each give their approval. Will people need to address questions about whether the *project* is sensible, has ethics approval or has its finances properly arranged.
- What questions need to be asked of users when they create a new *project* request? For example, there might be safety related questions about what they are intending to do.
- What information needs to be stored on a *project*?
- What per-*project* and per-*resource* information do needs to be stored? For example, the price for using each *resource* for a *project*.

Services

- What services are available?
- When ordering a service what information is required?
- Costing for services.

Workflows

- Are emails to be sent when *bookings* are cancelled?
- Should any cancellation trigger an email, or just some of them?
- Who should receive such emails?

Permissions

- Who should be allowed to do what within the system?

6.2 Initial Configuration

When **Calpendo** is first installed, have a play with it to help understand the way it works, and to become more familiar with the configuration options. There are some things that need to be set up as soon as you start playing with **Calpendo**, and some things that must be done before going live and using **Calpendo** for real. These are described here.

Resources

For each [resource](#)⁶⁵⁵ that is to be bookable (rooms, scanners, people or whatever), create a *resource* using the **Resource Editor** as described by [Resources](#)²⁸⁴. There is no need to worry about [resource groups](#)⁶⁵⁵ or locations just yet. Just make sure that all the *resources* required for now are created.

Email Preferences

Tell **Calpendo** how it should send email, so that **Automatic Emails**, reminders and manual emails will work properly. Set the email preferences on the Email tab of the [Global Preferences](#)⁵⁰⁹ page.

If **Calpendo** is hosted on **calpendo.com**, then the SMTP host and emailed base URL will have been configured already. However, it is possible to choose the name that **Calpendo** reminders and Automatic Email appear to come from.

New User Requests

When a new user registers in **Calpendo**, by default **Calpendo** is configured to send an email to administrators. This means any user registered in the system whose roles include the **Admin** role. Who gets [Email Workflow Actions](#)³⁷³ can be changed, but for now it is best to make sure that there is at least one user with the **Admin** role that has a valid email address configured. See [Modifying Users](#)¹⁷⁷ for information on how to do that.

Before Going Live

These are the things that need to be done before you go live, and use **Calpendo** for real:

- Work out what [properties](#)⁶⁵⁵ should exist on a [Project](#)²¹⁷ and [bookings](#)⁶⁵² for different *resources*.
- Work out how [projects](#)⁶⁵⁴ will be approved, and who will assign the *project* a unique [code](#)⁶⁵⁴.
- Use the [Bakery](#)⁵³⁷ to configure the *properties* on a [Project](#)²¹⁷ and then [modify the project template](#)²¹⁸, which is copied each time a new *project* is created.
- Use the *Bakery* to create any required **Booking** sub types and configure their *properties*.
- Create one or two *projects* in **Calpendo**, and configure them to mirror real *projects* to see if all the information required can be specified.
- Work out whether to assign all users a [User Type](#)⁶⁵⁶, and whether to assign all projects a [Project Type](#)⁶⁵⁵. It's much easier to assign the type to users and *projects* as

they are created, so it's better to work out how this is to work before creating users and *projects*. See [Configuring Types And Groups](#)²⁸⁰.

- Identify who will perform day-to-day administration of the system. This means working out who will approve *bookings* or *projects* (if appropriate) and who will users go to if they can't make a *booking* (because **Calpendo** has been configured to limit what they can do). Also, while **Calpendo** will make the management of the shared *resources* much easier, there are always some problems that require intervention to fix. For example, if some equipment breaks down.
- Customise the [FAQ](#)⁵⁰⁶ with questions pertinent to your installation especially who to contact in case of problems.
- Configure user *roles* using the bakery, if you want to add more *roles* to the system. These can be used by [Booking Rules](#)²³⁵, [Permissions](#)³¹⁴, [Workflows](#)³³⁴ and [Manual Emails](#)¹⁸⁶
- [Approve new users](#)¹⁶⁷ as they register or [change the roles](#)¹⁷⁷ bestowed upon users that have already registered.
- Finish [resource configuration](#)²⁸⁴ by choosing **Booking** sub types, locations, types and groups and choosing whether *bookings* for each *resource* require a *project*.
- Set up any [Workflows](#)³³⁴ so that people receive notification about events and add additional functionality to the system.
- Set up [Permissions](#)³¹⁴ to control who can do what, with what and when.
- Create [Time Templates](#)²²⁶ that earmark time for some users or *projects*.
- Create [Booking Rules](#)²³⁵ to control who can make what *bookings*. For example, to limit double *bookings* or the amount of time that can be booked by each user or *project*.
- Create custom [searches](#)¹¹⁸ and [reports](#)¹³³, such as *project usage*⁶⁵⁵ of *resources*, billing etc.
- Customise menus with the [Menu Editor](#)⁴⁸⁹ and by [assigning menus to users](#)¹⁸¹ or setting the [default menus for each user role](#)⁶²⁷.
- Create or modify the [Frequently Asked Questions](#)⁵⁰⁶ that are shown to the users.
- Configure the [global preferences](#)⁵⁰⁹.
- Finally, define Dynamic [Biskits](#)⁶⁵² using the *Bakery* in **Calpendo** to be able to create, read, update and delete any other table required, in the **Calpendo** database, subject to some limitations. This may be required if actual *resource usage* is captured (as opposed to *bookings*), and this needs to be stored in the **Calpendo** database and **Calpendo** used to examine and compare this with *bookings*. Alternatively, use **Calpendo** as a general-purpose data editing engine.

6.3 More Calpendo Configuration

Once **Calpendo** has been configured and it has been used for a while, there may be other functionality that you would like it to provide. Here are some possible examples.

- Do you need to record different information when booking different [resources](#)⁶⁵⁵? Set up a **Booking Sub Type** [Biskit](#)⁶⁵² so different *resources* have different [properties](#)⁶⁵⁵ to be filled in when being booked. See the section on [Setting Up Different Booking Sub Types For Resources](#)²¹⁵ below.
- Do you want to attach documents to some [Biskits](#)⁶⁵²? To do this, set up a *property* or set of *properties* (for more than one document) to hold file attachments. See the section on [Attaching Documents To Biskits](#)²¹⁶ below.
- Do the users need training before using a *resource*? One way to do this is to set up a **Training Biskit Type** in the [Bakery](#)⁵³⁷ and use the **Search Results Booking Rule** to make sure a *booking* cannot be made without the appropriate training record. See the section on [Search Results Booking Rule](#) for ideas on how this might work.

Setting Up Different Booking Sub Types For Resources

In order for *resources* to have different information on their booking forms, it is necessary to set up different **Booking Biskit Types** for those *resources*, known as **Booking Sub Types**.

In order to do this first create a *Biskit Type* that inherits from **Booking** (see [Creating An Inheriting BiskitDef](#)⁶⁰²) and then add the *properties* that are required for that *Biskit Type* (see [Adding Properties](#)⁵⁷⁴).

Once a **Booking Sub Type** has been defined in the *Bakery*, the following will be enabled:

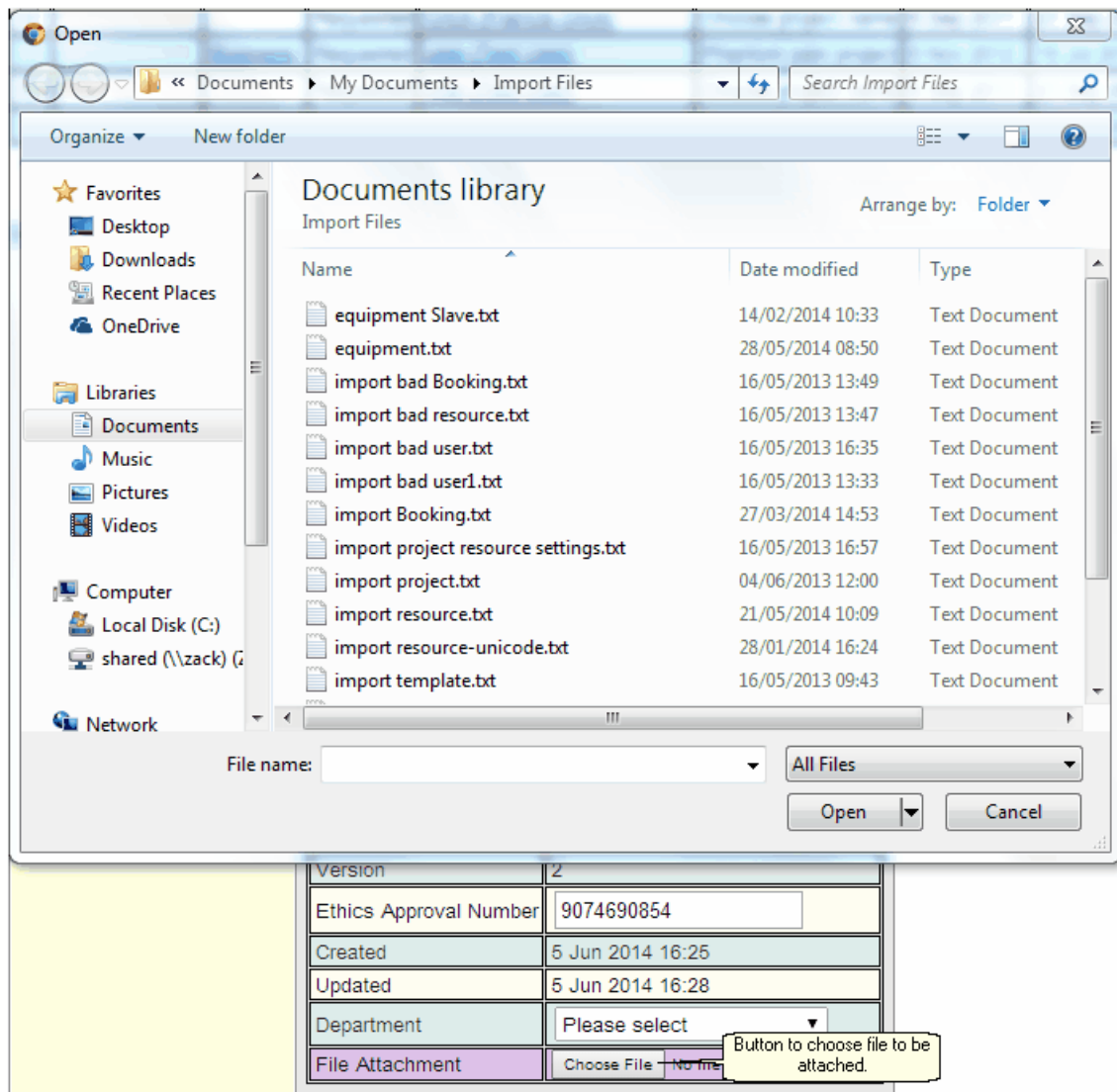
- The **Resource Editor** allows selection of which **Booking Sub Type** should be used for its bookings (see [Configuring Resources And Locations](#)²⁸⁴).
- The **Booking Rule Editor** allows selection of the **Booking Sub Type** that it applies to. Any [Rule](#)⁶⁵² that applies to *bookings* of type **Booking** will also apply to any *bookings* of the **Booking Sub Type**. A *Rule* that applies to a **Booking Sub Type**, will not apply to its parent i.e. **Booking** (see [Choosing Which Bookings A Rule Applies To](#)²³⁸).
- **Booking** searches allows selection of the **Booking Sub Type** to be searched for. Search for **Booking** and all **Booking Sub Types** will be found as well. However, in order to filter the returned *bookings* by a *property* unique to a **Booking Sub Type**, then make the search look only for that **Booking Sub Type**.
- A [Permission](#)⁶⁵⁴ that applies to **Booking** will also apply to any **Booking Sub Type**, although the reverse does not apply. See [How Permissions Work](#)³¹⁴ for details on how *permissions* apply to sub types.

If an administrator wishes to move to **Booking Sub Types** for *resources* with existing *bookings*, then they need to talk to the **Calpendo Support Team** in order to help with the transition including moving across historical data.

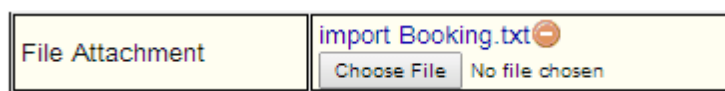
Attaching Documents To Biskits

In order to attach documents to a *Biskit* there needs to be a *property* created on its *Biskit Type* that will hold that document (see [Adding Properties For File Attachments](#)⁶⁸⁰). If a number of documents need to be attached to a *Biskit*, then a number of *properties* could be used or a single *property* of type **Set** (see [Creating A Set Of BiskitDef](#)⁵⁸⁸).

Once the *property* has been added, edit the *Biskit*, and there will be a **Choose File** button, which when pressed will provide a file browser so the file to be attached may be chosen.



Once selected, the file name is shown along with a (red) button to remove the attachment. Pressing **Choose File** will replace the attached file with a new one.



6.4 Converting From An Existing Booking System

If there are already [bookings](#)⁶⁵² and possibly [projects](#)⁶⁵⁴ that you would like **Calpendo** to take over, then the process of migrating to **Calpendo** may feel daunting. The following method is the recommended route:

- Pick a date in the future, maybe 6 to 8 weeks away.
- Any *bookings* for the [resources](#)⁶⁵⁵ beyond that date should be made through **Calpendo**. Any existing *bookings* should be imported into **Calpendo**. This can be done manually, and by picking a date far enough into the future, it should reduce the number of *bookings* sufficiently so that a manual import process is acceptable. If there are too many *bookings* to do this manually, and the existing *bookings* are stored in an electronic system, then it may be possible to automate their import. Email info@calpendo.com for help with this.
- If there are existing *projects*, then create them in **Calpendo**. It is important that the [properties](#)⁶⁵⁵ on a *project* get configured to your requirements first, so that when the *projects* are created, all the information is provided.
- Send an email to everyone who may want to use **Calpendo**, and invite them to register. If everybody is getting a [user type](#)⁶⁵⁶, then it is important to [configure the user types](#)²⁸⁰ before sending this email. This is because when users register, they select their own *user type*. An administrator can check or change the *user type* if required.
- Associate all users with their *projects*. This should be done after you've created the *projects*, and after most of the users have registered. It needs to be done before users start making their own *bookings*. This is because users can choose a *project* they are associated with when making a *booking*. Some or all *resources* can be configured to not require a *project* with their *bookings*.

There may be some apprehension about the process of switching to using **Calpendo**, however those that have converted so far have been pleasantly surprised about how painless it was.

Also, the above is only a suggestion and other methods may well work. For example, if there are multiple *resources* to migrate, it may make sense to migrate them one at a time.

6.5 Project Configuration

One of the most important aspects of configuring **Calpendo** is to make sure that [projects](#)⁶⁵⁴ are set up the way you want. This means making sure that the *projects* store the information required for the facility and that *projects* are approved using a suitable mechanism.

See Also

- [Types And Groups](#)²⁷⁹ to understand what types and groups are, and what the difference between them is.
- [Configuring Types & Groups](#)²⁸⁰ to understand how to go about setting them up for *projects*, as well as for users and [resources](#)⁶⁵⁵.

6.5.1 Project Template

Whenever somebody [creates a new project](#)⁷⁵, **Calpendo** first creates a copy of a [project](#)⁶⁵⁴, referred to as the template *project*. It doesn't have to be a special *project* that is used as the template, but it's generally appropriate that the template *project* is not used for [booking](#)⁶⁵² [resources](#)⁶⁵⁵, since real *projects* usually have lots of information stored in them which is not on the template. For this reason, the template *project* is normally assigned a status of **Unbookable** so that nobody can create a *booking* that references it.

Choose which *project* is used as a template using the [Global Preferences](#)⁵²⁹ [setting for the template project](#)⁵²⁹.

Note that, although [properties](#)⁶⁵⁵ from the template *project* are copied when creating a new *project*, some *properties* are modified. These are:

Property	Description
Name	This is always reset to Unique project name .
Owner	This is changed to the current user.
Users	The current user is <i>added</i> to the list of users.
Status	The status is changed to Requested .
Project Code	This is set to be empty.

The template *project* ignores the need for values to be input for all *properties* that are set up to have required values.

6.5.2 Configuring Project Properties

It seems that everybody has a different idea about what information should be collected in a [project](#)⁶⁵⁴. For example, some facilities will need to collect safety information that is particular to the [resources](#)⁶⁵⁵ they have; if there is an MRI scanner, you will probably want to know what people are intending to do both in the scanner room and possibly outside. Are they going to use lasers, gases, pain devices, collect blood or give drugs? Are they going to take any new equipment into the scanner room, and if so, what? These questions are just examples of things that may be appropriate for one facility and not for others. This is why **Calpendo** allows you to configure your *projects* to contain the [properties](#)⁶⁵⁵ that make sense for your facility.

Some *properties* exist for every **Calpendo** installation, and these are described in [Project Properties](#)¹⁵⁶. For example, every project has a name, status and a list of users that are associated with the *project*.

The *properties* on a new *project* in a new **Calpendo** installation are a default set that may well not be appropriate for your use, and so as well as adding your own required *properties*, it may also be necessary to remove some.

Adding, removing and relabelling *properties* is done in the [Bakery](#)⁵³⁷. *Properties* can be added to [bookings](#)⁶⁵² as well as *projects*, so its possible to always capture the information required.

When creating your own *properties* for *projects* each *property* has a 'required' attribute that can be selected, this will force a user who is creating a new *project* to enter data into this field. For text or numerical *properties* there can also be specified a minimum and/or maximum value. For text *properties*, the minimum and maximum refer to the length of the text provided by the user. This means a user could be required to provide at least 50 characters (or whatever) to describe their *project*.

If a large number of *properties* are being added to the **Project Biskit**⁶⁵² and they need to be displayed in particular groups under specific tab names then use the **Group meta-property**⁶⁵⁴. All *properties* with the same **Group meta-property** will be displayed under the same tab which will be labelled with the value of the **Group meta-property**. An alternative would be to use the [Layout Editor](#)⁶⁰³ to layout the **Project** using a combination of **Horizontal** and **Vertical** tabs.

General	Professor	<input type="text"/>
Project Resource Settings	Professor Sign Off	false ▼
Users	Accounts Sign Off	false ▼
Sign Off	Accounts Liason	<input type="text"/>
Project Groups		

[Permissions](#)³¹⁴ can be used to control who can enter what for a *project*. For example, each *project* has a unique [project code](#)⁶⁵⁴, and it may need to be limited as to who can change that *property*. By setting the [Permissions](#)⁶⁵⁴ in such a way that only the administrators may set the *project code* on a *project*, a user will not even be given the opportunity to enter a *project code* when creating or editing a *project*.

Additional *properties* may also be required if there is going to be a multi-step approval process. This is covered by the next section, [Configuring The Project Approval Process](#)²²⁰.

6.5.3 Configuring The Project Approval Process

This section describes how to set up the [project approval process](#)⁶⁵⁴ that is required. Please make sure to read [The Project Approval Process](#)¹⁵⁸ first, which describes some of the options and how the *approvals process* works from the perspective of the person that does the day-to-day administration of **Calpendo**.

As described in [The Project Approval Process](#)¹⁵⁸, there are two approaches to project approval: the single-step approval or the multi-step approval. This section will not cover the question of why to choose one method over the other, but will instead focus on how to set up each method.

Final Approval: When A Booking Can Use A Project

A [project](#)⁶⁵⁴ can be associated with a [booking](#)⁶⁵² when its [status](#)⁶⁵⁴ is set to **Approved**. This means that changing a *project's status* to **Approved** is the final step of a multi-step *approval process*, and the only step in a single-step *approval process*. The approval itself (that is, changing the *status*) is described in [The Project Requests Page](#)¹⁵⁹.

Single-Step Approval Process

As mentioned above, if a single-step *approval process* is being used, then the *project status* just needs to be set to approve or deny. Nothing special in this case needs to be configured. However, it is possible that **Calpendo** has been pre-configured with some [properties](#)⁶⁵⁵ intended for use in a multi-step *approval process*. Therefore, there may be some *properties* on *projects* that are not required. See [Configuring Project Properties](#)²¹⁸ for information on changing the *properties* on a *project*.

Multi-Step Approval Process

A multi-step *approval process* does not have specific support in **Calpendo**. However, the effect can be achieved by adding properties to **Project** to store the various approval stages. For example, suppose it is required to provide separate approvals for the science, finance and ethics before final approval can be given. In this case, the recommended configuration is as follows:

1. Create boolean (Yes/No) *properties* on **Project** called **scienceApproval**, **financeApproval** and **ethicsApproval** using the [Bakery](#)⁵³⁷ as described by [Configuring Project Properties](#)²¹⁸.
2. Set up [Permissions](#)³¹⁴ so that only the desired people can set **scienceApproval**, **financeApproval** and **ethicsApproval** to true on any *project*.
3. Set up [Permissions](#)⁶⁵⁴ so that only the desired people can modify the *project status* to **Approved** and only then when **scienceApproval**, **financeApproval** and **ethicsApproval** are already set to **True**.
4. Set up *Permissions* so that only the desired people can modify the *project status* to **Denied** but that this can be done regardless of **scienceApproval**, **financeApproval** and **ethicsApproval**.

If the approvals need to be given in a particular order, then configure this using the *Permissions*, so that whoever gives the financial approval may only do so once the ethics approval has been given, for example.

It would be a good idea to also set up [Workflows](#)³³⁴ to help notify people when it's time for them to give their approval. To do this, make sure that there are a set of [conditions](#)⁶⁵³ on the [Email Workflow Action](#)³⁷³ so that it is triggered by a change that means the *project* is now in a state suitable for them to give their approval.

6.6 Controlling Bookings

Calpendo offers three ways to control [bookings](#)⁶⁵². This section describes how to choose between the three, [Time Templates](#)²²⁶, [Booking Rules](#)²³⁵ and [Permissions](#)³¹⁴. It also describes [Time Templates](#)⁶⁵⁶ and [Booking Rules](#)⁶⁵².

The Automatic Booking Approval Process

First some preferences that are set up globally that will affect the process:

Default [booking status](#)⁶⁵³ set up in [Global Preferences->Bookings->Default Booking Status](#)⁵¹⁵,

1. Regular users the default value is **Requested**.
2. **Admin** users the default value is **Approved**

Also by using [Global Preferences](#)⁵⁰⁹ it is possible to specify that [Time Templates](#)⁵³⁵ and [Booking Rules](#)⁵³¹ do not apply to **Admin** users.

The process:

1. When a user creates a *booking*, they get a *booking* pop up with the *booking status* set to **Best Possible**.
2. The user may change the default *booking status* for that *booking* if they have [permission](#)⁶⁵⁴ to do so.
3. Once the user presses the **Create Booking** button, the **Calpendo** Client will check the *booking* against any [Time Template](#)⁶⁵⁶ that exists for that time period and [resource](#)⁶⁵⁵:
 - a. If the *booking* is automatically approved the *booking status* is set to **Approved** and the [property](#)⁶⁵⁵ **Approved by Template** is set to *True*.
 - b. If not **Automatically Denied** it will keep the *booking status* chosen in the pop up when the *booking* was created.

The full list of possible outcomes is in the table below (if there is no outcome mentioned, that combination of chosen *booking status* and **Default Booking Status** are not possible):

Template Response	Initial Chosen Booking Status	Final Booking Status Default Booking Status = Requested	Final Booking Status Default Booking Status = Approved
No Template	Best Possible	Requested	Approved
	Requested	Requested	Requested
	Approved		Approved
Warning Template	Best Possible	Requested	Requested
	Requested	Requested	Requested
	Approved		Approved by Template
Acceptable	Best Possible	Requested	Requested
	Requested	Requested	Requested
	Approved		Approved by Template
Auto-Approval	Best Possible	Approved by Template	Approved by Template
	Requested	Approved by Template	Approved by Template
	Approved		Approved by Template
Auto-Denial	Any	Denied	Denied

If the *booking* is not automatically denied it is then passed to the server and depending on the current *booking status* (and **Approved by Template** property) more checks are done:

a. **Requested:**

- i. the server checks against the *Time Template* again and any messages associated with that *Time Template's* target group is displayed
- ii. the *Booking Rules* are checked, and if the *booking* is denied a customised message may then be displayed
- iii. the *Permissions* are checked, and if the *booking* is denied the **Permission Denied** message will then be displayed
- iv. The **Calpendo** administrator will now have to manually approve the *booking*.

b. **Approved by Template:**

- i. the server checks against the *Time Template* again and any messages associated with that *Time Template's* target group is displayed
- ii. the *Booking Rules* are checked, and if the *booking* is denied a customised message may then be displayed
- iii. the *Permissions* are checked, and if the *booking* is denied the **Permission Denied** message will then be displayed

- c. **Approved**, (only for Users with a **Default Booking Status** of **Approved** who do not have a *Time Template* applying to them)
 - i. the *Booking Rules* are checked, and if the *booking* is denied a customised message may then be displayed (this is optional as **Admin** users may be set up to not have *Booking Rules* applying to them)
 - ii. *Permissions* are checked, and if the *booking* is denied the **Permission Denied** message will then be displayed

The *booking* is now approved.

See Also: [Permissions](#)³¹⁴

6.6.1 Choosing Between Time Templates, Booking Rules and Permissions

Calpendo allows control of the *bookings*⁶⁵² that people are allowed to make using [Time Templates](#)²²⁶, [Booking Rules](#)²³⁵ and [Permissions](#)³¹⁴. Some controls to be implemented could be done using any of these three, so it's important to understand the advantages and abilities of each method so that the most appropriate methodology is used to achieve what is required.

The general rule of thumb is to preferably implement controls using [Time Templates](#)⁶⁵⁶ where possible. This is because the *Time Templates* can be displayed on the background of the [Bookings Calendar](#)⁶⁵² so that users will be able to see at a glance when they will be allowed to make a *booking*. However, *Time Templates* only allows the placement of controls on the times when *bookings* can be made, whereas [Booking Rules](#)⁶⁵² and [Permissions](#)⁶⁵⁴ offer control in other ways.

Let's look at each option with an example that can be implemented in all three ways. Suppose it is necessary to stop a particular user from making *bookings* on Mondays. Here is an outline of how to implement this in each methodology:

Time Templates Example

Use the **Time Template Editor** to create a *Time Template* that applies to the person (or people) to be controlled, and set the *bookings* acceptability to **Automatic Denial**. Also specify a message that would be given to the user if they tried to make a *booking* in the forbidden time. Once the *Time Template* is created, then choose the times to apply it. For this, go to the *Time Templates* page and create a [repeating](#)⁶⁵⁵ entry, and apply the *Time Template* every Monday.

When the user logged in, they would then see the background of every Monday in the *bookings calendar* in red, and letting the mouse hover over the red background would show them the message that has been entered. If they tried to create a *booking* on a Monday, then as soon as they click on a Monday, they would be given the error message. This means *Time Templates* can give instant feedback to the user, before they even enter the details of the *booking*.

Booking Rules Example

To stop somebody booking on Mondays using *Booking Rules*, use the [Booking Rule Editor](#)²³⁵ to create a [Simple Booking Rule](#)²⁴³. Specify that the *Booking Rule* applies to the person or people you want to control, using the **Bookers** tab within the **Applies To** tab. In the **Conditions** tab, specify a [condition](#)⁶⁵³ that the *booking's* start day is a Monday. Next, specify the *Booking Rule's* **Rejection Type** as **Reject** and enter a message to be shown to the user.

When the user logged in, they would now see no indication that they cannot make a *booking* on a Monday. Only after entering the details of a *booking* they wanted to make would they see the error message that was set up in the *Booking Rule*.

Permissions Example

To deny somebody *permission booking* on Mondays, you would create a *Permission* that applied to *Bookings*, add a *Condition* the same as for *Booking Rules* that the *booking's* start day is a Monday, and use the **Applies To** tab to specify the users that the *Permission* [applies to](#)⁶⁵².

When a user tries to make a *booking* that is denied by a *Permission*, then they would see an error message that says **Permission Denied**. There is no way to provide your own error message to be shown when using *Permissions*. Consequently, this is the least user-friendly way of controlling *bookings* and so should be used sparingly.

From reading the examples above, it may appear that there's not a huge difference between the three methods. However, they differ in the information that they can each use, and in the results they can effect:

Time Templates: Information In And Out

Time Templates easily allows a choice of times when bookings should be requests, automatically approved, given a warning or automatically denied. They also allow decisions based on who makes the *booking*, the [project](#)⁶⁵⁴ it's for, and the [resource](#)⁶⁵⁵ it's for. No other information can be used by *Time Templates*.

Booking Rules: Information In And Out

Booking Rules can be configured to use *any* information. There are different types of *Booking Rules* that make it relatively easy to apply some controls, for example to prevent double *bookings* or limit how much time somebody can book in a week. However, a *Booking Rule* can be made to do almost anything by creating an [Advanced Booking Rule](#)²⁶⁹ (although this requires a programmer to write some Java code).

Permissions: Information In And Out

Permissions can only use the information in the *booking's* [properties](#)⁶⁵⁵, as well as information about who was making the *booking* and when. There is no access to other information in the **Calpendo** database. For example, *Permissions* cannot be used to control the time booked per week or to prevent double *bookings* because that would require information about other *bookings*.

Comparing Time Templates, Booking Rules And Permissions

Comparing the methods gives us these differences:

- They each provide different levels of feedback to the user. *Time Templates* provide graphical feedback before a user tries to make a *booking*, and can stop a user from even entering a *booking's* details by providing a message you have specified in the *Time Template*. *Booking Rules* provide no graphical feedback and can only give an error message after the *booking* details have been entered, but the message that will be provided can be chosen. *Permissions* also provide no graphical feedback, and can only give a generic **Permission Denied** error message after the *booking's* details have been entered.
- A single *Time Template* can provide a different message to different groups of people. *Booking Rules* and *Permissions* can each only apply to one set of people.
- *Time Templates* can be used to choose whether, when and which *bookings* should be made as a request or automatically approved. *Booking Rules* and *Permissions* can only deny or accept a booking. For example, *Booking Rules* and *Permissions* cannot choose to automatically approve a *booking*.
- *Time Templates* are the most convenient way to provide a warning for a specific period of time. For example, if a user is making a *booking* for an MRI scanner, and there are no operators available one particular week due to annual leave, there could be a *Time Template* that applies during that week to give users a warning that they need to make their own arrangements for a scanner operator to be available. While a warning message can be provided using *Booking Rules*, it's not as flexible in setting the start and finish time it should apply, and it provides no feedback to the user until after they make their *booking*.
- *Time Templates* work when the control to be applied relates to when the *booking* is made. *Permissions* apply to any *property* of the *booking*, while *Booking Rules* can use any information even if it's not on the *booking*. For example, *Time Templates* and *Permissions* cannot be used to prevent double *bookings*, or to limit the amount of time somebody can book per week.
- *Time Templates* are simpler than *Booking Rules* and *Permissions*.
- [Permissions](#)³¹⁴ are the most difficult of all three techniques to understand. This is because many *Permissions* may be set up, some of which give *permission* and some of which may refuse *permission*. **Calpendo** then chooses which *Permission* is the definitive one from all that may match depending on their relative priority. This means they can be confusing because if many of them are created, it's not always obvious which one will be definitive.

In summary:

1. If you can achieve what you want with a *Time Template*, then you should use that.
2. If the control needed only requires information that can be found on the *booking*, or the time the *booking* is being made, and it is not a problem that the user is given a generic **Permission Denied** error message, then it is possible to use a *Permission*.
3. If the above don't apply, then a *Booking Rule* is needed.

6.6.2 Configuring Time Templates

[Time Templates](#)⁶⁵⁶ let you assign an acceptability rating to [bookings](#)⁶⁵² depending on when they are booked, who makes the *booking* and the project the *booking* is for. This is done in two separate parts:

1. Create one or more specifications of acceptability according to who makes the *booking* and the [project](#)⁶⁵⁴, but ignoring when the *bookings* are for.
2. Apply those specifications to various periods of time.

The first part is done by the **Time Template Editor** and the second part is done on the [Time Templates Calendar](#)⁶⁵⁶.

6.6.2.1 How Time Templates Work

New Laboratory Example

Calpendo's [Time Templates](#)⁶⁵⁶ which are created in the [Time Templates Editor](#)²²⁹ are quite powerful, and best explained by an example. Suppose a new laboratory opens with a scanner, and the building is only open from 7am to 7pm, Monday to Friday. Since the laboratory is new, its scanner doesn't get much use yet, so there is no need to enforce much in the way of [booking](#)⁶⁵² controls, but it is required to prevent people from making out of hours *bookings*. To do this, create two [Time Templates](#)⁶⁵⁶: one for when the building is open, and another for when it is closed.

The two *Time Templates* are represented by the contents of this table:

Time Template Name	Applies To	Acceptability	Message
Open	Everybody	Automatic Approval	
Closed	Everybody	Automatic Denial	No <i>booking</i> at this time because the building is closed

This means that the **Open Time Template** allows everybody to make *bookings*, and those *bookings* will be automatically approved by **Calpendo**. The **Closed Time Template** stops everybody from making *bookings*, and whenever somebody tries, they will be given the message shown in the table about the building being closed.

The missing element from all this is to choose the times when the **Open** and **Closed Time Templates** apply. This is a process much like the [Bookings Calendar](#)³⁹, but instead this is a calendar where you choose the times and [resources](#)⁶⁵⁵ that you want to apply to *Time Templates*. Create an entry on Monday from midnight to 7am that applies the **Closed Time Template**, another entry from 7am to 7pm that applies the **Open Time Template**, and another entry from 7pm to midnight that applies to **Closed Time Template**. In each case, you make them [repeat](#)⁶⁵⁵ weekly for each day Monday to Friday. Finally, create an entry on Saturday and Sunday that applies the **Closed Time Template** all day long.

Exceptions To The Rule

Suppose that the scanner requires trained operators to use it, and that one or more operators are on holiday for a week. Anybody who makes a *booking* during that week will need to ensure that somebody suitably trained will be present. In this case, people are allowed to make *bookings*, but its needed to make sure they know that this problem exists. To handle this, create another *Time Template* that looks like this:

Time Template Name	Applies To	Acceptability	Message
Operator Shortage	Everybody	Automatic Approval	There is a shortage of scanner operators at this time. Please make sure operator will be present when you use the scanner.

Then add entries to the [Time Templates Calendar](#)⁶⁵⁶ that applies the **Operator Shortage** *Time Template* for the week in question, from 7am to 7pm each day. Now, when somebody tries to make a *booking* during this week, and during regular opening hours, there are two *Time Templates* that apply: **Open** and **Operator Shortage**. When **Calpendo** sees multiple *Time Templates* that apply, it always enforces the most restrictive one. In this case, both *Time Templates* have the same level of acceptability, but only one has a message attached to it, **Operator Shortage**, and so this is the one that will apply. This means that *bookings* made in this week will still be automatically approved, but will show the message about the shortage of scanner operators.

If required, create the **Operator Shortage** *Time Template* so that its acceptability is *Warning* instead of *Automatic Approval*. By doing this, two things will change:

1. *Bookings* would be created with a default [booking status](#)⁶⁵³ as defined by **Global Preferences->Bookings->Default Booking Status** which by default is **Requested** instead of **Approved** for normal users, so that they should be [manually approved](#)¹⁵¹, and **Approved** for **Admin** users.
2. *Bookings* would record the fact that the user making the *booking* had been warned

Indeterminate Templates

If a user has multiple projects, then it's quite possible that booking for one project would be approved, while booking for another project would be denied, depending on how the templates have been defined.

In case like this, it is indeterminate whether a booking would be approved or denied until a particular project is chosen.

To account for this, the background of the bookings calendar displays a different colour to indicate this "indeterminate" status.

No Projects To Book With

If a user has no projects, then they cannot make a booking for a resource that requires a project.

Consequently, the background colouring of the bookings calendar shows a message indicating that they can't book, and sets a colour accordingly.

In Summary

- Information within a Template Group is combined to make the check, so you would have to be both the user Type and in the User Group.
- When applying a single template to a single booking, it's the most lenient of the applicable groups that will apply.
- When multiple templates apply to a single booking, it the most restrictive of those templates that applies.
- When a user has multiple projects that could be used to make a booking, and those projects would yield a different template result, there is now the option of displaying an indication of it being indeterminate (until the project is chosen) or of displaying the least restrictive of those that apply.

6.6.2.2 The Time Template Editor

The **Time Template Editor** allows the specification of the acceptability of [bookings](#)⁶⁵² according to who makes the *booking* and the [project](#)⁶⁵⁴ it is for, but without worrying about the periods of time when those specifications apply.

The editor is, by default, found on the menu as **Admin-->Template Editor**.

Each item that the editor lets you create consists of the following:

Property	Description	
Name	This is the name used to represent the Time Template ⁶⁵⁶ . It can be anything required, and will be shown on The Time Templates Calendar ²³² .	
Colour	This indicates the colour that will be used to represent the <i>Time Template</i> when shown on The Time Templates Calendar ²³² .	
Time Template target groups	This is a list of items, each of which identifies a users and <i>projects</i> , and assigns them an acceptability and a message to be shown to the user:	
	Property	Description
	Name	This is the name used to represent this selection of users and <i>projects</i> .
	Applies To	This specifies who this <i>Time Template</i> target group applies to.
	Acceptability	This is where to specify the acceptability for <i>bookings</i> that fall into this <i>Time Template</i> target group.
	Message	This is a message that should be delivered to the user whenever they attempt to create or update a <i>booking</i> that matches this <i>Time Template</i> target group. If a message is not specified, in the <i>Time Template</i> , and the <i>Time Template</i> 's acceptability indicates the user should be given a warning, then they are presented with the default warning message.

This table shows what each of the *acceptability* options mean:

Acceptability	Description
Automatic Approval	This indicates that a <i>booking</i> will be automatically approved. As such, the booking status ⁶⁵³ will be set to Approved so that nobody has to manually approve a <i>booking</i> request.
Acceptable	Indicates that the <i>booking</i> may be created, and it will be given the <i>status</i> Requested .
Warning	Indicates that the <i>booking</i> may be created, and it will be given the <i>status</i> requested by the user the same as Acceptable , but the user will be warned that something is wrong, or that they may need to do something in particular. For example, it may be required to warn them that there are no operators available and they need to provide their own. Whenever a user is warned, the <i>booking</i> records that fact. The only difference between Acceptable and Warning is that Warning records the fact that the user was warned on the <i>booking</i> . Acceptable and Automatic Approval <i>bookings</i> may also provide a message to the user if you want.
Automatic Denial	Indicates that the <i>booking</i> is not permitted. No <i>booking</i> will be created, and an error message should be provided so that it can be shown to the user.

This is what the **Time Template Editor** looks like once opened up.

In the left hand pane is a list of all the *Time Templates* created so far.

In the right hand pane will be the currently selected *Time Template*. The following example is a *Time Template* that is not in edit mode.

The screenshot shows the Time Template Editor interface. On the left is a list of templates: Biology Time, Physics Time, Prohibited Time, Free for all, Physics Auto Approve, Far term, Approve for all, and Building access warning for all. A callout points to this list: "The Templates you have created". The main area shows the details for the "Biology Time" template. A callout points to the template name: "The Template you are currently working on". The details include: Name: Biology Time, Colour: (empty), Target Groups: Everybody, Biology, Physics. A callout points to the Target Groups: "The users this group will apply to". The Applies To section has a table with columns for User Roles, User Type, User Group, Project Type, and Project Group, all set to "Any". A callout points to the Booking Acceptability dropdown, which is set to "Automatic denial": "Booking Acceptability for this group". A callout points to the Message field: "Message to appear when an attempted booking matches this template target group." The Menu Bar at the top includes: Refresh, Open All, Close All, Edit, Create, Create copy, Delete, References, History.

Each *Time Template* can have a number of target groups, each group will have a list of the users or *projects* that the group will apply to with the *booking Acceptability* that will be applied to that group, and a message if an attempted booking has matched the target group. If an attempted *booking* matches more than one target group within a *Time Template* then the least restrictive group will apply.

If a *booking* matches multiple *Time Templates* then the most restrictive *Time Template* will apply.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

The following is a *Time Template* in edit mode.

The screenshot shows the 'Biology Time' template in edit mode. The sidebar on the left lists various templates, with 'Biology Time' selected. The main form has a 'Name' field set to 'Biology Time' and a 'Colour' field. Below these are 'Add', 'Remove', and 'Create Copy' buttons. A list of target groups is shown, with 'Biology' selected. The 'Applies To' section contains a table with fields for 'User Roles', 'User Type', 'User Group', 'Project Type', and 'Project Group'. The 'Acceptability' section has a dropdown menu with options: 'Acceptable', 'Automatic approval', 'Warning', and 'Automatic denial'. The 'Message' field is empty.

Callouts in the image:

- 'Colour of template' points to the 'Colour' field.
- 'Choose logic option' points to the 'Applies To' section.
- 'Users in target group' points to the 'Biology' target group in the list.
- 'How any booking will be dealt with.' points to the 'Acceptability' dropdown menu.

For a target group choose the users to constitute the group. If multiple **Applies to** are set up then any user that complies with all the restrictions is a member of the group. For a user that is in multiple target groups (in the example above, all users will be in the everyone target group, and those with a *User Type*⁶⁵⁶ of **Biology** will be in the **Biology** target group) the **Acceptability** that is chosen is the one with the least restrictions. (i.e. in this order **Automatic Approval**, **Warning**, **Automatic Denial**).

Any target group that gets an **Automatic Denial** or **Warning**, should have a message set up, although if there is no user defined message an automatic one will be generated by **Calpendo**.

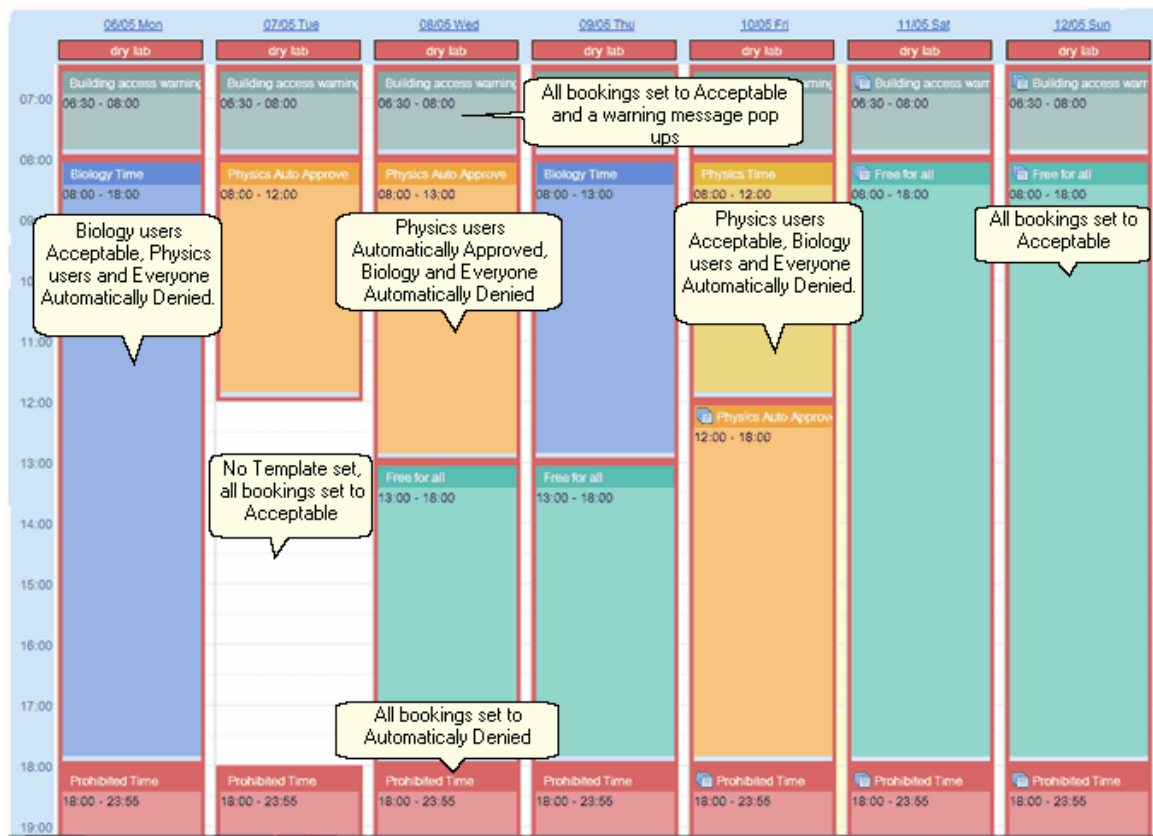
Once the target groups are set up for a *Time Template*, click on the **Save** button to save the new *Time Template*, or **Cancel** to discard.

6.6.2.3 The Time Templates Calendar

The [Time Templates Calendar](#)⁶⁵⁶ works much like the [Bookings Calendar](#)⁶⁵². Create a [Time Template](#)⁶⁵⁶ entry on the *calendar*, and specify the time period that it applies for. Also it is possible to set the *Time Template* entry to [repeat](#)⁶⁵⁵. For example, to earmark Tuesday mornings as **Physics Auto Approve**, then create a *Time Template* entry on a Tuesday morning, set it to use the **Physics Auto Approve Time Template** and make it *repeat* every week.

Here is an example of *Time Templates* set up for a [resource](#)⁶⁵⁵ over a single week. A *Time Template* can choose one of four possible states for a *booking* request:

1. **Automatic Approval**
2. **Acceptable**, [bookings](#)⁶⁵² will be given the status **Requested**.
3. **Warning**, *bookings* will be given the status **Requested** but a warning message will be issued.
4. **Automatic Denial**



The *Time Templates Calendar* works in very much the same way as the *Bookings Calendar*. Read the section on [Exploring the Bookings Calendar](#)⁴⁰ for more details on navigation, [bookmarks](#)⁶⁵³, choosing *resources* to display and day/week/month views.

In the above example what is described is the medium term *Time Template* affect for each *Time Template*. As described in [How Time Templates Work](#)²²⁶ each *Time Template* may also has have a [near](#)⁶⁵⁴ and [far term Time Templates](#)⁶⁵³ configured.

In order to view this information in the *Time Templates Calendar* move the mouse over a *Time Template* and a pop up will appear which will show the complete definition of the *Time Template*.

The screenshot shows a pop-up window for a Time Template named "Physics Only" with a time range of 08:00 - 17:00. The window is divided into two main sections: a left sidebar and a right main content area.

Left Sidebar:

- Physics Only
- 08:00 - 17:00
- Which term (Near, Medium, or Far) currently applies
- Template (medium term) definition
- Near term definition
- Far term definition

Main Content Area:

- Start** 3 Feb 2014 08:00
- Finish** 3 Feb 2014 17:00
- Resource** Dry Lab
- Resource Type** Room
- Enabled** Yes
- Repeat** Repeat every Monday
- Abdicated** Not abdicated
- Status** Medium term applies now
- Template** Physics Only
- Physics Automatic approval
- Others Automatic denial
- Near term** 1 days
- Near term template** Free for All
- Rolling transition
- Everybody Acceptable
- Far term** 6 weeks
- Far term template** Future
- Instant transition
- Everybody Automatic denial

Putting Time Templates On The Calendar

To put a *Time Template* on the *calendar*, in the *Time Templates Calendar*, navigate to the date that required for the *Time Template* to appear. Then either single-click in the *calendar* at the time when the *Time Template* needs to start, or click-and-drag to select both the start and end times of the *Time Template*. Both methods will cause the *Time Template* pop-up to be displayed:

This pop-up is where a user will create, edit and view the details of a *Time Template*.

Every *Time Template* must have a *resource*, therefore select which *resource* you want to make a *Time Template* for. Use the drop down arrow to choose the *resource*.

Then adjust the date and times of the *Time Template*.

Decide whether the *Time Template* is to be enabled, disabled *Time Templates* are displayed in the *calendar* but in a lighter colour.

Set up the *repeat* structure for the *Time Template*. For more information on setting up repeats read the section on [Repeat Bookings](#) ⁵⁷.

Select the *Time Template* required for the medium term.

Select the *Time Template* required for the *Near Term*

Define what is meant by *Near Term*

Decide whether the transition to the new *Time Template* will be **Rolling** or **Instant**. For example if the *Time Template* runs from 08:00 to 20:30, with **Instant** as soon as 08:00 is reached the whole time period becomes available under the *Near Term Template*, with **Rolling** the *Near Term Template* becomes available as the time roles through it. Therefore at 10:00, 08:00 to 10:00 is available under *Near Term* but the rest of the *Time Template* is still the Medium Term

Select the *Time Template* required for the *Far Term*

Define what is meant by *Far Term*

Decide whether the transition to the new *Time Template* will be **Rolling** or **Instant**.

Choose a message that will be unique for this particular instance of a *Time Template*. If there is a message defined in the original *TimeTemplate* definition then these two messages will be concatenated together with this message being first.

Once all the details have been entered then press **Create Template**, then the new *Time Template* will be sent to the **Calpendo** server.

Time Templates can be copied and pasted, for fast entry of similar *Time Templates*. Just click on a *Time Template* in the *calendar* and select **Copy** from the drop down menu. Then, click somewhere to create a new *Time Template*, if there's a *Time Template* in the clipboard, its content will be used to populate the *Time Template* pop-up. You can then edit it as required.

Editing Time Templates

Read the chapter on [Editing Bookings](#)⁵⁸ to see how to edit *Time Templates* on the *calendar*.

6.6.3 Configuring Booking Rules

Whenever a [booking](#)⁶⁵² is created or changed, **Calpendo** will consult its [Booking Rules](#)⁶⁵² to see whether to accept or reject the *booking*, or else give the user a warning about it.

6.6.3.1 How Booking Rules Work

Every time somebody creates or changes a [booking](#)⁶⁵², **Calpendo** checks its *Booking Rules* to see whether the change should be allowed. Unlike [Time Templates](#)⁶⁵⁶ which are applied in the user's web browser, *Booking Rules* are applied in the **Calpendo** server. This key difference has the following consequences:

- *Time Templates* can give feedback to the user before they even try to create a *booking*, whereas *Booking Rules* require the user to enter the *booking* details before any feedback can be given.
- *Time Templates* have limited information available to them, whereas *Booking Rules* have full access to everything in the **Calpendo** database, and even external information available on the internet as well if you need it.

When **Calpendo** checks its *Booking Rules*, it first looks to see which ones apply to the *booking* and the change being made. There are different [types of Booking Rules](#)²⁴³, but they all share the same way of identifying which *bookings* they [apply to](#)⁶⁵², and this is described in the next section, [Choosing Which Bookings A Rule Applies To](#)²³⁸.

Each *Booking Rule* has to decide whether the *booking* should be rejected or whether it is acceptable. If acceptable, then it must also decide whether to issue the user a warning. When a warning or rejection is issued, the *Booking Rule* responds with two extra things:

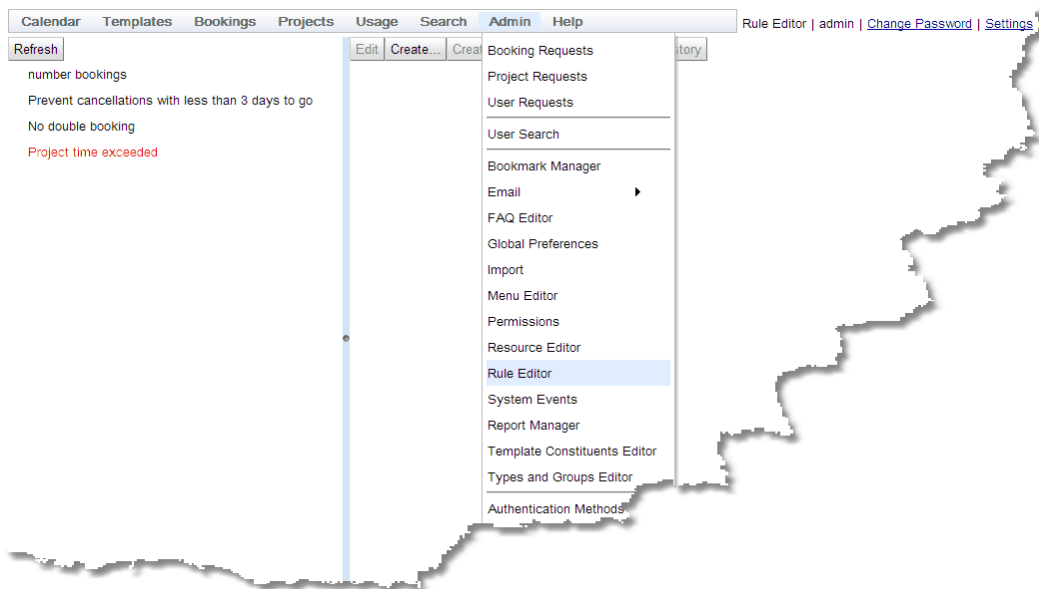
- a message that should be given to the user, preferably customised so it is specific to the problem it has encountered.

- for *bookings* whose [status](#)⁶⁵³ is **Approved**, the *Booking Rule* can indicate whether it might be acceptable if the *booking's status* were **Requested** instead. This allows the user the opportunity to downgrade their *booking* to a request, although the option should only be given if the *Booking Rule* would generate a different response to a *booking* request.

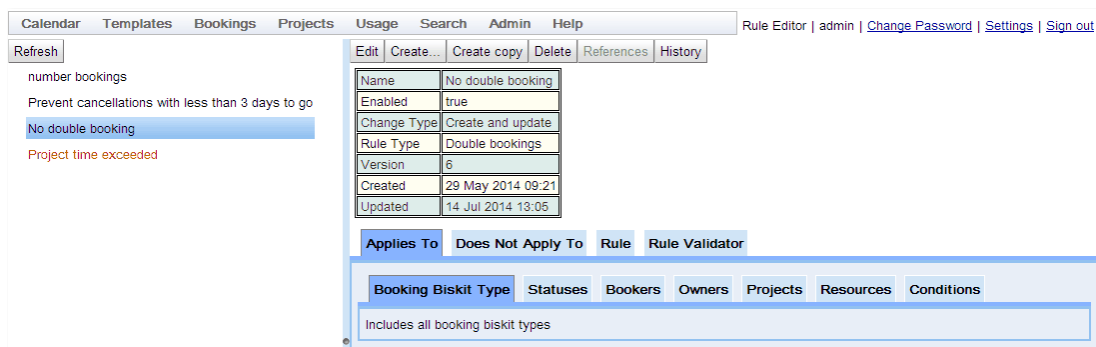
If **Calpendo** finds a *Booking Rule* that rejects the change, then no further *Booking Rules* are run. However, if a *Booking Rule* is found to generate a warning, then further *Booking Rules* are run, just in case there is a *Booking Rule* that would reject the *booking*. Also, **Calpendo** will only show one message to the user, so the order in which the *Booking Rules* run can affect any error message the user is given. The order in which *Booking Rules* are run is controlled by the order they appear in the **Rule Editor**.

6.6.3.2 The Booking Rules Editor

The **Booking Rules Editor** shows all the [Booking Rules](#)⁶⁵² and gives the ability to create, update and delete them. By default, it appears on the menu here:

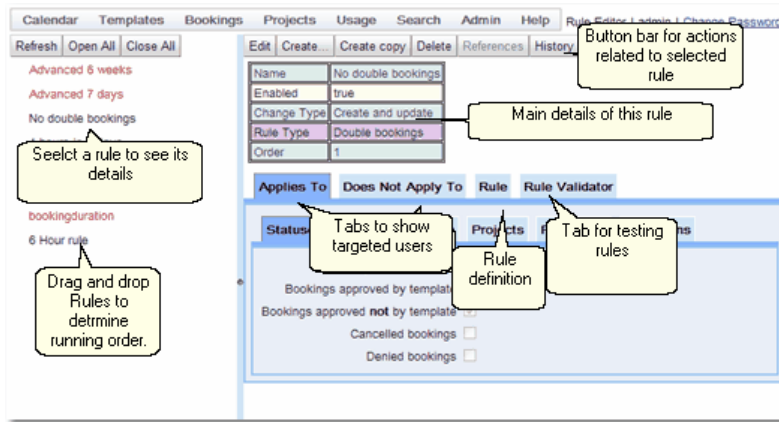


This is what the **Booking Rules Editor** looks like when first opened, disabled *Booking Rules* will be shown in red.



Booking Rule Details

Click on a [Booking Rule](#)⁶⁵² in the list, to see its details appear on the right:



For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Name	Prevent cancellations with less than 3 days to go
Enabled	true
Change Type	Update
Rule Type	Simple

A *Booking Rule* consists of the following:

Name

The name is only used to provide a way for whoever configures or examines *Booking Rules* to understand the purpose of each *Booking Rule*.

Enabled

The **Enabled** flag is a way for you to create a *Booking Rule* without making it take effect immediately, or to turn one off without deleting it. A disabled *Booking Rule* shows in red in the tree.

Change Type

What type of change to the [booking](#)⁶⁵² will fire this *Booking Rule*. There are three options: **Create**, **Update** and **Create and Update**.

Booking Rule Type

Which [Booking Rule Type](#)⁶⁵³ will be used. See [Types of Booking Rule](#)²⁴³ for a full list of the *Booking Rule Types*.

Finally the order the *Booking Rules* will run in is determined by their order in list of *Rules* on the left hand side of the **Rule Editor** starting at the top. To change this order drag and drop a *Rule* to its new position in the list.

6.6.3.3 Choosing Which Bookings A Rule Applies To

This is split into two sections:

[Applies To](#)⁶⁵²: Determines the [bookings](#)⁶⁵² that the [Booking Rule](#)⁶⁵² applies to.

[Does Not Apply To](#)⁶⁵³: Determines the *bookings* that the *Booking Rule* does not apply to.

In addition some rules have the option to define the *bookings* to be counted or ignored when applying the Rule. These are normally called *Bookings To Count/Bookings To Ignore* but may have slightly different names for different Rules.


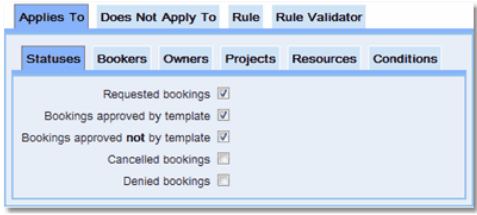
By default a *Booking Rule* will apply to all *bookings*, so if there is nothing in both *Applies To* and *Does Not Apply To* then the *Booking Rule* will apply to all *bookings*. Therefore using *Does Not Apply To* only the *bookings* the *Booking Rule* does not apply to can be specified. Also combine the two types and a *Booking Rule* will apply to a *booking* as long as its accepted by the *Applies To* but not accepted by the *Does Not Apply To*. Of course *Applies To* can be used by itself to figure out which *bookings* a *Booking Rule* will apply to.

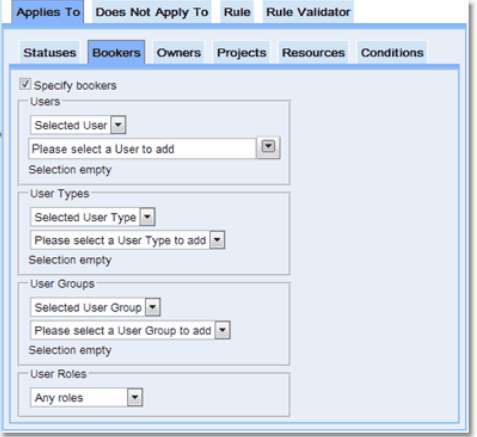
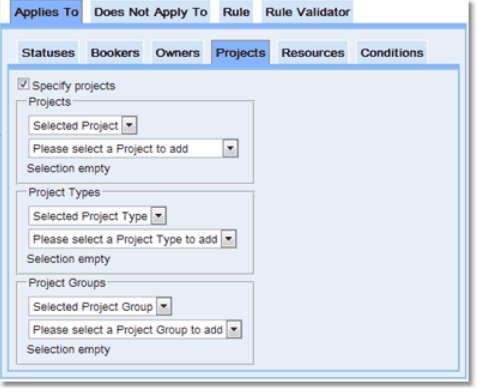
Bookings to Count/Ignore work in a similar way to *Applies To/Does Not Apply To* in that bookings are counted using the definition in *Bookings To Count*, as long as they do not also fall under the *Bookings To Ignore* definition.


Applies To

When running *Booking Rules*, **Calpendo** must decide which *Booking Rules* apply for a given *booking* change that a user is trying to make. To do this, **Calpendo** first looks to see which *Booking Rules* are marked as **enabled**. Then, it looks at whether the *Booking Rule* says it applies when a *booking* is created, when a *booking* is updated, or both.

Beyond those simple [properties](#)⁶⁵⁵, every type of *Booking Rule* provides additional tabs in which you can identify the *bookings* it should apply to as follows:

Tab	Description	
Booking Biskit Type	An administrator can define a number of Booking Biskit Types ⁶⁵² to allow for different <i>booking</i> information to be stored for resources ⁶⁵⁵ . This tab allows the administrator to decide which of the Booking Biskit Types the <i>Booking Rule</i> will work with. A <i>Booking Rule</i> will also work on any Booking Bisket Type that inherits from the selected one.	
Statuses	Every <i>booking</i> has a status ⁶⁵³ which is one of Requested , Approved , Cancelled or Denied . For those <i>bookings</i> whose <i>status</i> is Approved , Calpendo also remembers whether the booking was approved by a Time Template ⁶⁵⁶ , or approved manually. The Statuses tab allows a choice of which <i>booking statuses</i> this <i>Booking</i>	

	<p>Rule should apply to, and if approved bookings⁶⁵² are required, whether to apply to <i>bookings</i> approved by <i>Time Template</i>, manually or both.</p>	
Bookers	<p>The Bookers tab specifies for which registered booker⁶⁵² of these <i>bookings</i> you want this <i>Booking Rule</i> to apply to. First, tick the Specify Bookers box in order to activate this tab. Once activated specify the <i>booker</i> by identifying particular users, user types⁶⁵⁶, user groups⁶⁵⁶ or user roles⁶⁵⁶.</p> <p>If the <i>bookers</i> to be accepted are specified using a mixture of specified users, <i>user types</i>, <i>user groups</i> or <i>user roles</i>, then as long as the <i>booking</i> being changed has a <i>booker</i> that passes any of the <i>booker</i> specifications, then the <i>booker</i> will be accepted.</p>	
Owners	<p>This works exactly the same as <i>booker</i>, but specifies who owns the <i>bookings</i> the <i>Booking Rule</i> is to apply to.</p> <p>This works exactly the same as Bookers, but specifies who owns the <i>bookings</i> the <i>Booking Rule</i> is to apply to.</p>	
Projects	<p>Very much like Bookers and Owners, this tab specifies the projects⁶⁵⁴ a <i>booking</i> must have been made for in order for this <i>Booking Rule</i> to apply, by giving specific <i>projects</i>, project types⁶⁵⁵ and/or project groups⁶⁵⁴.</p>	
Resources	<p>This works in a similar fashion to <i>projects</i> but it specifies the resources⁶⁵⁵ the <i>booking</i> will apply to by giving specific <i>resources</i>, resource types⁶⁵⁵ and/or resource groups⁶⁵⁵.</p>	

Tab	Description	
Conditions	<p>The Conditions tab works just like the Conditions¹⁰⁷ used elsewhere. This allows custom conditions⁶⁵³ to be provided about the state of the <i>bookings</i> you want the <i>Booking Rule</i> to apply to.</p> <p>Note that all the other tabs implicitly provide specifications of what the <i>booking</i> would look like after the change being made, if it were allowed. For example, if somebody changes the <i>project</i> on a <i>booking</i>, then <i>Booking Rules</i> that apply only to certain <i>projects</i> will apply as long as it matches the new <i>project</i>, not the old one.</p> <p>However, <i>conditions</i> allow explicit reference to the state of the <i>booking</i> both before and after the change (Old Value, New Value) as well as the user information making the change (Meta-property user)</p>	

Within each activated tab (**Statuses**, **Bookers**, **Owners**, **Projects**, **Resources**) if a *booking* passes any one of the sections set up in that tab then the *booking* will be accepted for that tab. So if there are items in more than one section of a individual tab (i.e. groups, types etc.) then if any of the sections within a tab is true, then the tab is true. This if a *Booking Rule* is applied to *project type* P and *project group* G, then the *Booking Rule* applies to *projects* of Type P or any *projects* in *project group* G. (The *project* doesn't have to be both of Type P and in Group G for it to pass, just one of them). It may be required that the *Booking Rule* may only want to apply when the *project* is in a particular *group* and also has a particular *project type*. In order to achieve this, specify the *project group* in the **Projects** tab, and then add a *Condition* in the **Conditions** tab to require that the *project type* be what is required.

In order for a *booking* to be finally accepted it must be accepted in all the activated tabs. So if **Bookers** and **Resources** are activated and there is also a **Condition** then the *booking* must be accepted in each of the three tabs in order to be finally accepted. If it is accepted by only one or two of these tabs then it will be rejected.

Does Not Apply To

Finally, as well as allowing the specification of *bookings* the *Booking Rule* does *apply to*, it also allows the same sort of specification for those it *does not apply to*. Sometimes it may be easier to say which *bookings* a *Booking Rule* shouldn't *apply to*, and sometimes it's easier to target the *bookings* required by specifying some things in the **Applies To** section, and some other things in the **Does Not Apply To** section.

Does Not Apply To works in exactly the same way as *Applies To*.

For example, a *Booking Rule* may be needed that is to be applied to most people, but a small number of people should be exempt (perhaps because they are administrators or heads of group). Implement this by creating a *user group* with all the people such a *Booking Rule* should apply to. However, *user groups* tend to work best when a small number of users belong to them - otherwise maintaining the *groups* with the right membership can become onerous. It's therefore easier to maintain a *user group* of the people that should be exempt, and adding them to the **Does Not Apply To** section.

If users are classified by assigning them all a *user type*, then create a *Booking Rule* which targets a particular *user type* in the **Applies To** section, but then removes some users from that selection by the use of a **Does Not Apply To** specification of users.

To see examples of what can be done with this way of choosing which *bookings* a *Booking Rule* should apply to, please see the examples shown for [Simple Booking Rule](#)²⁴³.

One problem to be wary of is setting **Admin** in the **Bookers** of **Does Not Apply To** and having something in the **Condition** tab as well and expecting **Admin** too always be exempt. They will only be exempt if the *Condition* also applies. In this case add a *Condition* of **Meta-property user.roles includes any Admin**.

Bookings To Count/Bookings To Ignore

Some Rules allow the defining of the Bookings that will count or be ignored by the rule (there are slightly different names for the tabs depending on the **Rule** they are used in). These are **Total Time Booked (Bookings To Include In Count)**, **Number Of Bookings**, **Interval (Preceding Bookings, Following Bookings)** and **Double Booking** rules.

There are special options available within the tabs.

Tab	Description	
Bookers	Matched User	Matches the Booker user of the <i>Booking</i> being tested with the Booker user in the <i>Bookings</i> being counted.
	Matched User Type	Matches the Booker User Type of the <i>Booking</i> being tested with the Bookers User Type in the <i>Bookings</i> being counted.
Owner	Matched User	Matches the Owner user of the <i>Booking</i> being tested with Owner user in the <i>Bookings</i> being counted.
	Matched User Type	Matches the Owners User Type of the <i>Booking</i> being tested with the Owners User Type in the <i>Bookings</i> being counted.
Project	Matched Project	Matches the Project associated with the <i>Booking</i> being tested with the Project associated with the <i>Bookings</i> being counted.
	Matched Project Type	Matches the Project Type of the Project associated with the <i>Booking</i> being tested with the Project Type of Projects associated with the <i>Bookings</i> being counted.
Resource	Matched Resource	Matches the Resource of the <i>Booking</i> being tested with the Resource of the <i>Bookings</i> being counted.
	Matched Resource Type	Matches the Resource Type of the Resource associated with the <i>Booking</i> being tested with the Resource Type of the Resource associated with the <i>Bookings</i> being counted
Conditions	Value	The value to be compared is that in the <i>Bookings</i> being counted. Must always be first.
	Booking To Match	The value to be compared is that in the original <i>Booking</i> that is being tested against. Must always be second.
	Meta-property	The value to be compared is with that of the user making the change.

6.6.3.4 Types of Booking Rule

As previously mentioned, all **Calpendo Booking Rules**⁶⁵² specify which *booking*⁶⁵² changes they apply to. Beyond that, the different types of *Booking Rule* vary in their complexity. At one end of the scale is the **Simple Booking Rule**²⁴³, which always rejects or warns about a *booking*, with its real logic being restricted to the selection of the *bookings* it applies to. At the other end of the scale is the **Advanced Booking Rule**²⁶⁹ which requires a programmer to configure, but allows full access to the **Calpendo** database and beyond.

In between the two extremes, there are *Booking Rules* that specialise in particular areas. It is expected that more specialist **Booking Rule Types**⁶⁵³ will be developed over time. In many cases, these new *Booking Rule Types* are likely to start life as an **Advanced Booking Rule**. The following sections describe each of the different types of *Booking Rule*.

6.6.3.4.1 Simple Booking Rule

Beyond the standard settings for selecting which *bookings*⁶⁵² it applies to, a **Simple Booking Rule** only specifies three things:

Property	Description
Rejection Type	This sets up whether to reject the <i>booking</i> or just post a warning.
Retry approved bookings	An indication to be sent to the user of whether to retry sending the <i>booking</i> as a Request (only applicable when the <i>booking</i> received is Approved).
Message to show	The message to be displayed to the user if the <i>booking</i> is rejected or a warning is issued.

Please note that there is no intelligence here to decide whether or not to reject *booking* or give the user a warning - all the logic supported by a **Simple Booking Rule** is in the selection of the *bookings* the **Booking Rule**⁶⁵² applies to. While it may seem that this would make a **Simple Booking Rule** somewhat limited in scope, you can achieve quite a lot with it, as shown by the following examples.

Example: Disallowing Bookings More Than 6 Weeks In Advance

Suppose that the [resources](#)⁶⁵⁵ are very busy, and people are responding by *booking* well into the future, just in case they will have a need. So its decided to stop this behaviour by disallowing *bookings* more than 6 weeks away.

This result can be achieved with [Permissions](#)³¹⁴, although that wouldn't allow a nice error message to the user - they would just receive the standard **Permission Denied** message.

This can also be done with a [Time Template](#)²²⁶ as well as a *Booking Rule*, and it's probably better to do it with a [Time Template](#)⁶⁵⁶ so that the user would get feedback before they try to make such a *booking*. However this example may be helpful.

To do it, create a **Simple Booking Rule** with **Change Type** of **Create and Update** and add this as the [Condition](#)⁶⁵³:

Section	Tab	Description
Applies To	Conditions	New Value of dateRange.start later than now plus 42 days to the minute

This adds 6 weeks to the current time (the time when the user tries to make a *booking*), and then tests whether the *booking* starts (**dateRange.start**) after that time, when both times are rounded to the nearest minute. If the *booking* starts after the 6 week time, then the *condition* is **true**, and so the *Booking Rule* applies.

Please note that if [repeats](#)⁶⁵⁵ are allowed the **Global Preference->Bookings->Number of Days of Bookings** to check for *repeat bookings* must be larger than the number of days you are checking for by 1-2 weeks to stop *repeat bookings* going after that time, depending on what type of *repeat* is being used. A better option would be:

Section	Tab	Description
Applies To	Conditions	New Value of repeat.finish later than now plus 42 days to the day

Example: Physicist Projects Can Book 4 Weeks In Advance, Biologists 6 Weeks

This expands on the previous example, but this time we want to have a different time limit depending on the project used to make the *booking*. Again, this could be done with *Time Templates*, but we'll show how to do it with *Booking Rules*.

Create two **Simple Booking Rules** this time with **Change Type** of **Create and Update**, one to prevent physicists from *booking* too far ahead, and one to prevent biologists from *booking* too far ahead. The physicists' *Booking Rule* would look like this:

Section	Tab	Description
Applies To	Conditions	New Value of dateRange.start later than now plus 28 days to the minute
Applies To	Projects	Add the Physicist project type ⁶⁵⁵ to the list of targeted projects ⁶⁵⁴

and the biologists' *Booking Rule* would look like this:

Section	Tab	Description
Applies To	Conditions	New Value of dateRange.start later than now plus 42 days to the minute
Applies To	Projects	Add the Biologist <i>project type</i> to the list of targeted <i>projects</i>

This assumes that *project types* are used to classify the *projects*. An alternative would be to name the *projects* individually in the *Booking Rule*, or else create [project groups](#)⁶⁵⁵ and use those instead.

Example: Special Projects Can Book Any Time In Advance

Now suppose that there is a *project group* that lists **Special Projects** that should be allowed to book as far into the future as it wants to. A good example of this is if your [resource](#)⁶⁵⁵ needs regular maintenance and so you set up a **Maintenance project**. Most probably associate a small number of users with this *project*, to restrict who can create such *bookings*.

Modify the physicist and biologist *Booking Rules* from the last example so that each of them adds to the **Does Not Apply To** section:

Section	Tab	Description
Does Not Apply To	Projects	Add the Special Projects <i>project group</i> to the list of targeted <i>projects</i>

The effect of this is that whenever a *booking* is made for a *project* in the **Special Projects** *project group*, the *Booking Rules* we created above no longer apply, and so it removes the restriction about how far into the future such *bookings* can be for.

Example: Special Projects And Admins Can Book Any Time In Advance

Now suppose that as well as allowing **Special Projects** to be allowed to book as far into the future as they want to, administrators need to be able to make *bookings* for any *project* well into the future. The users to be excluded from the time limit could be specified by listing them individually, by their [user role](#)⁶⁵⁶ or by creating a [user group](#)⁶⁵⁶ for it. Any of these would be sensible, with the choice depending on how **Calpendo** should be set up to meet your general requirements.

Let's suppose we want to allow users with the **Admin** *user role* to book well into the future for any *project*. Then the physicist and biologist *Booking Rules* could be changed from the previous example so that their **Does Not Apply To** sections contain this:

Section	Tab	Description
Does Not Apply To	Projects	Add the Special Projects <i>project group</i> to the list of targeted <i>projects</i>
Does Not Apply To	Bookers	Add Admin to the bookers <i>user roles</i> selection.

However, this is probably not what is required. With this prescription, any booking whose **Booker** *property*⁶⁵⁵ is set to **Admin** and whose *project* is in the **Special Projects** group will be allowed to book well into the future. For a *Booking Rule* such as this, it may be required to exclude a *booking* from the *Booking Rule* if the *project* is "special" or if the *booking* were made by an administrator. To achieve this, replace the **Bookers** entry with a *condition*, like this:

Section	Tab	Description
Does Not Apply To	Projects	Add the Special Projects <i>project group</i> to the list of targeted <i>projects</i>
Applies To	Conditions	New Value of booker.role does not contain any Admin

Note that we also put the *condition* on the **Applies To** section. Had we used a *condition* in the **Does Not Apply To** section, then we would still have required all of the [Does Not Apply To](#)⁶⁵³ to be applicable for the exclusion to work.

There's also another caveat to this example: if the [Permissions](#)³¹⁴ do not stop non-administrators from making a *booking* with the **booker** *property* set **Admin**, then a non-administrator could work around this *Booking Rule* by setting the **booker** to somebody else when they create the *booking*. This practice can be stopped with [Permissions](#)⁶⁵⁴, but can also be stopped by making the *condition* in the *Booking Rule* checked the **user** that performed the change, rather than the **booker** *property* of the *booking*. This is shown in the next example.

Example: Prevent Cancellations With Less Than 24 Hours To Go

Depending on the policies employed at the facility, last-minute cancellations may need to be prevented apart from those made by an administrator. This will require a *Booking Rule* that targets changes to *bookings* that result in the [booking status](#)⁶⁵³ changing to **Cancelled** from something else, that was made within 24 hours of the *booking* time, and that the change was not made by an administrator.

This can be done with a **Simple Booking Rule** with **Change Type** of **Update** that has the following added to the **Applies To** section:

Section	Tab	Description
Applies To	Statuses	Tick only Cancelled Bookings
Applies To	Resources	Choose the <i>resources</i> the <i>Booking Rule</i> should apply to
Applies To	Conditions	Old Value of dateRange.start earlier than Now Plus 24 hours to the minute
Applies To	Conditions	Old Value of status not equal to Cancelled
Applies To	Conditions	New Value of status equals Cancelled
Applies To	Conditions	Meta-property ⁶⁵⁴ user.roles does not include any Admin

There are some important things to note about these *conditions*:

- By checking the *meta property* called **user, person** that makes the change is being checked, regardless of who is set to be the [booker](#)⁶⁵² or [owner](#)⁶⁵⁴ of the *booking*.
- We could have moved the *user.roles* check to the **Does Not Apply To** section by negating the sense of the *condition*. That is, by adding a *condition* to the **Does Not Apply To** section that said *Meta property user.roles* includes **Admin**. This would have been functionally identical, as long as there were no other entries in the **Does Not Apply To** section.

In the **Rule Tab**, enter a message to be displayed to people who try to cancel within 24 hours of the *booking*.

6.6.3.4.2 Double Booking Rule

Prevention of double [bookings](#)⁶⁵² is clearly something that a *booking* system ought to be able to do. **Calpendo** doesn't automatically prevent double *bookings*, but instead provides a **Double Booking Rule** to allow the system to define precisely how it ought to behave in this area. For example, to create a [Booking Rule](#)⁶⁵² that doesn't allow more than one *booking* for the same [resource](#)⁶⁵⁵ at the same time. That might seem sensible, but it's not always that simple.

Suppose there is a computer room that people can book, and there are 10 computers in there. **Calpendo** could be configured to handle this by creating 10 separate *resources*. However, the alternative is to configure **Calpendo** with a single *resource*, to represent the computer lab itself, and then create a **Double Booking Rule** that allows up to 10 *bookings* for it at the same time.

Another example that shows that double *bookings* are not always as simple as you might always think: suppose there is a MRI scanner with an adjoining room that can be used to treat patients. Users are allowed to book the MRI scanner whenever they like for subjects that are healthy, and therefore not expected to require the use of the adjoining treatment room. However, if there is a need to scan somebody that requires treatment facilities to be available because there's a risk they may require such help, then the user can't book the MRI scanner when somebody already has a *booking* for the treatment room.

This is also a type of double *booking* because it relates to *bookings* that exist at the same time, albeit across different *resources*. This can be implemented in a couple of ways in **Calpendo**. When booking for an MRI scanner for a patient that may require the treatment room **Calpendo** can:

- only allow the MRI scanner to be booked if there isn't already a *booking* for the treatment room, and also do not allow the treatment room to be booked if there's an MRI *booking* for a patient that may require the treatment room.
- only allow the MRI scanner to be booked if the same [project](#)⁶⁵⁴ or [booker](#)⁶⁵² already has a *booking* for the treatment room.

The screenshot shows the 'Rule' tab of a configuration window. It contains several settings for a double booking rule:

- Applies To**: (tab selected)
- Does Not Apply To**: (tab)
- Rule**: (tab selected)
- Rule Validator**: (tab)
- Reject or warn when exceeded**: (dropdown menu)
- Retry approved bookings**: (dropdown menu)
- Limit or require coincident bookings**: (dropdown menu)
- Number of coincident bookings**:
- Show advanced settings**: ☐
- Message to show**:

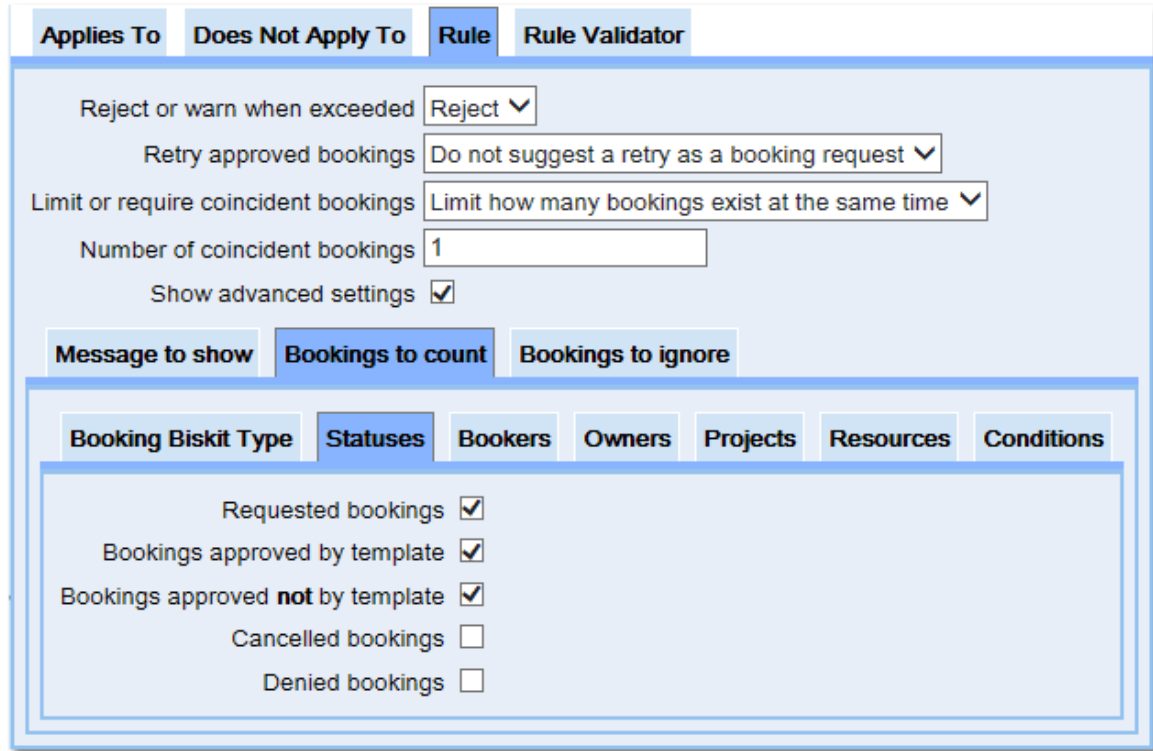
The default settings when creating a **Double Booked Rule** are configured for the simple case - to stop multiple *bookings* for the same *resource* at the same time.

Calpendo allows the handling of more advanced methods as shown in this table:

Property	Description
Rejection Type	Decides whether to reject the <i>booking</i> or just post a warning.
Retry approved bookings	An indication to be sent to the user of whether to retry sending the <i>booking</i> as a Request (only applicable when the <i>booking</i> received is Approved).
Limit or Require coincident bookings	Choose whether the <i>Booking Rule</i> should operate in a mode that limits the number of <i>bookings</i> that can coexist (the standard case), or whether the <i>Booking Rule</i> should require at least a given number of <i>bookings</i> to exist beforehand (for example to ensure that the treatment room is booked before being allowed to book the MRI scanner)
Number of coincident bookings	The standard Double Booking Rule sets the number of <i>bookings</i> allowed to one. This means that there can only be one <i>booking</i> at a time. In the case of the computer lab with ten computers, this could be set to ten.
Message to show	The message to be displayed to the user if the <i>booking</i> is rejected or a warning is issued.
Bookings to count	This tab is only shown when the advanced settings option is selected. This is described in detail below.
Bookings to ignore	This tab is only shown when the advanced settings option is selected. This is described in detail below.

Advanced Settings

Turning on the **advanced settings** option in the **Double Booking Rule** displays two extra tabs: **Bookings To Count** and **Bookings To Ignore**. These tabs are not required in the simple case, where we only want to prevent multiple *bookings* for the same *resource* to coexist. They are required when *bookings* for one *resource* need to influence *bookings* for another *resource*.



The **Bookings to Count** and **Bookings to Ignore** work in exactly the same way as [Applies To](#)⁶⁵² and [Does Not Apply To](#)⁶⁵³, read the chapter on [Choosing Which Bookings A Rule Applies To](#)²³⁸ for more information on how the sub tabs work.

In the MRI scanner and treatment room example, there are two ways to implement it: preventing a *booking* for a patient that may require the treatment room when the treatment room is already booked or requiring that the treatment room already be booked by the same *project*. In each case, we need to identify which MRI *bookings* might require the treatment room. For that, use the [Types And Groups Editor](#)²⁸⁰ to create a *booking type*⁶⁵³ to represent the type of subject being scanned. Classify subjects according to whether they are healthy volunteers, walking patients, wheelchair patients, or bedridden patients. Now suppose that healthy volunteers do not require the treatment room to be available, but the other types do.

Here's the first option for how to set this up in **Calpendo**, by preventing the treatment room from being booked at the same time as the MRI scanner:

Property	Setting
Applies To	Add condition ⁶⁵³ : <i>type</i> not equal to Healthy Volunteer
Applies To	On the Resources tab, specify the MRI Scanner
Reject or warn when exceeded	Reject
Retry approved bookings	Do not suggest retry as a <i>booking</i> request
Exclusivity	Limit how many <i>bookings</i> can exist at the same time
How many bookings allowed	One
Show advanced settings	Yes
Bookings To Count	On the Resources tab, change Matched Resource to Treatment Room
Bookings To Ignore	Not required

This means that any *booking* for the MRI scanner, where the *booking type* is not set to **Healthy Volunteer** will trigger the *Booking Rule*. The *Booking Rule* will then count how many *bookings* there are for the treatment room at the same time as the prospective MRI scanner *booking*. The limit for the number of allowed *bookings* is one, and this always includes the prospective *booking* (in this case the one for the MRI scanner). The net result is that MRI scanner *bookings* will be rejected when being made for non-healthy volunteers if there's a *booking* for the treatment room at the same time.

If it is implemented in this way, then a similar *Booking Rule* would be needed that prevented *bookings* for the treatment room when there are coincident *bookings* for the MRI scanner for non-healthy subjects.

The second option for how to set this up in **Calpendo** is to require a pre-existing *booking* for the same *project* in the treatment room. Here's what this looks like:

Property	Setting
Applies To	Add <i>condition: type</i> not equal to Healthy Volunteer
Applies To	On the Resources tab, specify the MRI Scanner
Reject or warn when exceeded	Reject
Retry approved bookings	Do not suggest retry as a <i>booking</i> request
Exclusivity	Require a number of pre-existing <i>bookings</i>
How many bookings allowed	One
Show advanced settings	Yes
Bookings To Count	<ul style="list-style-type: none"> On the Resources tab, change <i>Matched Resource</i> to Treatment Room On the Projects tab, tick Specify projects and change Projects from Any Project to Matched Project
Bookings To Ignore	Not required

The **Matched Project** setting means that, when **Calpendo** looks for pre-existing *bookings*, it will only include *bookings* that have the same *project* as the prospective one being created (in this case the *project* on the MRI scanner *booking*). This means that, if there's a *booking* for the same *project* in the treatment room, we're going to assume that the treatment room has been pre-booked for the purpose of providing back-up in case the patient requires it.

Bookings To Ignore

The previous examples showed how to use **Bookings To Count** to specify which *bookings* **Calpendo** should count when looking for *bookings* at the same time as the one being created or modified. The **Bookings To Ignore** tab provides an extra option to the way the counting is done, by allowing the specification of *bookings* that are not going to be counted. This is analogous to the way one can specify *Applies To* and *Does Not Apply To* settings.

Handling Holidays

Some *resources* require the presence of people while they are being used. For example, an MRI scanner requires radiographers. Sometimes, suitably qualified people book the scanner for their own use and don't require an additional radiographer, but on other occasions, *bookings* require additional support. When radiographers are on holiday, the *booking* system should be able to provide suitable information to anybody making a *booking* throughout the holiday period. There are several ways to do this with **Calpendo**, each with their own benefits and drawbacks.

Create a [Time Template](#)⁶⁵⁶ that issues warnings to everyone, and then add entries to the [Time Templates Calendar](#)⁶⁵⁶ whenever there's a holiday. Or, create a [Holiday Booking Rule](#)²⁵⁷. Finally, do it with a **Double Booking Rule** as follows:

1. Create a new *resource* where holiday information is stored. Let's call it **Radiographer Leave** for this example.
2. Create a **Double Booking Rule** that applies to MRI scanner *bookings*, and has its **Bookings To Count** set to count up the number of *bookings* for **Radiographer Leave**.
3. Set the exclusivity to limit how many bookings can exist at the same time and the **How many bookings allowed** to one.
4. Set the **Reject or warn when exceeded** to **warn**
5. Enter a message that there is limited radiographer cover (to show to the user).

Now, when a radiographer plans a holiday, they create a *booking* for the **Radiographer Leave** *resource*. Whenever somebody then makes a *booking* for the MRI scanner at the same time, the **Double Booking Rule** will give a warning to the user. The nice thing about implementing holiday handling with this method is that it's very easy to delegate the responsibility for giving the warnings to the radiographers, and there's an entry in the [calendar](#)⁶⁵² so everyone can see when there is no radiographer cover available.

This table compares the methods for displaying warnings to users during a holiday period:

Method	Pros	Cons
<i>Time Template</i>	Provides visual feedback for the duration of the holiday in the background of the <i>Bookings Calendar</i> .	<ul style="list-style-type: none"> No way to see who is on holiday or how many people are on holiday. Either need radiographers to modify Time Templates⁶⁵⁶, or an Admin has to set up holidays on behalf of radiographers.
Holiday Booking Rule	Conceptually easy.	<ul style="list-style-type: none"> No visual feedback before making the booking. New <i>Booking Rule</i> needed for each holiday, or else modify the old <i>Booking Rule</i> to add more holidays. Either need radiographers to modify <i>Booking Rules</i>, or an Admin has to set up holidays on behalf of radiographers.
Double Booking Rule	<ul style="list-style-type: none"> Radiographers are given their own "leave calendar". They can see and edit their own leave information. Other users can see radiographer leave information. 	Conceptually a little more difficult to set up than the other approaches, although once configured, it's easy for radiographers to add holidays and for users to comprehend.

6.6.3.4.3 Booking Duration Rule

The **Booking Duration Rule** allows limits to be set on how long [bookings](#)⁶⁵² can be for certain *resources*. Set a minimum or maximum time for *bookings* or specify exactly how long they may be.

Property	Description
Reject or warn when exceeded	Decide whether to reject the <i>booking</i> or just post a warning.
Retry approved bookings	An indication to be sent to the user of whether to retry sending the <i>booking</i> as a Request (only applicable when the <i>booking</i> received is Approved).
Required Duration	Set the duration to be a minimum, maximum or exactly a specific time period. The time period can be a Fixed Value that is specified or a Variable Value , something that is read from a property ⁶⁵⁵ accessed through the <i>booking</i> .
Message to show	The message to be displayed to the user if the <i>booking</i> is rejected or a warning is issued.

6.6.3.4.4 Holiday Booking Rule

A **Holiday Booking Rule** is very similar to a **Simple Booking Rule**, but also allows a list of dates to be specified. The [Booking Rule](#)⁶⁵² will reject or warn for any [booking](#)⁶⁵² that is on one of the dates provided. This is intended to be used so that there is a *Booking Rule* that's easy to set up that will either reject or produce a warning on particular days, for holidays, equipment servicing or anything else that might be needed.

Note that there is similar functionality using either a **Double Booking Rule** or [Time Templates](#)⁶⁵⁶. Which implementation is preferred will depend on the circumstances. See [Handling Holidays](#)²⁵⁵ for a comparison of the available methods.

Property	Description
Rejection Type	Decide whether to reject the <i>booking</i> or just post a warning.
Retry approved bookings	An indication to be sent to the user of whether to retry sending the <i>booking</i> as a Request (only applicable when the <i>booking</i> received is Approved).
Message to show	The message to be displayed to the user if the <i>booking</i> is rejected or a warning is issued.
Select holiday date	Allows selection of the holiday dates during which this <i>Booking Rule</i> will either reject <i>bookings</i> or give a warning.

6.6.3.4.5 Booking Interval Rule

The **Booking Interval Rule** lets the administrator specify that the time between some (or all) *bookings* must be at least a particular time. This can be used to enforce gaps between all *bookings*, or to prevent *projects*⁶⁵⁴ that overheat a *resource*⁶⁵⁵ from following each other too closely. The rule can also ensure that **Users** do not leave small gaps in the schedule.

Property	Description
Reject or warn when exceeded	Decide whether to reject the <i>booking</i> or just post a warning.
Retry approved bookings	An indication to be sent to the user of whether to retry sending the <i>booking</i> as a Request (only applicable when the <i>booking</i> received is Approved).
Interval Type	Choose whether to 1) Require Large Separation 2) Eliminate Small Gaps
Required Interval	Time between <i>bookings</i> for separation, or the minimum gap if eliminating small gaps.
Message to show	The message to be displayed to the user if the <i>booking</i> is rejected or a warning is issued.
Preceding Bookings	The Preceding Bookings and Following Bookings work in exactly the same way as Applies To ⁶⁵² , read the chapter on Choosing Which Bookings A Rule Applies To ²³⁸ for more information on how the sub tabs work. The only difference is, it is setting up <i>bookings</i> to proceed the <i>booking</i> and <i>bookings</i> to follow the <i>booking</i> rather than those it applies to and those it doesn't.
Following Bookings	

The algorithm used for the prevention of small gaps is that any booking that results in a gap smaller than the one declared will be rejected unless the gap before or after it is zero. That is, if the minimum gap is 45 minutes, then it is considered legal to put a 60 minute booking into a 90 minute slot so long as it starts when the previous booking ends, or it ends when the following booking starts.

6.6.3.4.6 Number Of Bookings Rule

The **Number of Bookings Rule** allows a limit on the total number of [bookings](#)⁶⁵² that can be made. For example, the number of approved/requested *bookings* can be limited for each [project](#)⁶⁵⁴. This can be a fixed limit, or the limit can be selected from the *project's* settings.

The number of *bookings* can be specified in the [Booking Rule](#)⁶⁵² or a value can be read from a [property](#)⁶⁵⁵, so research *projects* can be set up to have a limited number of *bookings* defined in their *project* definition, but it can be specified that *bookings* on the weekend do not count towards that total using **Bookings to Ignore**.

Property	Description
Reject or warn when exceeded	Decide whether to reject the <i>booking</i> or just post a warning.
Retry approved bookings	An indication to be sent to the user of whether to retry sending the <i>booking</i> as a Request (only applicable when the <i>booking</i> received is Approved).
Number Bookings	Set the number of <i>bookings</i> to be a specified Fixed Value or a Variable Value , something that is read from a <i>property</i> accessed through the booking. A negative number means unlimited.
Message to show	The message to be displayed to the user if the <i>booking</i> is rejected or a warning is issued.
Bookings to count	<p>The Bookings to Count and Bookings to Ignore work in exactly the same way as Applies To⁶⁵² and Does Not Apply To⁶⁵³, read the chapter on Choosing Which Bookings A Rule Applies To²³⁸ for more information on how the sub tabs work.</p> <p>The main change is that there is a Matched option for users, <i>projects</i>, resources⁶⁵⁵, user types⁶⁵⁶, project types⁶⁵⁵ and resource types⁶⁵⁵. This specifies that the <i>bookings</i> to be counted/ignored must be compared with the current <i>booking</i> and match this <i>booking</i> in some way, either by user, <i>project</i> or <i>resource</i> (or type of each). For example setting Bookers equals Matched User and Resource to Matched Resource means <i>bookings</i> for this user on the current <i>resource</i> only, will be looked at when counting <i>bookings</i> to count/ignore, which allows setting up very generic <i>Booking Rules</i> without needing a similar <i>Booking Rule</i> for every user in the system.</p>
Bookings to ignore	

Example: Allowing a user to only have a certain number of future bookings at any one time

The facility may want to limit users to a specific number of *bookings* at any one time (or in a particular time period). Once a *booking* is used the user can then make another *booking*. As an example a user may only have three future *bookings* on a *resource* at any one time.

To do this create a **Number of Bookings Rule**, give it a name and set the **Number of Bookings** to be 3. (**Rule** tab). Then set up some *Booking Rules*, the first part will count *bookings* matching the current user, the second part will count only future *bookings*. Use **New value** rather than **Old value** so that already created *bookings* that are moved from somewhere where they are not counted, to somewhere where they will be counted will be caught.

Section	Tab	Description
Bookings to count	Bookers	Users set to Matched User
Bookings to count	Conditions	Value of <code>dateRange.start</code> later than now to the second

The following is set up by default, defining that the *bookings* must be for the same *resource*.

Section	Tab	Description
Bookings to count	Resources	Resources set to Matched Resource

This could be expanded to *resource types* ensuring a user could book no more than three times over a number of *resources* of the same type or removed altogether to restrict a user to a maximum of three *bookings* over the whole facility.

To limit the period to the next three weeks set up an additional [condition](#)⁶⁵³:

Section	Tab	Description
Bookings to count	Conditions	Value of <code>dateRange.start</code> no later than now plus 21 days to the second

This *Booking Rule* could be on a per *project* basis rather than a per *booker*⁶⁵² basis, just use **Projects->Matched Project** instead of **Bookers->Matched User**.

To ignore *bookings* finishing before 9AM, starting after 6PM or on a Saturday or Sunday:

Section	Tab	Description
Bookings to ignore	Conditions	Value of <code>dateRange.finish.minuteOfDay</code> less than or equal to 540
Bookings to ignore	Conditions	Value of <code>dateRange.start.minuteOfDay</code> greater than or equal to 1080
Bookings to ignore	Conditions	Value of <code>dateRange.start.dayOfWeek</code> equals Specified Value 1 (Sunday)
Bookings to ignore	Conditions	Value of <code>dateRange.start.dayOfWeek</code> equals Specified Value 7 (Saturday)

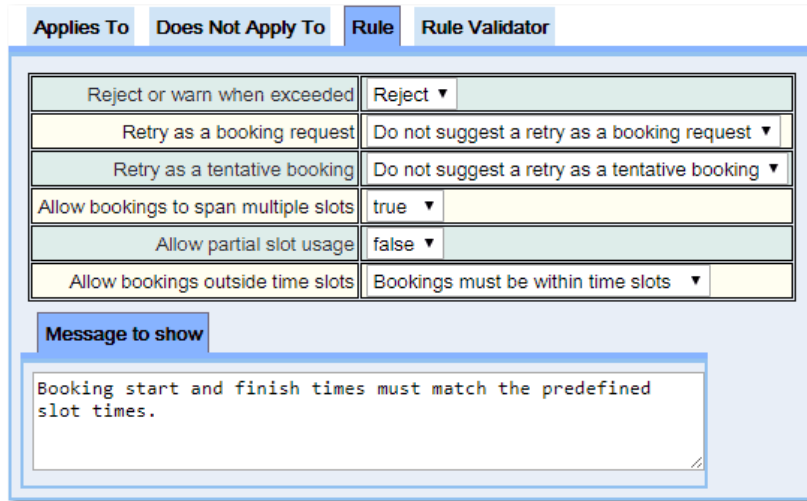
Please note the change to **Any of the following apply**.

Do something similar in **Does Not Apply To**. This will make sure that when creating the *booking* it is ignored in the count.

Section	Tab	Description
Does Not Apply To	Conditions	New value of <code>dateRange.finish.minuteOfDay</code> less than or equal to 540
Does Not Apply To	Conditions	New value of <code>dateRange.start.minuteOfDay</code> greater than or equal to 1080
Does Not Apply To	Conditions	New value of <code>dateRange.start.dayOfWeek</code> equals Specified Value 1 (Sunday)
Does Not Apply To	Conditions	New value of <code>dateRange.start.dayOfWeek</code> equals Specified Value 7 (Saturday)

6.6.3.4.7 Predefined Slots Booking Rule

The **Predefined Slots Booking Rule** allows the administrator to specify the times of the day that *bookings*⁶⁵² must adhere to. This can be used to enforce time slots starting and finishing at particular times. The *Booking Rule*⁶⁵² will also enforce whether *bookings* may go over multiple time slots or partial time slots may be booked. The time slots for this *Booking Rule* are set up in the *Resource Editor*²⁸⁴.



Applies To	Does Not Apply To	Rule	Rule Validator
Reject or warn when exceeded		Reject ▼	
Retry as a booking request		Do not suggest a retry as a booking request ▼	
Retry as a tentative booking		Do not suggest a retry as a tentative booking ▼	
Allow bookings to span multiple slots		true ▼	
Allow partial slot usage		false ▼	
Allow bookings outside time slots		Bookings must be within time slots ▼	

Message to show

Booking start and finish times must match the predefined slot times.

Property	Description
Reject or warn when exceeded	Decide whether to reject the <i>booking</i> or just post a warning.
Retry approved bookings	An indication to be sent to the user of whether to retry sending the <i>booking</i> as a Request (only applicable when the <i>booking</i> received is Approved).
Allow bookings to span multiple slots	Defines whether a <i>booking</i> can go over multiple defined slots
Allow partial slot usage	Defines whether a user can make a <i>booking</i> in the rest of a slot after someone not confined by the slots (such as Admin) has made a <i>booking</i> in only part of a slot.
Allow bookings outside time slots.	Defines whether bookings are allowed outside any defined time slots.
Message to show	The message to be displayed to the user if the <i>booking</i> is rejected or a warning is issued.

6.6.3.4.8 Search Results Booking Rule

The **Search Results Booking Rule** allows the administrator to check for particular information on a different *Biskit Type*⁶⁵² before allowing the *booking*⁶⁵² to go ahead. One possible use of this *Booking Rule*⁶⁵² is to check whether a *booker*⁶⁵² has been trained on a particular *resource*⁶⁵⁵. If the administrator has created a *Biskit Type* to hold training information, such as date, *resource*, trainee name, then the *Booking Rule* can check whether a potential *booker* is trained.

The *Booking Rule* searches for a *Biskit Type*, the administrator can then specify how many *properties*⁶⁵⁵ in the *booking* and the *Biskit Type* must match, and the names of those *properties* and how many record matches are required. As with other *Booking Rules* additional *conditions*⁶⁵³ can be set up. I.e. training, the *Booking Rule* could match the trainee name with the *bookers* name, the *resource* name with the training *resource* name, and have a condition to check whether the training was done within the last year.

Name	Double Bookings
Enabled	Enabled ▼
Change Type	Create and update ▼
Rule Type	Search Results ▼
Order	1

Applies To
Does Not Apply To
Rule
Rule Validator

Reject or warn when exceeded	Reject ▼
Retry approved bookings	Do not suggest a retry as a booking request ▼
Search For	Booking
Search must find	At least ▼ 1 items
Number of properties to match	2

Message to show
Conditions
Match 1
Match 2

Search property	Select a property
Relation	equals ▼
Booking property	Select a property

Property	Description
Reject or warn when exceeded	Decide whether to reject the <i>booking</i> or just post a warning.
Retry approved bookings	An indication to be sent to the user of whether to retry sending the booking as a Request (only applicable when the <i>booking</i> received is Approved).
Search For	The <i>Biskit Type</i> to compare against
Search must find	How many records must be found to match
Number of properties to match	How many <i>properties</i> will need to have their values matched between the <i>booking</i> and the searched for <i>Biskit Type</i> . For each one required a Match option will appear below.
Message to show	The message to be displayed to the user if the <i>booking</i> is rejected or a warning is issued.
Match ?	The names of the <i>properties</i> of the <i>booking</i> and the Search Biskit Type to be compared and the type of comparison.

6.6.3.4.9 Total Time Booked Rule

A **Total Time Booked Rule** checks how much time is being booked over a given period of time. For example, the amount of time booked per [project](#)⁶⁵⁴ or user within any 7 day period can be limited. A [Booking Rule](#)⁶⁵² like this may need to be implemented if the [resources](#)⁶⁵⁵ are extremely busy, and it is required to restrict the use an individual has of the [resources](#) so that everybody has a chance to get their fair share of them.

It could also be decided that it is necessary to limit how much time is used during some core operating hours, while allowing out-of-hours usage not to count towards the usage limit. This can then encourage people to use your [resources](#) during less busy periods. This can be done either day-by-day, or during whatever time periods you want. For example, there may be a **Total Time Booked Rule** that limits the time booked from 8am to 6pm Monday to Friday, while having another *Booking Rule* that puts a higher limit on usage from 6am to 8pm Monday to Friday, thereby encouraging use earlier and later in the day.



Note

If you limit usage to 4 hours in any 7 day period, and somebody has a *booking* for 4 hours on a Tuesday afternoon, and another *booking* for 4 hours on the following Tuesday morning, this will count as 8 hours use in 7 days, contravening the 4 hour limit.

For this to work, you need to tell **Calpendo**:

Applies To

Does Not Apply To

Rule

Rule Validator

Reject or warn when exceeded

Reject

Retry approved bookings

Do not suggest a retry as a booking request

Maximum booking time allowed

Fixed value

4

Hours

Applies

Every

7

Days

Show advanced settings ☒

Message to show

Bookings to include in count

Booking time to include

☒ Specify times to include in count

◀◀

◀

Week 20

▶

▶▶

Today

Refresh

Go to...

Day

Week

	13/05	14/05	15/05	16/05 Thu	17/05 Fri	18/05 Sat	19/05
00:00							
01:00							
02:00							
03:00							
04:00							

Name	Default Value	Description
Reject or warn when exceeded	Reject	Whether to reject or warn for <i>booking</i> too much time.
Retry approved bookings	Do not suggest a retry as a booking request	Whether approved bookings ⁶⁵² should be tried again as a request.
Maximum booking time allowed	4 hours	The total amount of time that is allowed to be booked. This can be a fixed value or read from a property ⁶⁵⁵ . A negative number means unlimited.
Applies	Every 7 days	The time period over which users can book the Maximum booking time allowed . This can be over a time period (Every) or in total (In Total)
Message to show	There can be a total of [hoursAllowed] hours and [minutesAllowed] minutes, but this booking would make a total of [hoursUsed] hours and [minutesUsed] minutes.	This is the message to be shown to the user, and this can include dynamically-generated text that indicates the amount of time allowed to be booked and the time used, if the requested <i>booking</i> were allowed.
Bookings To Include In Count	Requested, Approved and Approved by Template, Matched Booker and Matched Resource	Specifies which <i>bookings</i> should be counted towards the amount of the Maximum booking time allowed .
Bookings Time To Include	All time	Indicates which periods of time should count towards the amount of the Maximum booking time allowed . For example, should we only include time booked from 8am to 6pm?

Bookings To Include In Count

The **Total Time Booked Rule** is triggered when somebody tries to create or update a *booking*. When looking for other *bookings* to include in the count towards the amount of time used, specify *bookings* using the same tabs that appear in the **Applies To** and **Does Not Apply To** sections of every *Booking Rule*. However, by default, a simpler interface is shown. This chooses whether to count *bookings*:

- with the same [booker](#)⁶⁵²
- with the same [owner](#)⁶⁵⁴
- with the same [project](#)⁶⁵⁴
- with the same [resource](#)⁶⁵⁵
- of each possible [booking status](#)⁶⁵³

Tick the **Show Advanced Settings** check box if it is required to use the extended options that include the extra tabs. The advanced **Bookings to include in count** works in exactly the same way as [Applies To](#)⁶⁵², read the chapter on [Choosing Which Bookings A Rule Applies To](#)²³⁸ for more information on how the sub tabs work.

Booking Time To Include

The **Booking Time To Include** tab does not show unless the **Show Advanced Settings** checkbox is ticked. Without this, all time booked within the **Applies To** are included in the count. To choose the periods of time that should count towards the total, tick **Show Advanced Settings** and go to the **Bookings Time To Include** tab.

If a new *Booking Rule* is being created, then the *Booking Rule* must be saved first, and then go back into edit mode in order to specify the times that should be included in the count. A calendar display will appear allows the creation of entries very similar to the way that *bookings* are created on the [Bookings Calendar](#)⁶⁵². This is the way to specify the times that should be included in the count - if there's an entry in the **Bookings Time To Include** calendar, then the time will count.

As an example to make a **Total Time Booked Rule** count periods of time that are from 8am to 6pm, from Monday to Friday, create an entry for Mon-Fri that *repeats*⁶⁵⁵ weekly. For more information on setting up *repeat* time masks read the section on Repeat Bookings. Once the times are selected click the **Create time Mask** button. There can be multiple time masks on the calendar.

Example: Allowing a user to only have a certain number of hours of future bookings at any one time

The facility may want to limit users to having a specific number of hours of *bookings* at any one time (or in a particular time period). Once a *booking* is used the user can then make another *booking*. As an example a user may only have four hours of future *bookings* on a *resource* at any one time.

To do this create a **Total Time Booked Rule**, give it a name. Set the **Maximum booking time allowed** to four and **Applies to** **in Total**. (Both under the **Rule** tab)

Section	Tab	Description
Bookings to include in count	Conditions	Value of <code>dateRange.start</code> later than now to the second

The following is set up by default in this *Booking Rule*, which defines that the *bookings* must be for the same *booker* and the same *resource*.

Section	Tab	Description
Bookings to include in count	Bookers	Users set to Matched User
Bookings to include in count	Resources	Resources set to Matched Resource

of course this could be expanded to either [resource groups](#)⁶⁵⁵ or [resource types](#)⁶⁵⁵ ensuring a user could not book more than four hours over a number of *resources* or removed altogether to restrict a user to a maximum of four hours over the whole facility.

Read the example for the [Number of Bookings Rule](#)²⁶⁰ for limiting the *Booking Rule* to the next three weeks or making it *project* based rather than user based.

Use the calendar in [Booking Time To Include](#)²⁶⁷ to set those hours that will be counted towards the total.

6.6.3.4.10 Advanced Booking Rule

The **Advanced Booking Rule** is no longer used, use [Workflows](#)⁴⁸⁵ instead.

Advanced Booking Rules provide a means for the implementation of almost anything required. They allow the configurer to write a *Booking Rule* using [Java](#) that will be run using [BeanShell](#). This means it requires a programmer to create an **Advanced Booking Rule**. This documentation includes some example [Booking Rules](#)⁶⁵², so that a programmer may be able to produce the effect they want from what's here, but this is not a complete reference to the functionality available inside an **Advanced Booking Rule**. A complete programmer's API documentation may be made available at some time, but for now, to create an **Advanced Booking Rule**, either try to adapt the examples here, or [contact us](#) for help.

When creating an Advanced Rule a user needs to define how Advanced Rules will work with [repeat bookings](#)⁶⁵⁵. Normally rules are run once for each instance of a *repeat booking*.

Option	Description
Run for each repeating instance	The Rule ⁶⁵² is given a <i>copy</i> of the <i>repeat booking</i> . Consequently, if the rule modifies this copy, changes made to it will not be saved to the database nor seen by subsequently run <i>rules</i> .
Run once	The <i>rule</i> is given the original <i>repeat booking</i> , and changes the <i>Rule</i> makes to the <i>booking</i> will then be persisted and also available to subsequent <i>Rules</i> .

Note: that for *repeat bookings*, advanced *Rules* marked as being run once will always be run before all other *Rules* regardless of requested *Rule* execution order.

When your *Booking Rule* is run, the following local variables will be set in the *BeanShell* environment:

Name	Class	Description
biskitDAO	com.springsolutions.biskit.persistence.HibernateBiskitDAO	Interface to the Calpendo database.
biskitDefs	com.springsolutions.biskit.core.def.BiskitDefStore	Provides access to all the Biskit ⁶⁵² definitions.
booking	com.springsolutions.calpendo.domain.Booking	The booking ⁶⁵² being created or the new version of the <i>booking</i> being updated. Identical to newBooking .
newBooking	com.springsolutions.calpendo.domain.Booking	The <i>booking</i> being created or the new version of the <i>booking</i> being updated. Identical to booking .
oldBooking	com.springsolutions.calpendo.domain.Booking	The original version of the <i>booking</i> for an update, and null otherwise.
realUser	com.springsolutions.calpendo.domain.CalpendoUser	The user making the change.
user	com.springsolutions.calpendo.domain.CalpendoUser	The effective user making the change. This can be different from the real user when running with The Booking Rule Validator ²⁷⁷ .
isUpdate	Boolean	True if the user is making an update, and false if a <i>booking</i> is being created.
result	com.springsolutions.calpendo.domain.rule.RulesResult	The result that must be filled in by the <i>Booking Rule</i> .
rule	com.springsolutions.calpendo.domain.rule.AdvancedRule	The <i>Booking Rule</i> itself. Sometimes you may want to insert the <i>Booking Rule</i> name into the error message shown to a user.

API

The **RulesResult** class, as represented by local variable **result**, is the way to pass information back to **Calpendo**. It has the following API:

```
public interface RulesResult
{
    /** Sets the message that should be sent to the user. */
    public void setMessage(String msg);

    /** Sets an output string to be displayed when calling via
     * The Rule Validator. */
    public void setOutput(String output);

    /** Specify whether to reject, warn or accept the booking. */
    public void setRejectionLevel(RejectionLevel level);

    /** Specify whether to retry the booking as a request.
     * This is ignored if the booking status was not Approved. */
    public void setTryBookingRequest(boolean tryRequest);
}
```

RejectionLevel is an enum as follows:

```
package com.springsolutions.calpendo.domain.rule;

public enum RejectionLevel
{
    ACCEPT("Accept booking"),
    WARN("Warn"),
    REJECT("Reject");

    private String m_label;

    private RejectionLevel(String label)
    {
        m_label = label;
    }

    @Override
    public String toString()
    {
        return m_label;
    }
}
```

Example: Reject Bookings More Than 6 Weeks Into The Future

Here is an example of an **Advanced Booking Rule** that will reject *bookings* made more than 6 weeks into the future. Note that this can be done both with a **Simple Booking Rule** and also (better still) with [Time Templates](#)⁶⁵⁶. However, this serves as an example of how to use advanced *Booking Rules*.

```
import com.springsolutions.calpendo.domain.*;
import com.springsolutions.calpendo.domain.rule.*;
import java.util.Calendar;

// Throw away daylight savings effect.
// This is technically incorrect, but is what people expect when
// crossing DST boundaries (ie use the time shown on the wall
// clock, not the number of minutes between now and then)

then_cal = Calendar.getInstance();
then_cal.setTime(booking.getDateRange().getStart());
then_cal.set(Calendar.DST_OFFSET, 0);
then_time = then_cal.getTime().getTime();

now_cal = Calendar.getInstance();
now_cal.set(Calendar.DST_OFFSET, 0);
now_time = now_cal.getTime().getTime();

days = (then_time - now_time)/(1000.0*60*60*24);

if (days > 42)
{
    int idays = (int) days;
    int ihours = (int) ((days-idays)*24);
    int imins = (int) ((days - idays)*24*60 - ihours*60);
    result.setRejectionLevel(RejectionLevel.REJECT);
    result.setMessage(
        "You cannot book more than 6 weeks in advance "
        + "(your request was " + idays + " days, "
        + ihours + " hours and " + imins
        + " minutes in advance)."
        + " Please discuss any special requirements with a "
        + "moderator. A list of moderators can be found under "
        + "the search menu." );
}
```

Example: Enforcing Project Resource Settings

Calpendo uses [Hibernate](#) for accessing its database. The database can be accessed directly by obtaining a Hibernate SessionFactory as shown by this example. If this is done, make sure that the Hibernate session is closed. This *Booking Rule* does the following:

- Only run for *bookings* that have a [project](#)⁶⁵⁴ associated
- Reject the *booking* if trying to book for a [resource](#)⁶⁵⁵ that's not represented by the [Project's Resource Settings](#)⁶⁵⁴
- Reject the *booking* if it is for a time after the *project's finish property*⁶⁵⁵, where **finish** is a dynamic *property*.
- Reject the *booking* if its duration is more than the **minutesPerSession** *property* of the *Project's Resource Settings*
- Use Hibernate to count the total number of **Approved** and **Requested** *bookings* for the *project*, and reject the *booking* if there would be more *bookings* than the **numberOfSessions** *property* of the *Project's Resource Settings*

These tasks are more easily done using the dedicated [Booking Rule Types](#)⁶⁵³, but this *Booking Rule* serves as a good example of how to interact with the **Calpendo** database directly through Hibernate.

```
import com.springsolutions.calpendo.domain.*;
import com.springsolutions.calpendo.domain.rule.*;
import com.springsolutions.exprodo.core.server.HibernateUtil;
import java.util.Calendar;
import java.util.Date;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

resource = booking.getResource();
rpk = resource.getPrimaryKey();
project = booking.getProject();
if (project == null) return;

prsSet = project.getResourceSettings();
prs = null;
for (ProjectResourceSettings tmp : prsSet)
{
    if (tmp.getResource().getPrimaryKey() == rpk)
    {
        prs = tmp;
        break;
    }
}

if (prs == null)
{
    result.setRejectionLevel(RejectionLevel.REJECT);
    result.setMessage(
        "You are trying to book for a project ("
        + project.getProjectCode()
        + ", " + project.getName()
        + ") that has not been approved for using "
```

```

        + resource.getName());
    return;
}

endDateProp = project.get("finish");
endDate = (endDateProp == null) ? null : endDateProp.getValue();
if (endDate != null
    && booking.getDateRange().getFinish().after(endDate))
{
    result.setRejectionLevel(RejectionLevel.REJECT);
    result.setMessage(
        "You are trying to book for a project ("
        + project.getProjectCode()
        + ", " + project.getName()
        + ") after its project end date (" + endDate + ")");
    return;
}

duration = booking.getDurationInMinutes();
maxDuration = prs.getMinutesPerSession();
if (duration > maxDuration)
{
    result.setRejectionLevel(RejectionLevel.REJECT);
    result.setMessage(
        "Bookings for project (" + project.getProjectCode()
        + ", " + project.getName()
        + ") should be no longer than " + maxDuration
        + " minutes, but you've tried to make a booking for "
        + duration + " minutes");
    return;
}

hql = "select count(*) from " + Booking.TYPE + " where project.id = "
    + project.getPrimaryKey()
    + " and (status = 'REQUESTED' or status = 'APPROVED') "
    + " and resource.id = " + rpik;

if (booking.getPrimaryKey() != 0)
    hql = hql + " and id != " + booking.getPrimaryKey();

SessionFactory sf = HibernateUtil.getSessionFactory();
Session session = null;
try
{
    session = sf.openSession();
    count = session.createQuery(hql).uniqueResult();
    print("Query found a total of " + count
        + " bookings for this project and resource");
    if (booking.getPrimaryKey() == 0)
        count++;

    maxCount = prs.getNumberOfSessions();

    if (count > maxCount)
    {
        result.setRejectionLevel(RejectionLevel.REJECT);
        result.setMessage(

```

```

        "There can be a total of " + maxCount
        + " bookings for project (" + project.getProjectCode()
        + ", " + project.getName()
        + " on " + resource.getName()
        + "), but this booking would make a total of " + count);
    return;
}
}
finally
{
    if (session != null)
        session.close();
}

```

Example: Enforce 15 Minute Gaps Between Bookings

This *Booking Rule* will ensure that there is always a 15 minute gap between *bookings*. If using this *Booking Rule*, it may be a good idea to use the *Rule's Applies To --> Resources* tab to make sure that it only [applies to](#)⁶⁵² the appropriate *resources*. Before using this please look at the standard Booking Interval Rule as this may be much easier to implement what you want.

```

import com.springsolutions.biskit.core.Biskit;
import com.springsolutions.calpendo.domain.Booking;
import com.springsolutions.calpendo.domain.rule.RejectionLevel;
import com.springsolutions.exprodo.core.server.HibernateUtil;
import com.springsolutions.timeRepeating.domain.RepeatUtil;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import org.hibernate.Session;
import org.hibernate.Query;

session = HibernateUtil.getSessionFactory().openSession();
sqlFormatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

try
{
    Calendar cal = Calendar.getInstance();
    cal.setTime(booking.getDateRange().getStart());
    cal.add(Calendar.MINUTE, -15);
    Date start = cal.getTime();
    cal.setTime(booking.getDateRange().getFinish());
    cal.add(Calendar.MINUTE, +15);
    Date finish = cal.getTime();

    String hql = "from " + Booking.TYPE + " b where not (b.dateRange.start
        + sqlFormatter.format(finish)
        + "' or b.dateRange.finish < '"
        + sqlFormatter.format(start)
        + "') and b.resource.id="
        + booking.getResource().getPrimaryKey()
        + " and b.status != 'CANCELLED' "
        + " and b.status != 'DENIED'";
}

```

```
searchResult = session.createQuery(hql).list();

found = RepeatUtil.expandRepeatables(searchResult, start, finish);
boolean doubleBooking=false, bufferInfringed=false;
bookingStart = booking.getDateRange().getStart().getTime();
bookingFinish = booking.getDateRange().getFinish().getTime();

for (Biskit b : found)
{
    Booking clash = (Booking) b;
    dr = clash.getDateRange();
    if (b.getPrimaryKey() == booking.getPrimaryKey()
        || dr.getStart().getTime() >= finish.getTime()
        || dr.getFinish().getTime() <= start.getTime())
        continue;

    if (dr.getStart().getTime() >= bookingFinish
        || dr.getFinish().getTime() <= bookingStart)
        bufferInfringed = true;
    else
        doubleBooking = true;
}

String msg;
if (doubleBooking)
    result.setMessage("Double bookings are not allowed");
else if (bufferInfringed)
    result.setMessage(
        "You must leave a gap of 15 minutes between bookings");

if (doubleBooking || bufferInfringed)
    result.setRejectionLevel(RejectionLevel.REJECT);
}
finally
{
    session.close();
}
```


6.6.3.5 The Booking Rule Validator

The **Booking Rule Validator** tab of the **Booking Rule Editor** provides a means of testing any user created [Booking Rules](#)⁶⁵². In particular, please note that when a *Booking Rule* is edited, it can be tested before it is saved, so that its behaviour can be tested to make sure it works as expected.

The way the **Booking Rule Validator** works, is to simulate creating a booking or updating a *booking* by changing the **Booking** change type. The **Booking** tab is where the values of the *booking* being created is specified, or, when simulating an update, this is replaced by two tabs **Old Booking** and **New Booking** where the old and new values for the simulated *booking* can be specified. Then decide which user the test is being run as. **Running** the test while pretending to be another user is a way to see how the *Booking Rule* would behave when a *booking* is created or update by somebody else.

Finally, the **Test Rule** button runs the test. The text area at the very bottom of the tab shows the output and the rejection level returned from the *Booking Rule* with information about what happened. If there's a message that would be presented to the user, or the *Booking Rule* provided some other output by calling [RulesResult.setOutput\(String\)](#)²⁷¹, then it will be shown in the **Message to show** user area.

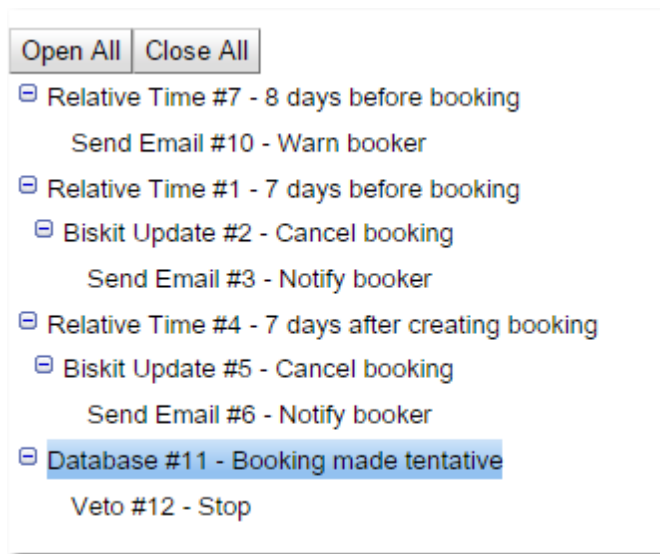
6.6.4 Tentative Bookings

A new [booking status](#)⁶⁵³ **Tentative** has been added. By default this *booking status* has been disabled, but can be easily enabled by adjusting pertinent [booking](#)⁶⁵² [permissions](#)⁶⁵⁴. An example implementation of how to use Tentative *bookings* has also been implemented in a [Workflow](#)³³⁴.

The permissions that need to be changed are:

- **Create->Booking->Nobody can create tentative bookings** (needs to be disabled)
- **Create->Booking->Anybody can create tentative bookings** (needs to be enabled)
- **Update->Booking->Nobody can make a booking tentative** (needs to be disabled)

The [Workflow](#)³³⁴ is defined below:



There are four event triggers.

1. Relative Time #7 which fires eight days before the start of the *booking*, if the *booking* is more than two days old and warns the [booker](#)⁶⁵² that the *booking* will be cancelled tomorrow if not approved.
2. Relative Time #1 which fires seven days before the start of the *booking*, if the booking is more than two days old and cancels the *booking* and then emails the *booker* informing of the cancellation.
3. Relative Time #4 which fires seven days after the *booking* was created, as long as its still in the future and cancels the *booking* and then emails the *booker* informing of the cancellation.
4. Database #11, which fires if the *booking* is being updated, if the *booking status* is being changed to **Tentative** and the *booking* was not created within the previous hour then the user is stopped from making the change and a message pops up to inform them why.

This is an example of how [Workflows](#)³³⁴ could be used to implement a **Tentative booking** strategy. Any or all of these options could be changed by the administrator by editing **Workflow Booking->Tentative Bookings**

6.7 Types And Groups

For various reasons, it is useful to be able to split users into sets of one sort or another. This is done in **Calpendo** using types and groups. These are two independent means of segregating users, projects and resources.



The Difference Between Types and Groups

The key thing to understand is that a user has precisely one type, but may belong to any number of groups.

Your **Calpendo** should be configured with types and groups that make sense for you. A default installation comes with no types and no groups. If your installation does use [User Types](#)⁶⁵⁶, then when a user registers, they will be asked what their type is. An administrator can override that and change it when they approve the user, but the user will always have a single type. When users have types, then the type can be used for [Permissions](#)³¹⁴, [Workflows](#)³³⁴, [Manual emails](#)¹⁸⁶ and [Booking Rules](#)²³⁵.

User Types are used if there is some logical separation of users that fits the notion of a user having precisely one type. For example, it may be that there are multiple departments in your company, with everybody belonging to only one department. If that were the case, then perhaps you would have as many *User Types* as there are departments.

[User Groups](#)⁶⁵⁶ can also be used for [Permissions](#)³¹⁴, [Workflows](#)³³⁴, [Manual Emails](#)¹⁸⁶ and [Booking Rules](#)²³⁵, the same as for types. However, there is greater flexibility due to the way a user can belong to multiple groups (or no groups). So, while a *user's type* may indicate their department (or whatever), groups could be created that reflect any sort of grouping that makes sense in your situation. Given this apparent flexibility in groups, one might ask what the point of types is. The answer is that it is sometimes useful to have a taxonomy in which every user has a *type*, whereas a user may belong to no groups at all.

In exactly the same way that a User has a *User Type* and may belong to *User Groups*, a [Project](#)⁶⁵⁴ has a [Project Type](#)⁶⁵⁵ and may belong to [Project Groups](#)⁶⁵⁴, and a [Resource](#)⁶⁵⁵ has a [Resource Type](#)⁶⁵⁵ and may belong to [Resource Groups](#)⁶⁵⁵. The same Booking Rule about having precisely one type and belonging to any number of groups applies to *projects* and *resources* in exactly the same way as it does to users. These types and groups can be used for [Permissions](#)³¹⁴, [Workflows](#)³³⁴, [Manual Emails](#)¹⁸⁶ and [Booking Rules](#)²³⁵.

[Bookings](#)⁶⁵² may also have a type. If you create one or more types in the **Types and Groups Editor**, then when somebody creates a *booking*, they will be asked for its type. However, there are no *booking groups* in **Calpendo**. One might want to use a [booking type](#)⁶⁵³ for several reasons. For example, in a clinical or mixed clinical and research environment, it may be required to categorise scanner *bookings* according to the nature of the subject of the scan. That is, to record whether they are fully healthy, walking patients, patients in a chair or patients in a bed.

If using *booking types*, then this allows the setting of different [Booking Rules](#)⁶⁵² or [Permissions](#)⁶⁵⁴ based on the *booking type*, and the different *booking types* can also be displayed differently on the [calendar](#)⁶⁵² so that the *booking type* is obvious at a glance. The *Booking Type* can also be used for [Permissions](#)³¹⁴, [Workflows](#)³³⁴, [Manual Emails](#)¹⁸⁶ and [Booking Rules](#)²³⁵.

6.8 Configuring Types And Groups

The **Types And Groups Editor** is used to configure the types and groups in the system. Before reading more about how to configure them, please take a look at [Types And Groups](#)²⁷⁹ which describes what a type and a group is, and how an item can have exactly one type but may be a member of many groups (including zero).

When it comes to configuring types and groups in **Calpendo**, consider first whether there is a good way to categorise the users, [projects](#)⁶⁵⁴ and [resources](#)⁶⁵⁵ such that they each fall into precisely one category.

For example, distinguish between students, staff and external users using that for the [user type](#)⁶⁵⁶. Alternatively if multiple departments use the *resources*, its possible to assign a *user type* according to which department a user belongs to.

[Project types](#)⁶⁵⁵ could be assigned according to the department they originated from, or perhaps according to thier funding source.

[Resource types](#)⁶⁵⁵ tend to be more obvious, but are just as important. Categorise *resources* as Room, Scanner, Person, Computer, Desk, etc.

Once the types have been chosen, then they can be used throughout **Calpendo**. For example, to apply a particular [Booking Rule](#)⁶⁵² to Scanner [bookings](#)⁶⁵² for users of a particular type, or for [projects](#)⁶⁵⁴ of a particular type. Types are used to control [Booking Rules](#)²³⁵, [Time Templates](#)²²⁶, [Permissions](#)³¹⁴, [Workflows](#)³³⁴ and [Manual Emails](#)¹⁸⁶.

Types And Groups Editor

The **Types and Groups Editor** page consists of three parts:

1. The menu tool bar: this is the tool bar at the top of the page, and all the buttons operate on all types and groups at a time.
2. The left pane: this allows the selection of a type or group or an item within a type or group
3. The right pane: this shows the detail of the currently selected type or group item

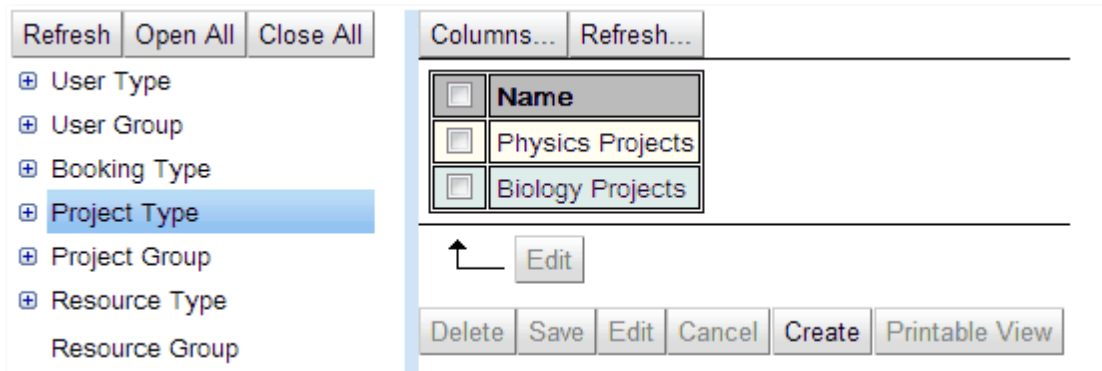


For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

The List View

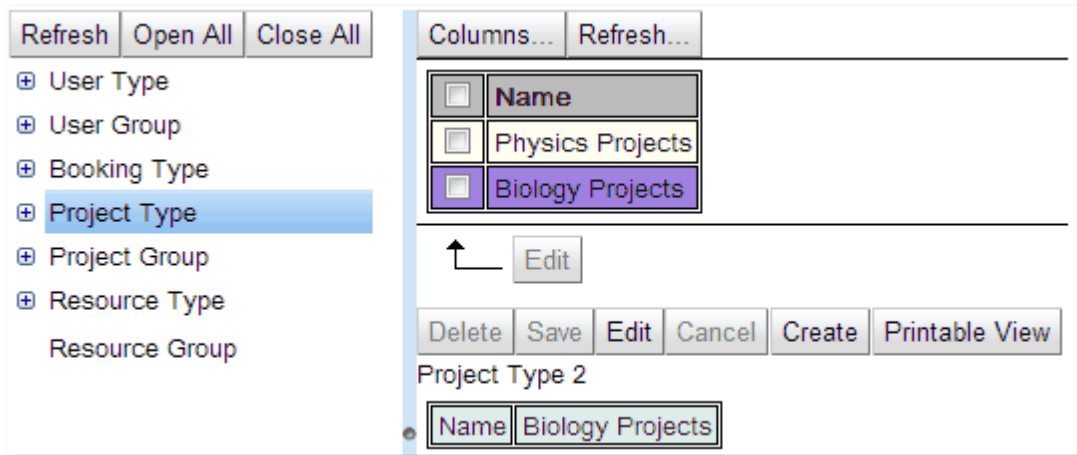
By clicking on one of the types or groups in the left hand pane, in the right hand pane will appear a list of all the entries for that type or group. This will include two menus, one above and one below the list.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.



Many items can be edited in one go quite easily in the list view, using the check boxes read the section on [How To Edit Multiple Items At Once](#)¹⁴⁰ in the [Data Explorer](#)¹³⁹ chapter.

To edit an individual item from this list click on it and it will appear below the list and the edit functions allowed will no longer be greyed out.



The Item View

Once a group or type has been opened up in the left hand pane select individual items to work on. Once selected these items will appear in the right hand pane.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Edit	Create	Create copy	Delete	References	History	Fullscreen
Name	Research					
Colour						
Border Width	3					
Border Style	3D Outset Border					

Edit	Create	Create copy	Delete	References	History
Name	Paul & Clare's group				
Users					
Login name	Given name	Other name	Family name		
paul	Paul		Robinson		
clare	Clare		Mackay		

[Booking Type](#)⁶⁵³

[User Group](#)⁶⁵⁶

Types and Groups appear slightly differently, Types are just a list of the [properties](#)⁶⁵⁵ that can be changed. A group has that list but also a list of all the appropriate items that are in its group. For instance a **User Group** will have a list of all the logins associated with that group.

Once in edit mode, edit any *properties* with [permission](#)⁶⁵⁴ to access and add or delete items from group lists.

Cancel	Save			
Name	Paul & Clare's group	Edit individual properties		
Users				
Please select a User to add		Add items to the group list		
<input type="checkbox"/>	Login name	Given name	Other name	Family name
<input type="checkbox"/>	paul	Paul		Robinson
<input checked="" type="checkbox"/>	clare	Clare		Mackay
Remove		Remove selected items from the group		

To remove items from the list first select the items to be removed using the check box and click the **Remove** button.

To add items to the list use the pull down to get a full list of possible items and select the one required.

Resource Usage Outcome

The **Types And Groups Editor** is also used to configure the outcomes that may be recorded when **Calpendo** is configured to collect [resource usage](#)⁶⁵⁵ information.

The screenshot shows the 'Types And Groups Editor' interface. On the left, a tree view lists various categories: User Type, User Group, Booking Type, Project Type, Project Group, Resource Type, Resource Group, and Resource Usage Outcome. Under 'Resource Usage Outcome', the 'OK' outcome is selected. On the right, a table displays the configuration for the 'OK' outcome.

Edit	
Name	OK
Border Colour	
Border Width	2
Border Style	Solid
Content Shade	MEDIUM
Display Order	1

Calpendo comes with some default outcomes, the administrator can add as many additional ones as required as well as configuring the choice of colour and borders for each outcome.

6.9 Configuring Resources And Locations

[Resources](#)⁶⁵⁵ and **Locations** are configured in the **Resource Editor** which is, by default, available on the menu as **Admin-->Resource Editor**.

Locations

Locations have a name and a parent, this allows a hierarchy of locations to be created.

The screenshot shows a form for creating or editing a location. It has two input fields: 'Name' and 'Parent' (with a dropdown menu showing 'Please select a Location'). Below these is a tab labeled 'Children'. Under the 'Children' tab, there is a message box that says 'No locations found'. At the bottom of the form, there is a row of buttons: 'Remove', 'Add', 'Apply', 'Edit', 'Cancel', 'Create', 'Create Copy', and 'Printable View'.

Also once a location is created the child locations can be created directly using the **Create** button in the **Children** tab.

Resources

When you create a *resource* you get the following pop up:

The screenshot shows a resource configuration pop-up. It contains a table with the following settings:

Name	Confocal
Location	Campus 1
Type	Microscope ▼
Project Required	Any Project Required ▼
Manager User Group	Microscope managers ▼
Booking Subtype	Booking
Require Reason for Cancellations	<input checked="" type="checkbox"/>
Colour Group ID	
Drag And Drop Bookings	Supports drag and drop ▼
Allocate Calendar Column	<input checked="" type="checkbox"/>
Collect Actual Usage	<input checked="" type="checkbox"/>
Allow Old Changes	<input checked="" type="checkbox"/>
Enable pre-defined time slots	<input checked="" type="checkbox"/>
Import bookings from external iCal feed	<input checked="" type="checkbox"/>
Custom Booking Tooltip	<input checked="" type="checkbox"/>

Below the table are tabs for: Actual Usage, Allowing Changes to Old Bookings, Time Slots, iCal Import, Picture, **Groups**, and Booking Tooltip. The 'Groups' tab is selected, showing a list of resources with checkboxes for 'Name' and 'Scanners', and buttons for 'Apply' and 'Refresh'.

Resources have the following properties:

Property	Description
Name	This is the name of the <i>resource</i> . It can be anything required, except that it should be unique and it is better if the name is not very long so that its display doesn't take up much space.
Location	It is not necessary to set the location of a <i>resource</i> , but there are many <i>resources</i> , it may help to specify the location of each. The location can be used with Permissions ³¹⁴ and Workflows ³³⁴ .
Type	Resource types ⁶⁵⁵ are configured using the Types and Groups Editor as described in Configuring Types And Groups ²⁸⁰ . <i>Resource types</i> can be used for Time Templates ²²⁶ , Permissions ³¹⁴ and Workflows ³³⁴ .
Project Required	<p>There are three possible values:</p> <ul style="list-style-type: none"> • Project Not Required, • Any Project Required, • Authorised Project Required. <p>This allows a choice of whether bookings⁶⁵² for this <i>resource</i> must specify a project⁶⁵⁴. There might be different types of <i>resource</i> and <i>projects</i> are only required for some. For example, <i>projects</i> may need to be associated with <i>bookings</i> for a Microscope, but not for a conference room. Also there is the option of allowing <i>projects</i> to book a <i>resource</i> only if they have an entry for that <i>resource</i> in their Project Resource Settings⁶⁵⁴.</p>
Manager User Group	To allow the administrator to point to a group of users that will manage this device. This property can then be used in Rules , Permissions , Workflows etc. This also gives the members of the group access to all Projects when making <i>bookings</i> .
Booking Sub-Type	To allow the administrator to define different Booking Biskit Types ⁶⁵² for different <i>resources</i> . All new Booking Biskit Types are created in the Bakery ⁶⁵² and must inherit from Booking . They then may have their own list of properties ⁶⁵⁵ , allowing the set up of different <i>booking</i> information for different <i>resources</i>
Require Reason for Cancellations	By default a user can cancel a <i>booking</i> without giving a reason, if this <i>property</i> is set to True then any cancellation of this resource type ⁶⁵⁵ will require a reason to be input.
Colour Group ID	Default colour for this resource for all new users.
Drag and Drop Bookings	Indicates whether the calendar supports drag and drop for this resource.
Allocate Calendar Column	This allows the user to specify whether this <i>resource</i> should always use a column in the calendar when in Always present mode. Some <i>resources</i> such as Annual Leave may only be required to be shown in a column when someone has booked a half day or some hours off, the rest of the time as full day <i>bookings</i> they would be shown at the top of each days columns, and a column for each day is not required.

Property	Description
Collect Actual Usage	Indicates whether actual <i>resource usage</i> information should be collected for this <i>resource</i> .
Allow Old Changes	By default, <i>bookings</i> that are in the past or in progress cannot be modified. However, it may be necessary to create or modify past <i>bookings</i> so information relating to actual resource usage ⁶⁵⁵ can be recorded after the event. In order to do this set Allow Old Changes and also add <i>properties</i> to <i>Bookings</i> using the Bakery ⁶³⁷ to record any information that needs to be captured. It is also possible to specify the types of changes allowed and the time period these changes must be made in.
Enable pre-defined time slots	This enables the ability to create pre defined time slots for <i>bookings</i> . Ticking the check box enables the Time Slots tab where time slots can be set up for each of the <i>resources</i> for each day of the week.
Import bookings from external iCal feed	Allows the importing of an iCal feed into Calpendo and its updating at regular times. These imported bookings cannot be updated or deleted, they are completely controlled by the iCal importer. A booking that is removed from the iCal feed is deleted inside Calpendo.
Custom Booking Tool tip	This allows the user to specify which <i>properties</i> will appear in the tool tip that appears when the cursor rolls over a <i>booking</i> in the Booking Calendar . This tool tip can now display custom <i>properties</i> .
Self	Self is always a link back to the <i>resource</i> in question. This is particularly useful for Permissions ⁶⁵⁴ where a <i>resource</i> needs to be hidden from some people. Use a condition ⁶⁵³ like value of self equals 3T where 3T is selected from a drop-down of <i>resources</i> . Previously, a <i>condition</i> like value of name equals 3T would have been required, where the text 3T needed to be typed in. The <i>Permission</i> would have failed if the <i>resource</i> 3T were ever renamed. This is not seen in the Resource Editor but will be available when creating <i>conditions</i> .

The **Resource Editor** shows a list of locations and *resources*. Select a location or *resource* and press the **Edit** button to modify it, or press the **Create** button to create a new one. For more information on how the **Resource Editor** works read the chapter on [Configuring Types and Groups](#)²⁸⁰ as the **Resource Editor** works in the same fashion.

After creating a new *resource* go to the [Bookmark Manager](#)¹⁸² and make sure to add the resource to the **System Bookmarks->Show All [bookmark](#)**⁶⁵³ so that all users will have the *resource* in the default *bookmark*.

Enabling Actual Resource Usage Recording

Resource Usage is configured by the [Resource Usage](#)⁵³⁰ tab in the [Global Preferences](#)⁵⁰⁹ page, and also by settings on each *resource*. If **Collect Actual Usage** is set to true for a *resource*, then additional information will be requested, as shown below:

Setting	Description	
Font size	The Resource Usage Recorder page is normally used on a computer that is next to the equipment whose <i>resource usage</i> is being recorded. If it is required to view the computer from across a room, then it might be preferable to enable the large font setting for that <i>resource</i> .	
Access protection	Specifies the IP address of a computer that will use the Resource Usage Recorder page for this <i>resource</i> . This can be set to work from any address, or restrict it to one IP address.	
Resource User Selectable	Allows the administrator to define whether the person creating the <i>resource usage</i> can select the user creating the <i>resource usage</i> or whether it defaults to the logged in user.	
Usage Session ID Template	The Resource Usage Recorder page will provide a unique session ID for each <i>resource usage</i> session. However, the way that session ID is formatted can be configured, and also what goes into it. This editor allows entry of any text and selection of any number of <i>properties</i> that are stored on the <i>resource usage</i> itself. The items that might be useful are:	
	Property	Description
	id	A unique integer assigned to the <i>resource usage</i> .
	projectUsageCount	This is the number of <i>resource usages</i> recorded for the usage's <i>project</i> so far.
	resourceUsageCount	This is the number of <i>resource usages</i> recorded for the usage's <i>resource</i> so far.
	resource	The <i>resource</i> itself is also accessible. For example, the session ID may be required to include the <i>resource's</i> name.

The outcomes that may be recorded for each *resource usage* can also be customised. To do this, use the [Types And Groups Editor](#).²⁸⁰

Note that there are *resource usage* menu items that remain hidden unless there is at least one *resource* with its **Collect Actual Usage** *property* to true.

Creating Bookings In The Past, Or Enabling Changes To Past Or In Progress Bookings

To be able to create *Bookings* in the past, or allow changes to past *Bookings* or those currently in progress, set **Allow Old Changes** to **true** for a *resource*.

Then define whether some or all changes can be made. If **Allow some changes** is specified then the *resource*, *status* nor the time can be changed but all other *properties* may be (this can of course be managed by using [Permissions](#)³¹⁴). If **Allow any changes** is specified then all *properties* may be changed unless denied by [Permissions](#)³¹⁴.

The screenshot shows a form titled 'Allowing Changes to Old Bookings'. It has two sections: 'While booking is in progress' and 'After booking has finished'. Both sections have a dropdown menu set to 'Do not allow any changes'.

If **Allow some changes** is chosen then specify whether to allow changes for a particular time period or always (Thereafter).

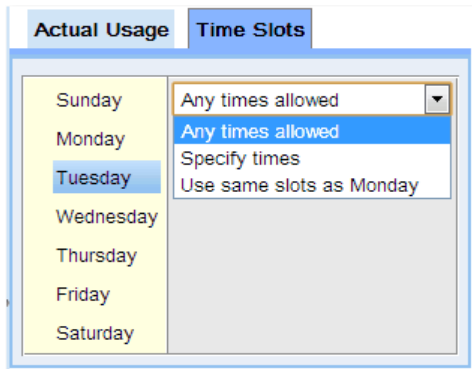
The screenshot shows the same form. In the 'While booking is in progress' section, the dropdown is set to 'Allow some changes'. In the 'After booking has finished' section, the dropdown is set to 'Allow some changes', and the 'Until' dropdown is set to '5 minutes'.

If **Allow any changes** is chosen for **After booking has finished** it can be specified whether to allow them for a particular time period or always, but it can also be specified whether to **Allow some changes** after that time. All times are taken from the time the *booking* finishes. It will not be possible to put in a shorter time into the **Allow some changes** option than is in the **Allow any changes** option.

The screenshot shows the same form. In the 'While booking is in progress' section, the dropdown is set to 'Allow some changes'. In the 'After booking has finished' section, the dropdown is set to 'Allow any changes', and the 'Until' dropdown is set to '5 minutes'. Below this, there is a section for 'and then' with a dropdown set to 'Allow some changes' and the 'Until' dropdown set to '6 minutes'. Below that, there is a section for 'and then no changes thereafter' with a dropdown set to 'Thereafter'.

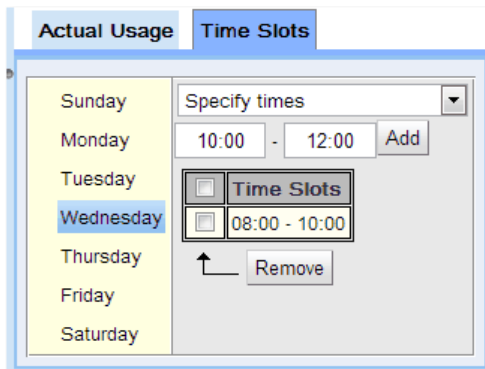
Enabling Predefined Time Slots For Bookings

If the **Enable predefined time slots** box is checked then the **Time Slots** tab can be viewed.



For each day the time slots available can be defined.

Setting	Description
Any times allowed	This is the normal operation where anyone can book any time slot.
Specify times	Define any time slots required.
Use same slots as ???day	Use the time slots defined for a different day. If that days time slots are changed then they will change on this day as well, with the exception that if the day that is being copied is changed to Any times allowed , then the current day will keep the slots defined before that happened and its option will change to Specify times .

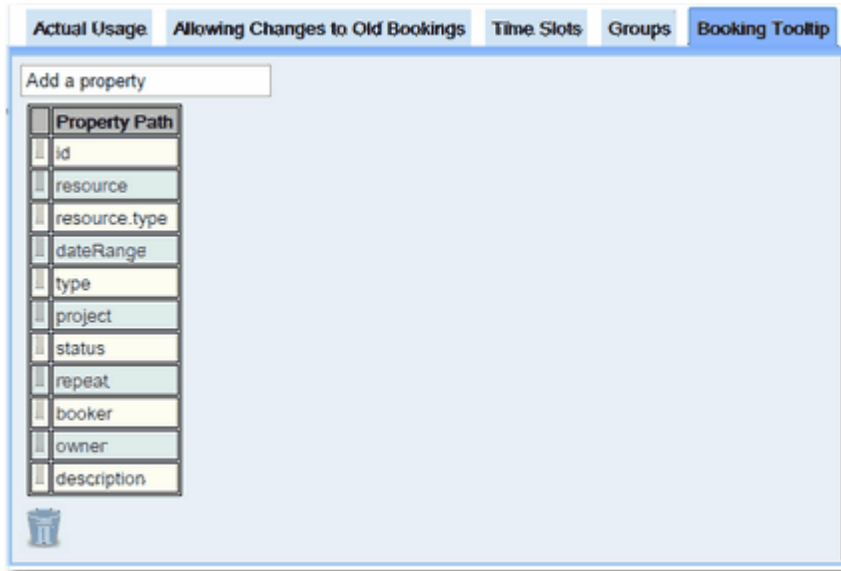


To specify the times choose the start time and end time, and press the **Add** button. Once the first one is created Calpendo will set up the times for the next slot to be of the same duration and starting at the end of the current time slot, so if that is what is required just press **Add** again. Click the check box next to a slot and use the **Remove** button to delete any slots that are no longer required. The final slot of the day will end at 23:59 not 24:00.

Also required is that the **Predefined Slots Booking Rule (Predefined slots must be adhered to)** is the default Rule) is enabled.

Custom Booking Tooltip

This allows the user to specify which *properties* will appear in the tool tip that appears when the cursor rolls over a *booking* in the **Booking Calendar**. This tool tip can now display custom *properties*. Create a list of the *properties* required by selecting within **Add a property**, drag and drop them into the order required. Any *property* not required drag to the bin.




iCal Import

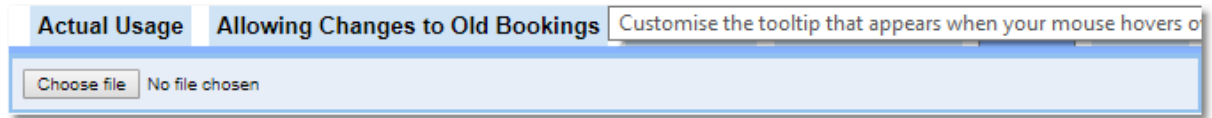
Add support for importing bookings from an external iCal feed.

Setting	Description
iCal Feed URL	Holds the URL the feed is coming from
iCal Fetch Delay (minutes)	Defines the update frequency of the iCal feed.

If an iCal feed contains a repeat booking, then it will be imported as multiple non-repeating bookings, all having the same iCal ID.

Picture

This allows the administrator to upload a picture which will be displayed on the calendar when the user moves their mouse over the resource name at the top of the column. This does not work if resources are sharing a column. Click , and upload the picture to be viewed.

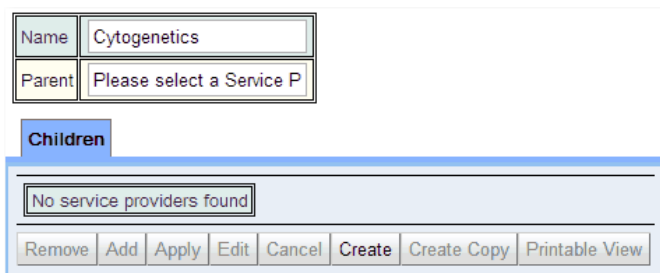


6.10 Configuring Services and Service Providers

[Service Providers](#)⁶⁵⁵ and [Services](#)⁶⁵⁵ are configured in the **Service Manager** which is, by default, available on the menu as **Admin-->Service Manager**.

Service Providers

Service Providers have a name and a parent, this allows a hierarchy of *Service Providers* to be created.



Also once a *Service Provider* is created the child *Service Providers* can be created directly using the **Create** button in the **Children** tab.

Service Orders

[Service Orders](#)⁶⁵⁵ are created in the [Bakery](#)⁶⁵² by creating a [Biskit Type](#)⁶⁵² which inherits from the *Service Order Biskit Type*, and has all the [properties](#)⁶⁵⁵ defined to create the *Service Order* and monitor it. Create one *Service Order Biskit Type* for each type of *Service* that will be required.

Services

When you configure a *Service* you get the following pop up:

Name	Sanger DNA Sequencing
Location	Please select a Location
Provider	Molecular Research
Project Required	Any Project Required
Enabled	true

[Description](#)
[Users](#)
[User Groups](#)
[Orders](#)

Order Definition

Sanger Sequencing Order

Property	Resource
Booking	Fortessa

Services have at least the following [properties](#)⁶⁵⁵:

Property	Description
Name	This is the name of the <i>service</i> . It can be anything required, except that it should be unique and it is better if the name is not very long so that its display doesn't take up much space.
Location	It is not necessary to set the location of a <i>service</i> , but there are many <i>services</i> , it may help to specify the location of each. The location can be used with Permissions ³¹⁴ and Workflows ³³⁴ .
Provider	The name of the <i>service provider</i> .
Project Required	<p>There are three possible values:</p> <ol style="list-style-type: none"> Project Not Required, Any Project Required, Authorised Project Required. <p>This allow a choice of whether a requester of this service must specify a project⁶⁵⁴. Typically, there might be different types of <i>services</i> and <i>projects</i> are only required for some. Option three allows <i>projects</i> to book a <i>service</i> only if they have an entry for that <i>service</i> in their project service settings⁶⁵⁴.</p>
Enabled	Allows the service to be disabled/enabled.
Description	A full description of the <i>service</i> .
Users	Those users that will be emailed when an <i>order</i> is made for this <i>service</i> .
User Groups	Groups ⁶⁵⁶ of users that will be emailed when an <i>order</i> is made for this <i>service</i> .
Order Definition	The <i>Service Order Biskit Type</i> to be used to make an order for this <i>service</i> . This allows the administrator to set up different <i>properties</i> to be input for each <i>service</i> that could be ordered.
Booking Properties	These properties allow the administrator to specify the resource(s) that will be used to fulfill the service.

6.11 Recording Actual Usage

Manual Recording

This methodology is good for **MR, PET, CT** scanners where there is a dedicated operator at the machine, and it is acceptable for them to go to a **Calpendo** page and click a button when the session starts and again when it stops. This page allows them to provide information about who the user is, which [booking](#)⁶⁵² this usage is associated with, what the [project](#)⁶⁵⁴ is, and any other custom information that needs to be recorded (such as the patient details). It also allows the recording of an "outcome" to specify whether the scan went well or there was a problem, perhaps with the scanner, a no-show or something else. (This is the only system that works with **Calpendo LITE**).

This creates [Resource Usage](#)⁸⁵ information on the [Resource Usage Calendar](#)⁸⁸, linked to actual *bookings*.

View the [Resource Usage](#)⁸⁵ chapter for more information on how this works.

Windows PC Based Instruments Where Everybody Logs in to the PC with a Unique Login Name

This methodology uses scripts that can be scheduled to run using the **Windows Scheduler**. These scripts will tell **Calpendo** when somebody logs in, and this information is updated every 5 minutes (or however frequently its been configured to update), with who is logged in and what processes are running.

This can be used to calculate actual usage, including checking for periods when particular software is running. This method also includes sending regular updates back to **Calpendo** using **http**, so that we know how long software is being used for, and also works properly when the **PC** crashes. These updates are loaded into a database table, [Workflows](#)⁶⁵⁶ can then be created to use this information to either create [Resource Usage](#)⁸⁵ information on the [Resource Usage Calendar](#)⁸⁸, to adjust *bookings* on the [Booking Calendar](#)⁶⁵², or any other way a user would like to store usage information.

Windows PC Based Instruments Where We Can Install pGina

(<http://pgina.org/>)

pGina is an open source **Windows** authentication and authorisation system. Along with a **Calpendo** plugin, this supports recording actual usage regardless of whether everybody has a different login on the **PC**.

On **Windows Pro** and **Windows Enterprise**, it can also prohibit running certain software on the computer, while still allowing them to log in.

This method also includes sending regular updates back to **Calpendo** so that we know how long software is being used for, and also works properly when the **PC** crashes. These updates are loaded into a database table (the same one used in the above methodology), [Workflows](#)⁶⁵⁶ can then be created to use this information to either create [Resource Usage](#)⁸⁵ information on the [Resource Usage Calendar](#)⁸⁸, to adjust *bookings* on the [Booking Calendar](#)⁶⁵², or any other way a user would like to store usage information.

Calpendo Activity Recorder (CAR)

CAR (Calpendo Activity Recorder) is a stand-alone **Windows** native program that runs both as a service and also an instance in each user session.

- Authenticates the user, using the administrator's choice of authentication method such as local **Windows** authentication, **LDAP**, **Active Directory** or sending a username/password to **Calpendo**.
- Keeps track of particular processes that are running.
- Asks **Calpendo** for authorisation for the user to run those processes.
- When **CAR** finds one of its target processes running, and the user has not been authorised, then **CAR** will can kill or close the process, or warn the user they shouldn't be using it, or do nothing.
- All activity is recorded in the **Calpendo** database.
- The format of the data sent over the network has changed, which means that if either the pGina or PowerShell are already installed as methods of gathering remote actual usage information, then Calpendo will need to be upgraded to the latest versions to remain compatible.
- The data recorded in the Calpendo database has also changed slightly.

Sign In/Sign Out System

This methodology allows the administrator to put a tablet next to an instrument, or possibly use a computer, and have a web page that lets people sign in. At the end, they sign out again. The tablet would be permanently set to be talking to a specific **URL**, which would display a custom **Calpendo** page for this purpose. The page could not only ask for login details, but could also display such things as a clock, to show how long they have been logged in, how long until the next *booking* is due to start, or just general information about the next *booking*. This page is generated by a *Workflow* inside **Calpendo**.

Also, **Calpendo** can be configured to turn a computer monitor on and off when somebody signs in and out allowing and stopping access. This is done by using some hardware on the network which the **Calpendo** talks to, to switch a relay on and off which switches the monitor on and off.

Workflows would use the login/logout information to either create [Resource Usage](#)⁸⁵ information on the [Resource Usage Calendar](#)⁸⁸, to adjust *bookings* on the [Booking Calendar](#)⁶⁵², or any other way a user would like to store usage information.

Actual Usage Data Storage

The data from the remote **Actual Usage** systems is stored in two [Biskits](#)⁶⁵², **Remote Process** and **Remote User Log**.

Remote Process:

Property	Type	Details
name	String	Name of the process.
userLog	Biskit	Pointing to Remote User Log (many to one)
start	Datetime	Time the process started.
processID	Int	Remote systems ID for the process

Remote User Log:

Property	Type	Details
Type	JavaEnum	Possible values: AUTHENTICATION, AUTHORISATION, PING, LOGOUT.
Accepted	Boolean	Whether the user login was accepted
Comments	String	Additional comments to be stored.

Process Tab	Type	Details
Processes	Set	Pointing to Remote Process

JSON Tab	Type	Details
JSON		.

Time Tab	Type	Details
Created	Datetime	Time the log data was created.
RemoteTime	Datetime	Time on the remote device.

User Tab	Type	Details
Authorized Login Name	String	Login name of the authorised user.
OS Domain Name	String	Name of the computer domain.
OS Login Name	String	Name of the user.
User	Biskit	Pointer to User Biskit

Environment Tab	Type	Details
Hostname	String	Name of the Host sending the usage information.
OS Version	String	OS Version of the host.
IP Address	String	The IP Address Calpendo thinks the system is at. This can vary due to routing, address translation.
Declared IP Addresses	String	The absolute IP Address(es) of the system. The address the system thinks it is.
Remote Activity Recorder	String	Specifies which software is being used to return the Remote User Log information.
Activity Recorder Version	String	Holds the version number of the activity recorder.

6.11.1 pGina

Before installing the **Exprodo** plugin, make sure the **pGina** application is installed and correctly working on the computer. Please refer to the **pGina** official instructions at <https://github.com/pgina/pgina/wiki/Install>.

Following documentation refers to **pGina** 3.1.8.0, which was the latest stable version, at the time of writing this guide.

The **Exprodo** plugin is made up of two DLL files:

- **Newtonsoft.Json.dll** (external library)

and depending on your Operating System:

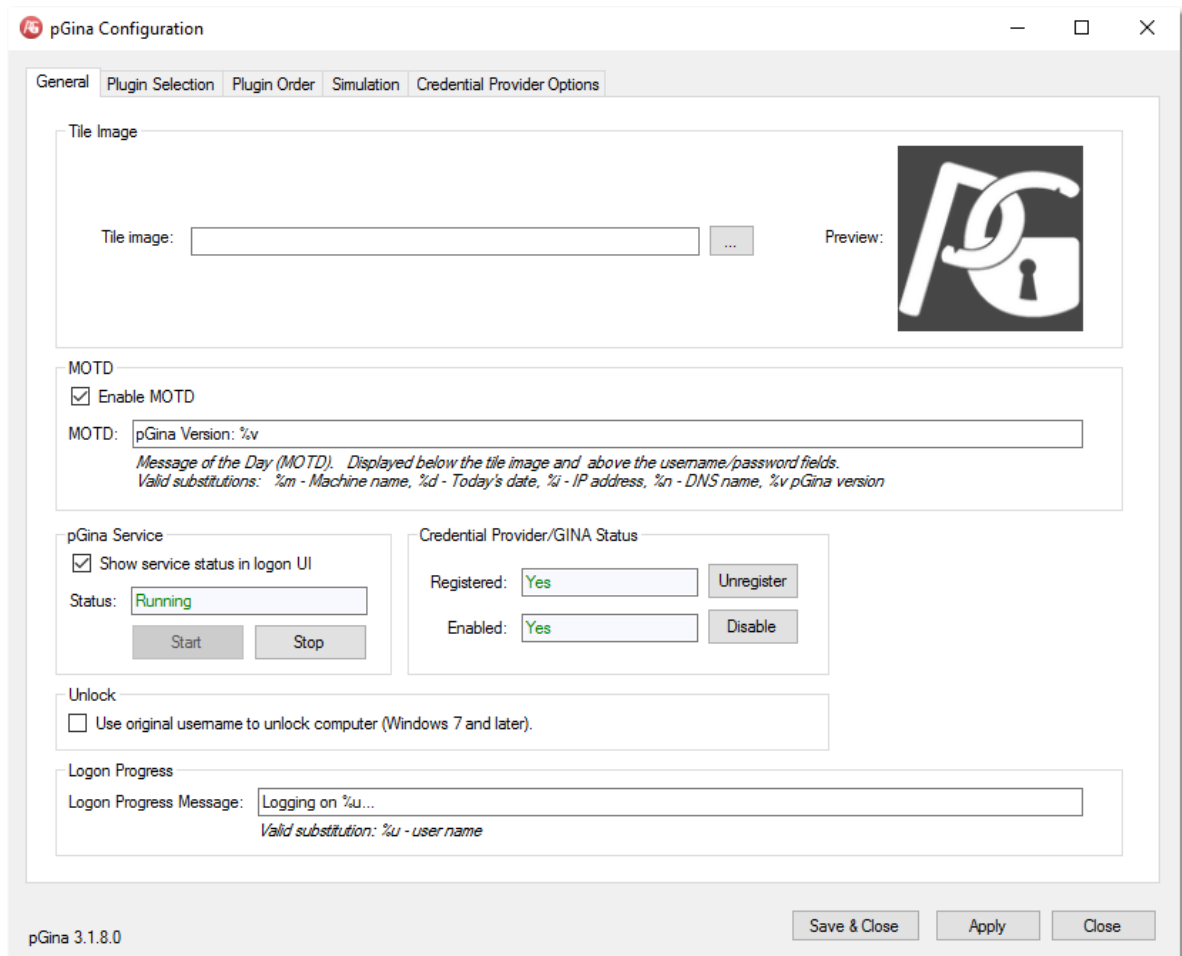
- **pGina.Plugin.Exprodo.dotNet45.dll** (.NET 4.5 compatible - Windows 7 and later)
- **pGina.Plugin.Exprodo.dotNet40.dll** (.NET 4.0 compatible - Window XP)

Both DLLs are provided in the tar.gz download file that contains **Calpendo**. They are located in the **INSTALL\pGina** folder.

Plugin Initial Installation

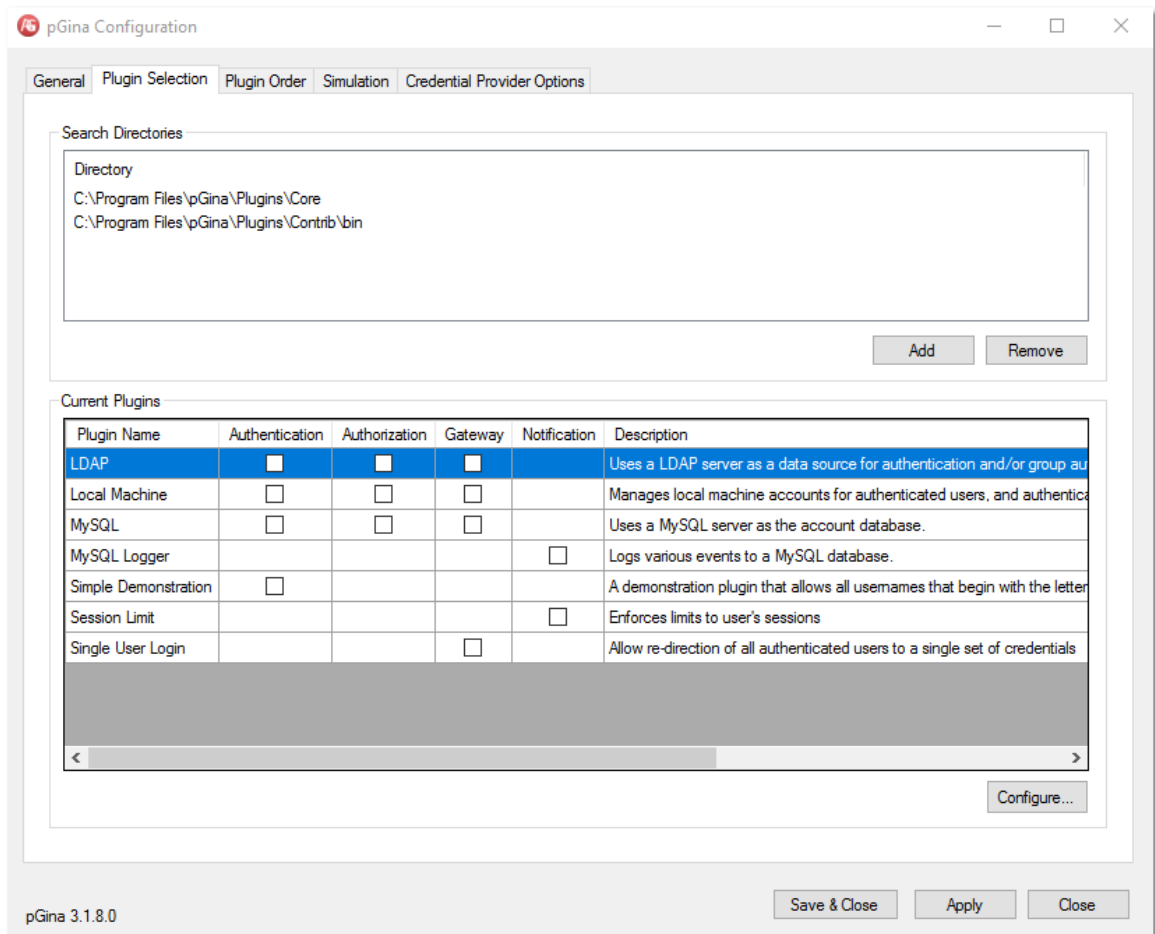
Both **pGina.Plugin.Exprodo.dotNet45.dll** (or **pGina.Plugin.Exprodo.dotNet40.dll**) and **Newtonsoft.Json.dll** need to be copied into the **pGina** directories. To see which one, follow these steps:

1. Start **pGina** configuration application, by locating **pGina** in your **Start** menu and clicking on the relevant icon:



2. From "**General**" Tab -> **pGina Service** section > click on **Stop**. This will stop **pGina** from running.

3. From "**Plugin Selection**" Tab -> Look at the "**Search directories**" section.
They should be:
c:\your installation root\pGina\Plugins\Core
c:\your installation root\pGina\Plugins\Contrib (or\Contrib\bin)

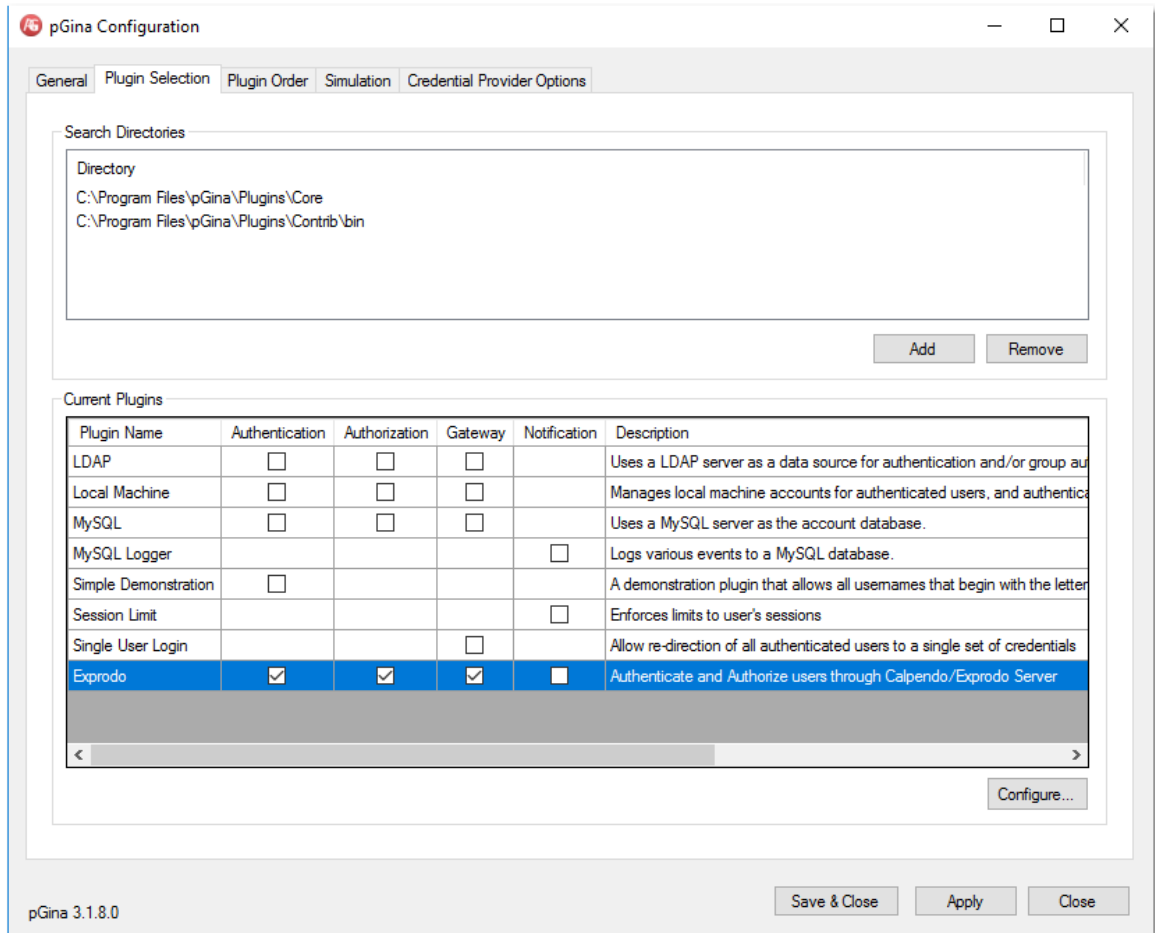


According to **pGina** documentation all external plugins should be added in
“..**pGina\Plugin\Contrib**” directory (or “..**pGina\Plugin\Contrib\bin**”).

(It's not mandatory to use one of these directories; any directory may be used as long as they are added to the "**Search Directories**" list.)

4. Copy and paste “**pGina.Plugin.Exprodo.dll**” and “**Newtonsoft.Json.dll**” into ..
pGina\Plugin\Contrib (or ..\pGina\Plugins\Contrib\bin).
5. Click on "**Save & Close**" button.
6. Restart the **pGina** Configuration application and in the **pGina Service** section, on the **General Tab**, click on the **Start** button.

7. The **Exprodo** plugin will appear in the “**Current Plugins**” list showed in “**Plugin Selection**” Tab with all services provided: Authentication, Authorization, Gateway and Notification.



If the **Plugin** does not appear in the list, it is possible that the **DLL** files have been blocked by **Windows** and need to be unblocked for use. Follow the steps below:

Updating Plugin









In order to update the Exprodo plugin follow the next steps:

- Stop pGina service
- Go to the directory where your old exprodo plugin is located
- Replace the old plugin with the new version
- Restart pGina service (before doing that be sure that there is only one exprodo plugin in the directory).

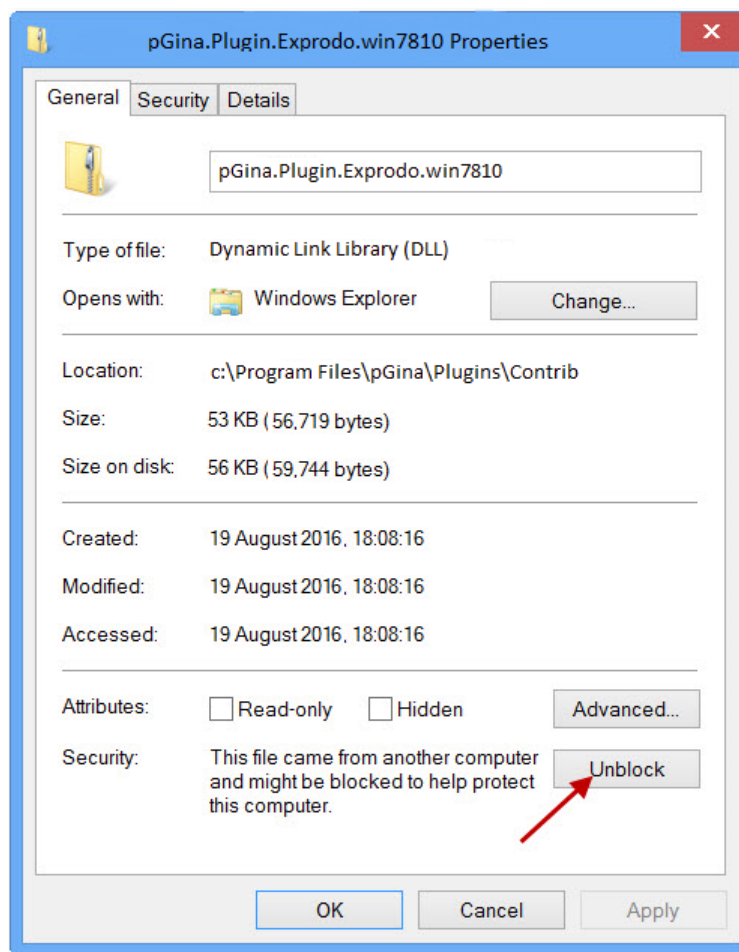
All your previous settings will be saved.

Unblocking the DLL files

1. Open **Windows Explorer** and locate the **pGina** folder that contains the files e.g. **C:\Program Files\pGina\Plugins\Contrib**
2. The following files should be displayed:

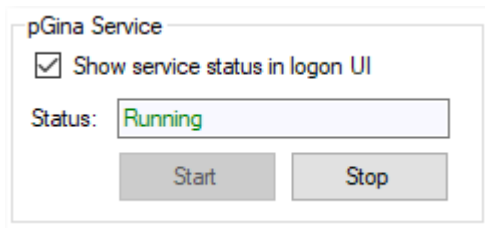
Name	Date modified	Type	Size
 Abstractions.dll	6/4/2013 12:02 AM	Application extens...	38 KB
 log4net.dll	12/20/2012 9:00 PM	Application extens...	264 KB
 Newtonsoft.Json.dll	5/4/2017 2:26 PM	Application extens...	514 KB
 pGina.Plugin.Email.dll	6/4/2013 12:03 AM	Application extens...	25 KB
 pGina.Plugin.Exprodo.win7810.dll	5/4/2017 2:26 PM	Application extens...	56 KB
 pGina.Plugin.RADIUSPlugin.dll	6/4/2013 12:03 AM	Application extens...	39 KB
 pGina.Plugin.UsernameMod.dll	6/4/2013 12:03 AM	Application extens...	44 KB
 pGina.Shared.dll	6/4/2013 12:02 AM	Application extens...	30 KB

3. Right click on the first file, **Newtonsoft.Json.dll**. There should be an **Unblock** button at the bottom of the **General** Tab. Click on that button and then select **Apply** and **OK**.



4. Repeat this process for **pGina.Plugin.Exprodo.dotNet45.dll**.

5. Re-open the **pGina** Configuration application and click on the **Stop** and then **Start** buttons to restart the service.



6. Now click on the **Plugin Selection** screen, you should have the **Exprodo Plugin** in the list:

Current Plugins					
Plugin Name	Authentication	Authorization	Gateway	Notification	Description
MySQL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Uses a MySQL server as the account database.
MySQL Logger				<input type="checkbox"/>	Logs various events to a MySQL database.
Simple Demonstration	<input type="checkbox"/>				A demonstration plugin that allows all usernames that begin with the
Session Limit				<input type="checkbox"/>	Enforces limits to user's sessions
Single User Login			<input type="checkbox"/>		Allow re-direction of all authenticated users to a single set of creden
Email Authentication	<input type="checkbox"/>				A plugin that authenticates against a POP or IMAP server.
Exprodo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Authenticate and Authorize users through Calpendo/Exprodo Serve
RADIUS Plugin	<input type="checkbox"/>			<input type="checkbox"/>	A RADIUS Authentication and Accounting Plugin
Modify Username Plugin	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Modify the username at various stages of the login process

Configuration

Before starting to use it, the **Exprodo** plugin needs to be configured.

1. Click on the **Plugin Selection** Tab. (see image above)
2. Left click on the **Exprodo** plugin to select it.
3. Click on the "Configure..." button. The **Exprodo - Plugin Setting** screen will be displayed.

Exprodo - Plugin Setting

Calpendo/Exprodo Web Link

☐ Computer Monitoring Every minutes

Processes to monitor

▶	
*	

Authentication / Authorisation

Authentication Methods

▶	
*	

Local Account

User Name

Password ☐ Show/Hide

Gateway

Local Group

Notification

☐ Log Out

Note: Each setting group is related to a specific feature of the plugin. Configure only the ones relevant to the system needs.

Exprodo Plugin Settings

Function	Description
Calpendo / Exprodo Web Link	This is the web link of the Calpendo server to which the plugin will send all the information to be stored. For example, https://yourCalpendoServer
Computer Monitoring	<p>If enabled, the plugin will activate a repeating timer that will cause a notification to be sent to the Calpendo server with actual usage of the current machine and thereby indicate who is using the computer.</p> <p>This information can be improved with the list of processes users want to monitor.</p> <p>Note: To activate any change to Process Monitoring, Stop and Start the pGina Service.</p>
Processes to Monitor	<p>This contains a list of process names that the plugin will monitor. A notification will be sent to the Calpendo server, with the following information:</p> <ul style="list-style-type: none"> - name of running process - starting dateTime - name of logged-in user <p>The name of a process is the name of the actual file executed, without any extension or path information. For example notepad rather than notepad.exe.</p> <p>Note: To activate any change to Process Monitoring, Stop and Start the pGina Service.</p>
Authentication Methods	This contains a list of authentication methods stored in the Calpendo server. They will be used when the Exprodo plugin attempts to authenticate users trying to login.
Local Account: User Name	<p>This is the local Windows account login name, with which users authenticated by the Calpendo server will login in to the local machine.</p> <p>The User Name can be selected from the existing local users.</p>
Local Account: Password	This is the password of the selected Local Account.
Local Group	<p>This is the name of the group that should be assigned to users authorised by the Exprodo plugin. The group can be selected from the existing local groups.</p> <p>If your operating system is Windows Home Edition, then two advanced options are provided:</p> <ul style="list-style-type: none"> - adding a new group

Function	Description
	<p>- removing an existing group (only if it was previously added by Calpendo)</p> <p>If using Windows Professional or Windows Enterprise, then please use the Windows-provided methods for adding and removing groups.</p>
Advanced Options (Windows Home Edition only)	This will open a new window to add or remove local groups.
Add New Group (in Advanced Options)	Allows the addition of a new group that will be immediately recognised for selection. All groups added with the Exprodo plugin will have the Description property set to Added by Calpendo .
Remove Existing Group (in Advanced Options)	Allows the removal of groups previously added by Calpendo .
Logout	If enabled, the plugin will send "logout" action information to the Calpendo Server whenever any user authenticated and/or authorised by Calpendo logs out.

Exprodo Plugin Services

The **Exprodo** plugin provides four independent services (Authentication, Authorization, Gateway and Notification) individually selected and one service (Computer Monitoring) activated when the plugin is loaded into **pGina**.

Computer Monitoring

This service sends actual computer usage to the **Calpendo** server. There are two kinds of notification:

- Simple
- with Processes

"Simple Notification" notifies the **Calpendo** server which user is using the computer. This notification will still be sent if nobody is using the computer, but it will be empty.

All the information will be stored in the **Remote User Log** biskit and the data can be reviewed by clicking on **Search -> Search**, in **Calpendo**, selecting **Remote User Log**, in the **Search for** field, and then clicking on **Go**.

Type	Created	Remote Time	Login Name	Domain Name	IP Address	Accepted
Ping	16 Nov 2016 16:20	16 Nov 2016 16:20	mary or <empty>	DESKTOP-ONE	150.150.10.10	<True> or <False>

User	Remote Authentication Method	Remote Activity Recorder	Comments
mary (mary brown) or <empty>	Exprodo or <other plugin name> or <empty>	pGina	OK

☒ Computer Monitoring Every minutes

Processes to monitor	
<input checked="" type="checkbox"/>	notepad
<input type="checkbox"/>	*

Notification with processes notifies the **Calpendo** server with the information sent for a **Simple Notification**, but also includes extra information about the running processes. Only those processes listed in **Processes to monitor** will be included in this service.

All information related to monitored processes will be stored in the **Remote Process** biskit.

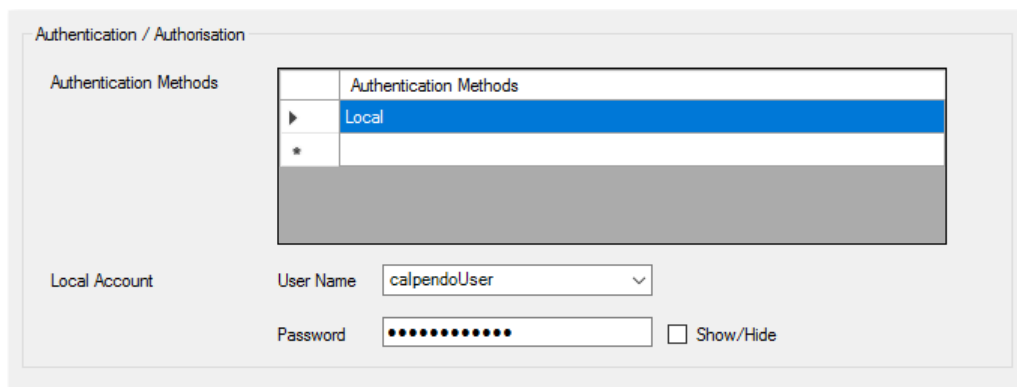
Name	User Log	Start	Process ID	Remote User
notepad	Remote User Log #1612	30 Nov 2016 10:08	457	mary
powershell	Remote User Log #1612	30 Nov 2016 10:08	643	mary

Both notifications run with the interval specified in the **Exprodo - Plugin Setting** screen. E.g. If the **Computer monitoring** time is set to 30 Minutes, as in the screen above, then every 30 minutes, a check will be made to see if the relevant applications are running e.g. **Notepad**, and if it is, a new line will be added to the relevant **Calpendo** database, which will log the application, time of check and the username of the user currently logged into the machine.

Authentication

This service authenticates users using the **Calpendo** server. This means that any **Calpendo** user can login to the local machine using their **Calpendo** account at the **pGina** user login screen. Note that this only works for those authentication methods where **Calpendo** expects to be handling the user's password and cannot work when users are authenticated outside **Calpendo**, such as single-sign-on.

Windows, however, doesn't let any user login in without having a local account. So, once the **Authentication** service has authenticated a **Calpendo** user, a local user account (supplied in the **Exprodo** plugin settings screen) is then automatically used to log in to the local machine.



This additional implementation will not affect either **Calpendo** server data consistency nor correctness of local machine behaviour.

The **Authentication** service always sends the user's original name to the **Calpendo** server, and not the local **Windows** user name, to guarantee data consistency. This also provides additional support to the local machine, in order to manage **Calpendo** users.

Because all **Calpendo** users are converted into the same local username, the local machine doesn't know who the real user is performing the action. For this reason, the **Exprodo** plugin will supervise all "lock" and "switch" actions (using **Notification** service) in case they are executed from a **Calpendo** user. In particular, it will execute a double check to be sure that only the corresponding original user could unlock the machine (analogous behaviour for switch action).

For Windows XP Only

When **Windows XP** locks the screen, it doesn't go through **pGina**. Consequently, the **Exprodo** plugin cannot supervise the action as previously described.

This presents a problem because the user credentials seen by **Windows** are different from the credentials the user supplied at login; the **pGina** plugin uses the same **Windows** user regardless of who logs in, with only **pGina** and **Calpendo** knowing the actual user involved.

Consequently, if the screen were locked, the password required would not be the password the user entered to log in initially, but the password **Windows** sees for the actual underlying **Windows** user.

This can be got around by either sharing this password amongst all users, having a **Windows** administrator unlock the screen or else by preventing **Windows XP** from ever locking the screen.

The most convenient of these is to prevent the screen from being locked.

Preventing Windows XP Professional from Locking The Screen

1. Open **Group Policy Editor** (from 'Run' > **gpedit.msc**)
2. Go to: **User Configuration -> Administrative Templates > System > Ctrl+Alt+Del Options(*)**
3. Double-Click on **RemoveLockComputer** and Select "**Enabled**"
4. Click on **OK** to confirm selection.

(*) If "Administrative Templates" doesn't contain "System" you can add it with these simple steps:

- Right-Click on "**Administrative Templates**" and click on "**Add/Remove Templates**"
- Select "**System**" and click on the "**Add..**" button
 - ? A "**Policy Template**" windows will be opened
 - ? Select "**system.adm**" and click the "**Open**" button

Now you can continue from Step2 above

Furthermore, as **pGina** allow the use of multiple authentication plugins, **Exprodo** plugin does not make any change to original login credentials, so that any other plugin can work with the correct original login values.

Every time the plugin tries to authenticate a user, the operation will be recorded in the **Remote User Log** *biskit* (see table below).

Type	Created	Remote Time	Login Name	Domain Name	IP Address	Accepted
AUTHENTICATION	16 Nov 2016 15:52	16 Nov 2016 15:52	john	DESKTOP-ONE	150.150.10.10	<True> or <False>

User	Remote Authentication Method	Remote Activity Recorder	Comments
john (john smith) or <empty>	Exprodo or <empty>	pGina	OK Failed - User john can not be authenticated in Calpendo with given authentication methods Failed - User john can not be identified in Calpendo with given authentication methods

Authentication (Calpendo / Exprodo Server Side)

The **Calpendo** server authentication is made up of two phases:

1. Identification;
2. Authentication.

In Identification, the **Calpendo** server tries to retrieve the **Calpendo User** record from the provided login name and authentication methods list. This can be customised by setting up a workflow and using a "[Remote User Identification Request Workflow Event](#)³⁴⁰". This is triggered by the identification requests, and allows the administrator to choose how to perform this action. p

In **Authentication**, the **Calpendo** server tries to authenticate the identified **Calpendo User** with the password provided by the **Exprodo** plugin.

Authorisation

This service provides users with the authorisation to login to the local computer by asking the **Calpendo** server whether it should be allowed. This allows an opportunity to check things like whether you are trained to use an instrument associated with this local computer. This is done using a **Calpendo** server workflow that uses "[Remote User Identification Request Workflow Event](#)³⁴⁰" and/or "[Remote Authorisation Request Workflow Event](#)³⁴⁰". By default, authorisation requests succeed without a workflow whenever a user is correctly identified.

Note that authentication can be performed using another service, such as **LDAP**, and still go through the **Calpendo** authorisation process; it is not a requirement that all users pass through **Calpendo** authentication, only that at least one **pGina** plugin authenticates the user.

If the authorisation fails, then login to the local computer will be interrupted with a suitable message.

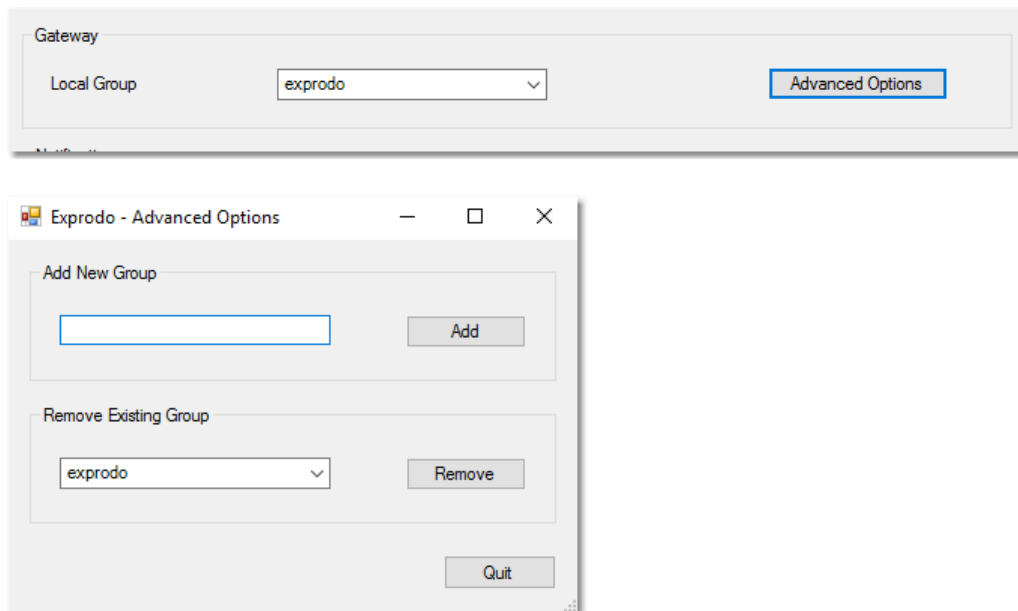
Regardless of whether authorisation succeeds, the operation will be recorded in the "**Remote User Log**" biskit (see table below).

Type	Created	Remote Time	Login Name	Domain Name	IP Address	Accepted
AUTHORISATION	16 Nov 2016 15:53	16 Nov 2016 15:53	mary	DESKTOP-ONE	150.150.10.10	<True> or <False>

User	Remote Authentication Method	Remote Activity Recorder	Comments
mary (mary brown) or <empty>	Exprodo or <other plugin name>	pGina	Ok - User authenticated by Calpendo Ok - User authenticated by Local Machine Failed - Can't find corresponding Calpendo User - mary has been authenticated by Local Machine

Gateway

This service will add users, authorised by **Calpendo**, into the group supplied in the **Plugin Settings** screen.



This does not affect the login procedure, but provides a means by which a computer administrator can ensure that only **Calpendo**-authorised users can run certain applications. This is possible by setting **Windows** security to require that a user is a member of the appropriate local user group in order to run the software you wish to control.

This allows the computer to be used for general purposes, while only allowing an instrument attached to the computer to be used when a user has been authorised to do so.

If for any reason the **Exprodo** plugin cannot perform this action, a notification to the **Calpendo** server will be sent, with a message explaining the kind of problem but the Login process will not stopped. However, the logged-in user would not be a member of the relevant local user group and so would not be able to run any application that required membership of said group.

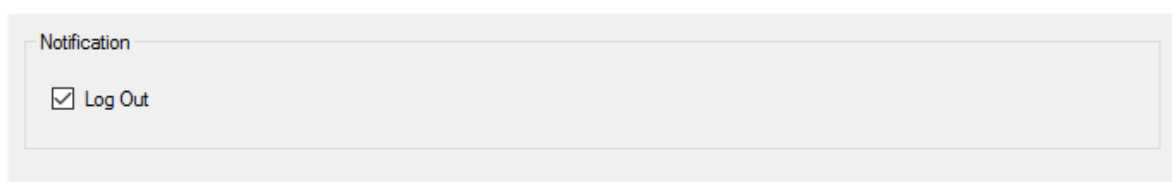
Type	Created	Remote Time	Login Name	Domain Name	IP Address	Accepted
GROUP	16 Nov 2016 16:13	16 Nov 2016 16:13	mary	DESKTOP-ONE	150.150.10.10	<True> or <False>

User	Remote Authentication Method	Remote Activity Recorder	Comments
mary (mary brown)	Exprodo or <other plugin name>	pGina	<p>Error - I cannot extract user and group name from Exprodo plugin</p> <p>Error - I cannot add user "CalpendoUser" to "Calpendo" group - Local Calpendo user doesn't exist!</p> <p>Error - I cannot add user "CalpendoUser" to "Calpendo" group - Local Calpendo group doesn't exist!</p> <p>Error - I cannot add user "CalpendoUser" to "Calpendo" group</p>

Notification

This service allows the **Calpendo** server:

- To get a notification about a **Logout** action
- To allow the **Authentication service** to know who is really using the computer when "lock" and "switch" actions are performed by users authenticated by **Calpendo** (see **Authentication Service**).
- To allow the **Ping** service to get all information about current user.



Notification

☒ Log Out

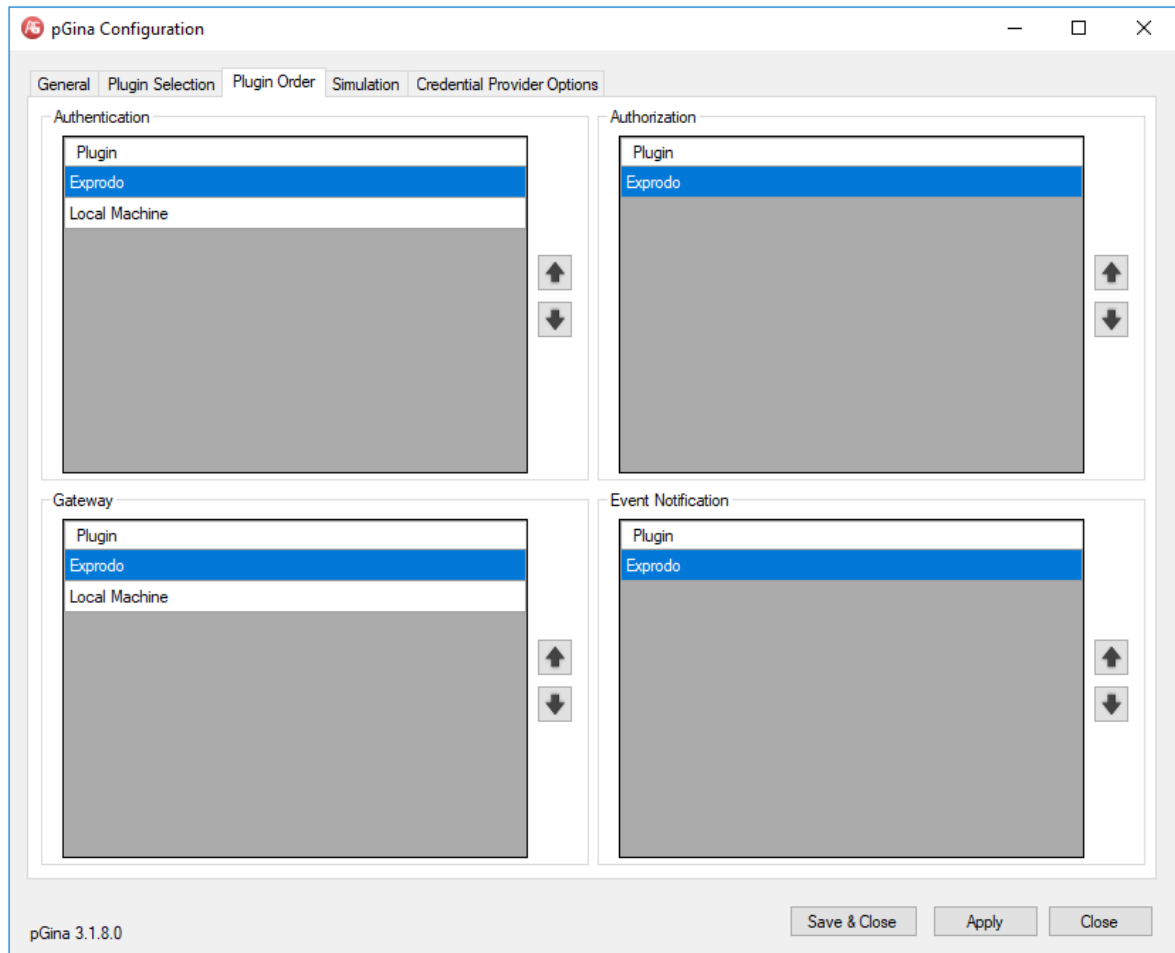
"**Logout**" runs every time a user authenticated and/or authorised by **Calpendo** performs a logout action.

Type	Created	Remote Time	Login Name	Domain Name	IP Address	Accepted
LOGOUT	16 Nov 2016 16:13	16 Nov 2016 16:13	mary	DESKTOP-ONE	150.150.10.10	<True> or <False>

User	Remote Authentication Method	Remote Activity Recorder	Comments
mary (mary brown)	Exprodo or <other plugin name>	pGina	Ok

Plugin Order Tab

To ensure the authentication to **Calpendo** and **Windows** is completed successfully, you must check that **Exprodo** appears at the top of the lists, in the **Plugin Order Tab**. You can do this by clicking on **Exprodo** and then using the arrows, to right hand side of each box, to change the order.



6.12 Permissions

[Permissions](#)⁶⁵⁴ are designed to be very flexible, so that you have fine-grained control over who can perform what action on what data. This can be done by creating multiple *Permissions* objects. **Calpendo** looks at all the *Permissions* that apply to the action the user is trying to perform and uses them to work out if Permission should be granted or denied.

6.12.1 How Permissions Work

Some systems have a large master list of actions, and then for each role, there would be a tick against each of the actions that users with that role can do.

The problem with that kind of approach is that it is best employed when the system has a predefined set of actions that need to be controlled. However, **Calpendo** is designed to allow for precise control over the detail of what people are allowed to do. This means that it is much more flexible for a variety of situations. That flexibility also leads to a degree of complexity, but this section of the manual will try to make this topic as simple as possible.

[Permissions](#)⁶⁵⁴ are not implemented with an all-encompassing list of actions. The method that is used is to define a small set of simple [actions](#)⁶⁵⁴, and then create customised and highly targeted individual *Permissions* to control what can be done. The overall picture is then obtained by aggregating those individual *Permissions*.

That aggregation is possible due to a mechanism that allows layering of *Permissions*, where some *Permissions* can override others. The crucial element in allowing this is the concept that **Calpendo** supports both authorising *Permissions* that permit something to be done, and non-authorising *Permissions* that deny the right to do something.

Breaking An Activity Into Its Parts

Let's start by looking at an example, viewing the content of a [booking](#)⁶⁵².

This breaks down into the following items:

1. **Action**

Start by looking at the underlying *action*. Everything is broken down into an *action* of some sort, and the most common *actions* are create, read, update and delete. In the example, it is view, or read.

2. **[Biskit Type](#)**⁶⁵²

Next, look at the type of the data the *action* will affect. In the example in order to control viewing of the content of a *booking*, the *Biskit Type* is *booking*.

3. **[Property](#)**⁶⁵⁵

When trying to read or update something, as in the example, then the *action* has an impact on individual *properties*. That means the user may have *permission* to read some of the *properties* on the *booking*, but not all of them.

So for some *Permissions*, it is necessary to define the *property* controlled by the *Permission*. For example, to grant or deny the ability to view a particular *property* on a *booking*, create a *Permission* that affects just that one *property*.

Permissions that apply to individual *properties* are only relevant to reading and updating things. For example, when deleting something, the whole thing is deleted and not just individual *properties*. So it makes no sense to specify a *property* on a *Permission* that controls who can delete data.

Layering Permissions

When defining *Permissions*, specify the *action* to be controlled and optionally the *Biskit Type* and *property*. *Permissions* that specify only the *action* are known as *action Permissions*. *Permissions* that specify the *action* and a *Biskit Type* are known as *Biskit Type Permissions*, and *Permissions* that specify *action*, *Biskit Type* and *property* are known as *property Permissions*.

In a typical system, everybody can read most things. So it is usually easier to start by giving read *Permission* to everybody to read everything. This can be done with an *action Permission* for the read action. Then, layered on top of that, create a non-authorising *Permission* that takes away *permission* for the things that are not allowed to be read.

The layering works by using *action Permissions* only if there isn't a relevant *Biskit Type* or *property Permission*. Similarly, *Biskit Type Permissions* are used if there isn't a relevant *property Permission*.

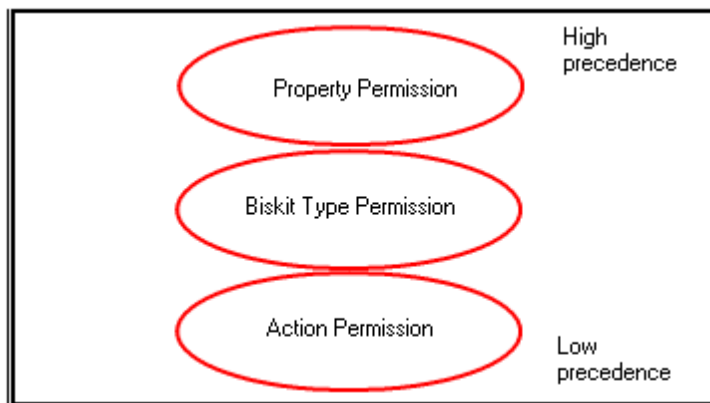
There is also a numerical priority that can be specified for each *Permission*, and that provides another way to layer *Permissions* because the highest priority *Permissions* within each of the three layers, *action*, *Biskit Type* and *property*, are considered first.

If a relevant *Permission* can not be found, then permission to perform the *action* is denied.

Precedence Of Permissions

When sorting through the *Permissions* to decide which one to apply, **Calpendo** will only consider those that match the *action*, *Biskit Type*, *properties*, [conditions](#)⁶⁵³ and that also target the user concerned. Also, note that those *Permissions* that do not specify a *Biskit Type* will also be considered.

Once that filtering has taken place, **Calpendo** will use *property Permissions* first, then *Biskit Type Permissions*, and finally *action Permissions*. Within each of those three, *Permissions* are further sorted by priority, and the *Permission* with the highest numerical value for its priority will be used. If there are multiple *Permissions* with the same priority, then it is undefined which one will be used. The reason this is required will become apparent after going through the way in which the *conditions* under which each *Permission* may be used can be specified.



Precedence of permission types.

When a *BiskitDef* has another *BiskitDef* as its parent it will have access to the *Permissions* created for the parent.

When checking *Permissions* all the child *Permissions* will be filtered and acted on. If there is not a relevant *Permission* then the parents *Permissions* will be filtered and acted on, for each type of *Permission* in order of precedence. For the action **Update** this will be done first using *property Permissions*, and then *BiskitDef Permissions* and finally the action **Update Permissions**. This means all *property Permissions* on the parent will be checked before any *BiskitDef Permissions* on the child will be checked.

Conditions

The *conditions* assigned to a *Permission* provide a means of very precise control over the situations in which a *Permission* will apply. For example, there may be different *Permissions* that apply depending on:

- the *resource* a *booking* is made for
- the *status*⁶⁵³ of a *booking* (for example, a *booking* may be editable while it is a request, but not once it has been approved)

or a particular *Permission* is to apply when:

- approving a booking
- cancelling a booking

All of these things are specified by the *conditions*. Here are some examples. Suppose there are two *Permissions*, both of which apply to updating *bookings*, and both of which are *Biskit Type Permissions* (which means they specify the *action* and *Biskit Type* but not a *property*). *Permission A* has the *condition*:

- *status* equals **Requested**

and *Permission B* has the *condition*:

- *resource* equals **Wet Lab**

In this case, assuming both *Permissions* apply to the current user, then *Permission A* will apply whenever the user tries to update a *requested booking*⁶⁵⁵ and *Permission B* will apply whenever the user tries to update a *booking* for the **Wet Lab** resource.

What happens when trying to update a *requested booking* for the **Wet Lab**. In that case, both *Permissions* could apply. This is where the *Permission* priority comes in. If one *Permission* has a higher priority than the other, then the highest priority *Permission* will be used. If they have the same priority, then either one could be used.

Authorising And Non-authorising Permissions

When thinking about giving *permission* for something, one normally thinks in positive terms. That is, authorising *permission* for something. However, the layering used in **Calpendo** requires that there is also the notion of non-authorising *permissions* as well as authorising ones. That means that a *Permission* can be written that will grant the right to do something at one layer, and then have another *Permission* at another layer that will explicitly deny the right to do something.

For example, there may be an authorising action *Permission* that allows everybody to read everything, and then a non-authorising *property Permission* to deny the ability to read the *booking* price for some people.

Targeting Users

For each *Permission*, choose which users it will affect.

However, it's sometimes easier to say who a *Permission* shouldn't affect, or to say that it's everybody meeting some criteria, apart from those that meet some other criteria.

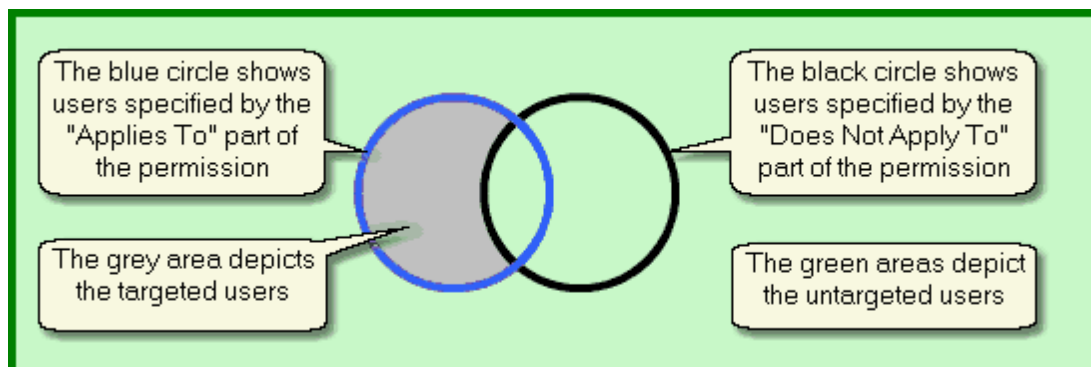
So **Exprodo** applications allows the specification of the users a *Permission* applies to⁶⁵² as well as those it does not apply to⁶⁵³. The affected users will be everybody the *Permission* applies to that are not amongst those it *does not apply to*.

Within each of the *applies to* and *does not apply to* sections, identify users by employing one or more of the following methods:

- Name individual users or user groups⁶⁵⁶.
- Specify users by the roles⁶⁵⁶ they have.
- Specify a property on the object being controlled that will identify a user or *user group*.

This last item, specifying a *property* that identifies a user or *user group*, needs more explanation. For example, a Calpendo *booking* has an owner⁶⁵⁴ that is the user that owns the *booking*, a booker⁶⁵² that is the user that created the *booking* and a *project*. Each *project* also has an *owner* and users.

Consequently, when writing a *Permission* that affects a *booking*, users can be targeted by using any of the *properties* *owner*, *booker*, **project.owner** and **project.users**.



Showing the targeted and untargeted users, defined by the "Applies To" and "Does Not Apply To" users.

Available Actions

The following table shows the available actions, and indicates whether each action uses *action*, *Biskit Type* or *property Permissions*:

Action	Applies When You...	Uses Action Permissions	Uses Biskit Type Permissions	Uses Property Permissions
Create	Create a new object	X	X	X
Read	View an object	X	X	X
Update	Update an object	X	X	X
Delete	Delete an object	X	X	
Exists	Display a list of objects	X	X	
Email Readable	Include data in an email	X	X	X
Run Report	Run a report	X	X	
Execute	Run ExecuteSystemCommand Action Layout Running the Booking Rule Validator ⁶⁵³	X	X	
Manage Add-ons	Which users can access the add-on feature.	X		
Checked Edit	Edit a list of Biskits using Checked Edit	X		
Dump Database	Take a backup of the whole database	X		
Update Database Schema	Add or remove tables or columns in the database	X		

Create

When a new object is created, action and *Biskit Type Permissions* for the **Create** action are checked. The *property Permission* is checked to see if the *property* should be rendered read or read/write only by comparing against the default value of the *property*.

Read, Exists and Email Readable

When a user tries to view an object, **Exprodo** applications check whether the user has:

- **Exists permission** on the object to check whether the user is allowed to know that the object exists
- **Read permission** on each *property*, which means that *property* checks are performed on every *property* on the object.

The value of any *property* on which the user does not have **Read permission** will be hidden from the user.

When displaying a list of objects, **Exists** *permission* will be checked so that only those objects that the user is allowed to know exist will be seen. This also means that when generating a report that counts the number of objects matching some criteria, then the count will not include objects for which the user does not have **Exists** *permission*.

When sending an email, each object *property* that is to be included in the text is checked for both **Read** *permission* and **Email Readable** *permission*. Also, for every *property* inserted into the email, the object the *property* comes from is checked for **Exists** *permission*.

Emails may be read by anybody, and so when **Exprodo** applications check *permissions* for what can be included in an email, it checks whether the special user [nobody](#)⁶⁵⁴ has the relevant *permission* as well as the user that performed the action that caused an email to be sent. Therefore it is necessary to create a *Permission* that gives **Email Readable** *permissions* to user *nobody* for any information that is required to go into an email. See [Special Users](#)¹⁸¹ for more details on user *nobody*.

Update

When a user tries to update an object, **Exprodo** applications will check for **Update** *permission* on the object and also on each *property*. If *permission* is denied for any *property*, then **Update** *permission* for the whole object is denied. This is different from viewing an object, where being denied *permission* to read one *property* does not stop the user from seeing the other *properties*.

When editing an object, if **Exprodo** applications can determine that *Permissions* prevent the user from changing a *property*, then that *property* will be rendered in an unmodifiable form. This will prevent the user from making changes that would only be rejected later.

Delete

When deleting an object, **Delete** *action* and *Biskit Type Permissions* are checked.

Run Report

Run Report *action Permissions* are checked whenever a report is run. This allows the administrator to prevent some types of report being run (for example, to stop some people from running summary or group reports), or the *Biskit Type* being reported on (for example, to restrict reports about system usage or other things).

To control which reports somebody can run, create a **Run Report** *action Permission* for the *Biskit Type Report*. Then place conditions on which reports can be run. If most users are set up not to be able to run a report then make sure the user **nobody** is in the Does Not Apply To section in order that scheduled reports will run.

Run Report *Permissions* for any *Biskit Type* other than **Report** will be ignored.

Execute

There is a workflow *Biskit* of type **System Command** which is used to specify which system commands can be run. The administrator must set up an **Execute** *permission* on this *Biskit*, **Data Property** to **None**, **Authorisation** to **Grant Permission**, and specify in the **Conditions** which commands are allowed. eg. Value of command equals groff. They can then specify who it applies to etc.

The **Execute** permissions can be applied to the **Layout Biskit** to define which **Layout** individuals can see, rather than using **Exists** which also causes problems when editing the **Layout**, as if it doesn't exist for **Admin** role they cannot see it to edit it.

This is also checked when trying to run the [Booking Rule Validator](#)⁶⁵³ so that only permitted users may test Booking Rules. This is important because an [Advanced Booking Rule](#)²⁶⁹ contains user-supplied Java code that will be executed on the server. To cover the potential security issue, users require **Execute permission** on the [Booking Rule](#)⁶⁵².

To allow a user to validate *Booking Rules*, there must be a *Permission* for the **Rule Biskit Type** that must have no *conditions*. A future version of **Calpendo** will allow the administrator to control which *Booking Rules* a user can validate. Currently, users can either validate all *Booking Rules* or no *Booking Rules*.

Checked Edit

This defines who can use the **Checked Edit** feature of **List Views**.

Manage Add-ons

This defines who can use the Add-ons feature

Dump Database

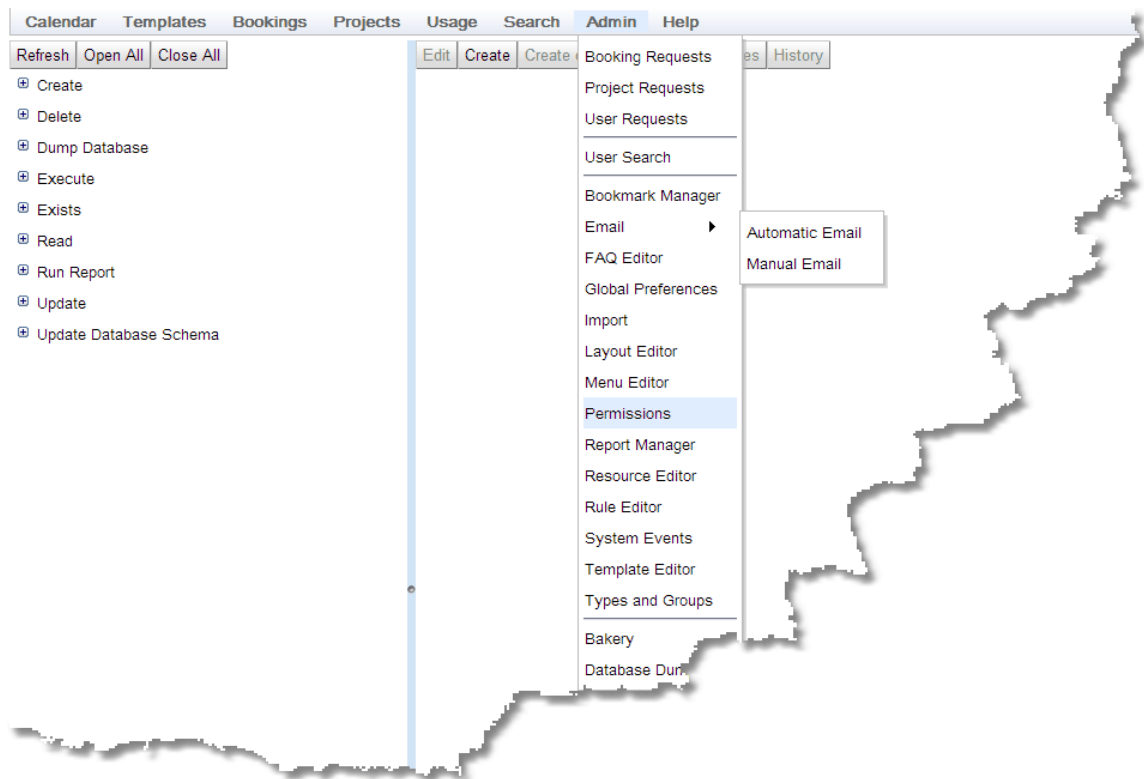
Dump Database action *Permissions* are checked when trying to using the **Database Backup** page to generate a copy of all the data in the database.

Update Database Schema

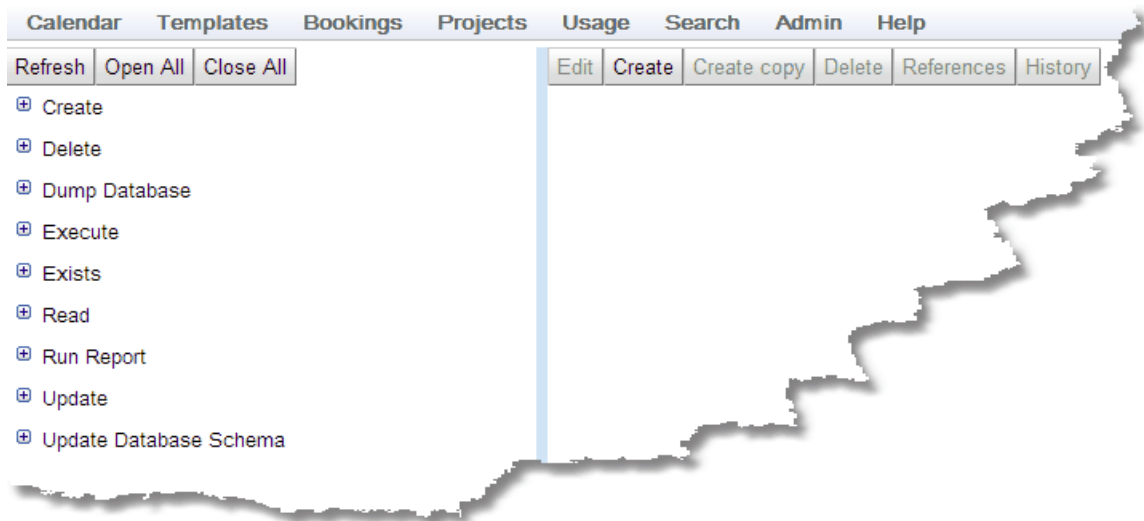
Checked when using the [Bakery](#)⁶⁵² to modify the database schema.

6.12.2 The Permissions Editor

The **Permissions Editor** shows all the [Permissions](#)⁶⁵⁴ and allows the user to create, update and delete them. By default, it appears on the menu here:



This is what the **Permissions Editor** looks like first opened:

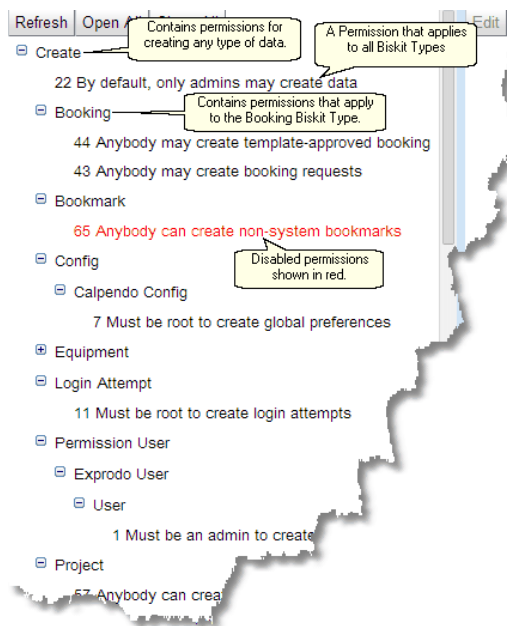


The Tree Of Permissions

Permissions apply to a particular [action](#)⁶⁵⁴ and [Biskit Type](#)⁶⁵². The above screen shot shows all the *Biskit Types* for which there are *Permissions*. Note that a *Permission* can also be defined that will apply to any *Biskit Type*.

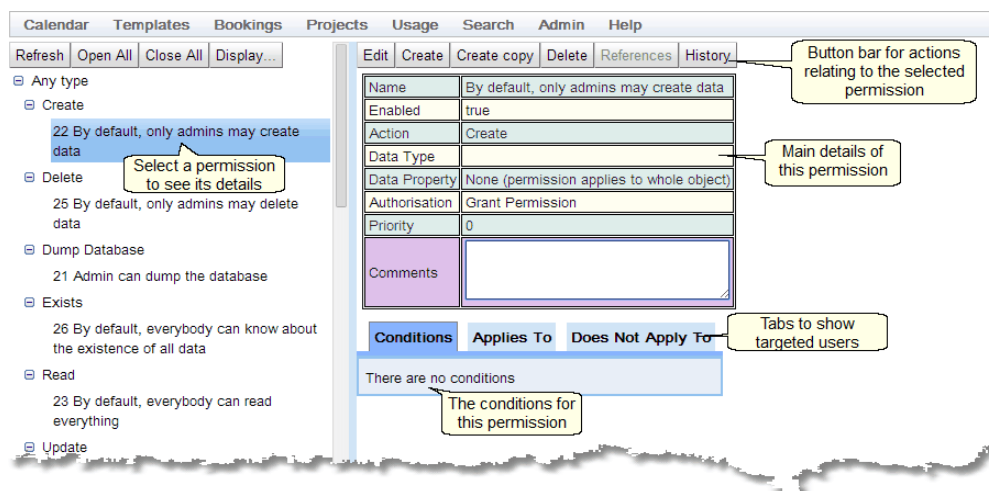
For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Once all the items in the tree are viewable, the *Permissions* are viewable in their *Action*, *Biskit Type*, *Permission* categorisation with any that are disabled shown in red.

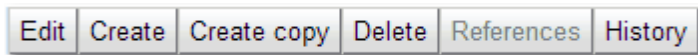


Permission Detail

Click on a *Permission* in the tree, and the details for that *Permission* will appear on the right:

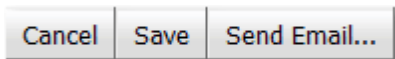


Now that a *Permission* is selected, the button bar is no longer greyed out:



For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Press the **Edit** button to make the page to be editable, and the button bar changes:



The main details of the *Permission* contain all the information represented in the tree on the left. These two screen shots show the details in read-only and in edit mode:

Name	By default, only admins may create data
Enabled	true
Action	Create
Data Type	
Data Property	None (permission applies to whole object)
Authorisation	Grant Permission
Priority	0
Comments	

Name	By default, only admins may c
Enabled	true ▼
Action	Create ▼
Data Type	Please select a Biskit Def
Data Property	Data type not set ▼
Authorisation	Grant Permission ▼
Priority	0
Comments	

When changing these details:

- The **Name** can be as required, but should not be the same as any other *Permission*.
- The **Comments** can also contain anything and is only there to record whatever notes there are about the *Permission*.
- The **Enabled** flag is a way to create a *Permission* without making it take effect immediately, or to turn one off without deleting it. A disabled *Permission* shows in red in the tree.

The other [properties](#)⁶⁵⁵ are described in [How Permissions Work](#)³¹⁴.

The Conditions Tab

[Conditions](#)⁶⁵³ underpin much of the configuration that can be done in **Calpendo**, and they are used in several places (most notably in [Workflows](#)³³⁴). Consequently, they have their own section of this configuration guide. See [Conditions](#)¹⁰⁷ for the details.

Please note that the section that describes *conditions* indicates that the **Updated Value** and **Change condition** types are considered an error in some contexts. For *Permissions*, they are only usable when the *Permission action* is **Update**.

The Applies To Tab

In read-only and editable mode, the **Applies To** tab looks like this:

The image shows two side-by-side screenshots of the 'Applies To' configuration tab. Both screenshots have the 'Include Everybody' checkbox at the top, which is unchecked. Below it are four sections: 'Individual Users', 'User Groups', 'Property Path to User(s) or User Group(s)', and 'Roles'. In the 'Individual Users' section, there is a dropdown menu with the text 'Please select a User to add' and 'Selection empty' below it. In the 'User Groups' section, there is a dropdown menu with the text 'Please select a User Group to add' and 'Selection empty' below it. In the 'Property Path to User(s) or User Group(s)' section, there is a red box with the text 'Data type not selected' and 'Selection empty' below it. In the 'Roles' section, there is a checkbox for 'Use Roles' which is checked, a dropdown menu with the text 'Require all selected roles', and four checkboxes for 'Root', 'Admin', 'User', and 'Guest'. In the left screenshot, 'Admin' is checked. In the right screenshot, 'Admin' is checked and 'User' is also checked.

Include Everybody

If set to **true**, then this means that the *Permission* applies to all users, and so the rest of the tab is hidden. As described in [Introduction To Permissions](#)³¹⁴, the targeted users are those users covered by the **Applies To** tab that are not also covered by the **Does Not Apply To** tab.

Individual Users

This allows the specification of particular users that the *Permission* should apply to.

User Groups

The *Permission* will apply to any user in the specified [groups](#)⁶⁵⁶.

Property Path

A *Permission* [property path](#)⁶⁵⁵ works exactly the same as the property path in an [Email Workflow Action](#)³⁷³, and it is described more fully in that section. The *property path* is a *property* or list of *properties* that lead from the *Biskit Type* defined on the *Permission* to a user or *user group*, and those users will also have the *Permission* applied to them.

User Roles

Users can have multiple [roles](#)⁶⁵⁶. Any number of *roles* can be selected by ticking the check box next to the *role*, and then choose with the drop-down whether the *Permission* is to apply to users that have *all* of the selected *roles*, or users that have any of the selected *roles*.

The Does Not Apply To Tab

This tab is almost identical to the **Applies To** tab. The only difference is that the check box at the start is labelled **Exclude Nobody** instead of **Include Everybody**. When ticked, it means that no users are taken away from the [Applies To](#)⁶⁵² list of users, and so the rest of the **Does Not Apply To** tab will be hidden. When unticked, the users that should not be targeted are selected.

6.12.3 Example Permissions

Example 1: Anybody May Create A Booking Request

This example will create a [Permission](#)⁶⁵⁴ that authorises anybody to create a *booking* when the *booking* to be created has its status set to **Requested**. Note that by creating a *Permission* like this, it doesn't influence what happens when somebody tries to create a *booking* whose *status* is set to something other than **Requested**.

1. Create a *Permission* that applies when a *booking* is **Created** and give it a name

The screenshot shows a form for creating a permission. Annotations point to various fields:

- Name:** "Anybody may create a booking request" (Annotation: "Give the permission a name")
- Enabled:** "true" (Annotation: "Leave the permission enabled")
- Action:** "Create" (Annotation: "Set the action to Create")
- Data Type:** "Booking" (Annotation: "Set the BiskitDef to Booking")
- Data Property:** "None (permission applies to whole object)" (Annotation: "Leave the data property set to none")
- Authorisation:** "Grant Permission" (Annotation: "Make this a grant authorisation so targeted users are granted permission")
- Priority:** "0" (Annotation: "Use the default priority of zero")
- Comments:** (Empty field)

2. Add a [condition](#)⁶⁵³: **Status is Requested**

The screenshot shows a condition configuration bar with the following values:

- Value: (dropdown)
- of: status
- equals: (dropdown)
- Specified value: (dropdown)
- Requested: (dropdown)

3. Under [Applies To](#)⁶⁵², ensure **Include Everybody** is ticked

The screenshot shows the **Applies To** tab with the **Include Everybody** checkbox checked.

4. Under [Does Not Apply To](#)⁶⁵³, ensure **Exclude Nobody** is ticked

The screenshot shows the **Does Not Apply To** tab with the **Exclude Nobody** checkbox checked.

5. Save the *Permission*.

Example 2: Anybody Can Modify A Booking Request They Created If The Booking Is Still A Request

This example shows the difference between using a **Old Value** condition and an **New Value** condition. We need a *Permission* that will allow somebody to modify a *booking* if they created it and the status was set to **Requested** before they tried to change it, and its value would still be **Requested** after the change.

1. Create a *Permission* that applies when a *booking* is **Updated** and give it a name:

Name	Anybody modify their own booking request
Enabled	true
Action	Update
Data Type	Booking
Data Property	None (permission applies to whole object)
Authorisation	Grant Permission
Priority	0
Comments	

2. Add a *condition*: **Status** is **Requested**. Note that for an update, to put a *condition* on the value after the update, then use a **New Value** type of *Condition*.

New value	of	status	equals	Specified value	Requested
-----------	----	--------	--------	-----------------	-----------

3. Add a *condition*: **Status** was **Requested**. Here, we use a **Old Value** type of *Condition* to put a *condition* on the value that the status had before the update.

Old value	of	status	equals	Specified value	Requested
-----------	----	--------	--------	-----------------	-----------

4. We now need to limit the *Permission* to applying to whomever created the *booking*. This is easy because each *booking* stores a record of who booked it in the *booker*⁶⁵² property. To do this, go to the **Applies To** tab, untick **Include Everybody** and then add a *property* path to *booker*.

Conditions	Applies To	Does Not Apply To
<input type="checkbox"/> Include Everybody		
Individual Users Please select a User to add Selection empty		
User Groups Please select a User Group to add Selection empty		
Property Path to User(s) or User Group(s) booker		
<input type="checkbox"/> Property Path <input type="checkbox"/> booker		
Roles <input type="checkbox"/> Use Roles		

5. Save the *Permission*.

Example 3: Admins May Create Or Update Anything

This example shows how to use User Roles to decide who gets *permission* to do something, as well as showing how to create a *Permission* that applies to every [Biskit Type](#)⁶⁵². Use a *Permission* like this when the Admin Role needs to bestow the ability to create or update whatever they like. The *Permission* can still be revoked in some circumstances by using a *Permission* of a higher level elsewhere that overrides the *Permission* we're creating here.

To achieve the ability to let the administrator create or update anything, we need to create two separate *Permissions*, one for the **Create** and one for the **Update**.

1. Create a *Permission* that applies when **Anything** is **Created** and give it a suitable name. Leave all the other options at their default values:

Name	Admins can create anything
Enabled	true ▼
Action	Create ▼
Data Type	Please select a Biskit Def
Data Property	Data type not set ▼
Authorisation	Grant Permission ▼
Priority	0
Comments	

2. Under **Applies To**, untick **Include Everybody**, select **Use Roles** and tick the check box next to **Admin**

The screenshot shows the 'Applies To' tab with the following configuration:

- ☐ Include Everybody
- ☐ Individual Users
- ☐ User Groups
- ☐ Property Path to User(s) or User Group(s)
- ☒ Roles
 - ☒ Use Roles
 - ☐ Require all selected roles ▼
 - ☐ Root
 - ☒ Admin
 - ☐ User
 - ☐ Guest

3. Save the *Permission*.
4. Repeat step 1, but this time set the [Action](#)⁶⁵⁴ to **Update** and set the *Permission*'s name accordingly.
5. Repeat of steps 2 and 3 above for this **Update Permission**.

Example 4: Only Allow Specified Users To See The Price Being Charged For Use Of A Resource

This example protects the **Cost Per Session** property of a *Project Resource Setting*, as stored in each *project*. The idea is that we would like to stop anybody apart from one particular administrator being able to see what the price being charged is. This serves to demonstrate how to protect individual *properties* and how to apply Permissions to individuals. This is known as a [Property Level Permission](#)^{§15}.

There are two ways we can set up *Permissions* to achieve the result we want, depending on what we'd like to achieve. We can create a *Permission* for each methodology:

- Authorise the user **Admin** to read the the **Cost Per Session**, but say nothing about whether other users can read it.
- Deny everyone other than the user **Admin** from being able to read the **Cost Per Session**, but say nothing about whether **Admin** can read it.

When a *Permission* says nothing about whether a user is granted or denied *permission*, then you rely on other *Permissions* to specify what you want. This is the way that multiple *Permissions* are superimpose to get the result required. To show how this works, we'll go through each of the options above in turn.

1. Create a *Permission* that applies when a *Project Resource Setting* is **Read** and set the **Data Property** to **Cost Per Session**

Name	Only admin can see prices
Enabled	true ▼
Action	Read ▼
Data Type	Project Resource Settings
Data Property	Cost per Session ▼
Authorisation	Grant Permission ▼
Priority	0
Comments	

2. On the **Applies To** tab, untick **Include Everybody** and under **Individual Users**, select one or more users. In this example, a user called **Admin** has been added.

The screenshot shows the 'Applies To' tab with the following configuration:

- Conditions:** (Not shown)
- Applies To:**
 - ☐ Include Everybody
 - Individual Users:**

<input type="checkbox"/>	Login name	Given name	Other name	Family name
<input checked="" type="checkbox"/>	admin	admin		

The callout box shows a detailed view of the 'Individual Users' table:

<input type="checkbox"/>	Login name	Given name	Other name	Family name
<input checked="" type="checkbox"/>	admin	admin		

3. Save the *Permission* as it is, and with the default values of **Authorisation** being **Grant Permission**, it means that the targeted users (in this case, the user **Admin**) will be granted *permission* when trying to view the **Cost Per Session**. This *Permission* says nothing about whether other users can read **Cost Per Session**.

If you now wanted to change the *Permission* so that it says nothing about whether **Admin** can read **Cost Per Session**, but that it denied *permission* for all other users, follow the following steps:

4. Press the **Edit** button (unless you didn't actually save it and so it's still shown in an editable form).
5. On the **AppliesTo** tab, tick **Include Everybody**.
6. On the **Doesn't Apply To** tab, untick **Exclude Nobody** and add **Admin** to the list of individual users.
7. In the main *Permission* details, change **Authorisation** to **Deny Permission**.

What we've now done is to target everybody apart from **Admin**. We've made the *Permission* negative, so that the targeted users will be denied permission to read **Cost Per Session**.

Example 5: Approval Of A Booking Requires An Admin

In this example, we are going to create a *Permission* that allows anybody with the Admin Role to change a *booking's status*⁶⁵³ to **Approved**. We will also add a *condition* to make sure we're matching the situation where an administrator modifies the *status* themselves, rather than where the *status* is modified indirectly as a result of moving the *booking* to a period in which a *Time Template*⁶⁵⁶ has approved the *booking*.

1. Create a *Permission* that applies when a *booking* is **Updated**, give it a name, and make it **Grant Permission**.

Name	Booking approval requires an A
Enabled	true ▼
Action	Update ▼
Data Type	Please select a Biskit Def
Data Property	Data type not set ▼
Authorisation	Grant Permission ▼
Priority	0
Comments	

2. Add a *condition*: **Old value Status** was not **Approved**

Old value ▼	of	status	not equal to ▼	Specified value ▼	Approved ▼	
-------------	----	--------	----------------	-------------------	------------	--

3. Add a *condition*: **New value Status** is **Approved**

New value ▼	of	status	equals ▼	Specified value ▼	Approved ▼	
-------------	----	--------	----------	-------------------	------------	--

4. Add a *condition*: **New value templateApproved** is **false**. The **templateApproved** *property* is set automatically when a *booking* is approved by a *Time Template* and so we can use it to detect how the status was changed.

New value ▼	of	templateApproved	equals ▼	Specified value ▼	false ▼	
-------------	----	------------------	----------	-------------------	---------	--

5. Under **Applies To**, untick **Include Everybody**, select **Use Roles** and then add **Admin**.
6. Save the *Permission*.

Example 6: No One Can Email Password Details

In this example, we are going to create a *Permission* that stops everybody from putting the value of the password property of the **User Biskit Type** into an email. To do this we will be using the special user [nobody](#)⁶⁵⁴

1. Create a *Permission* that applies when a **User** is **Read**, give it a name, select the *property Password*, make it negative (**Authorisation** is **Deny Permission**).

Name	No one can email password details
Enabled	true ▼
Action	Read ▼
Data Type	User
Data Property	Password ▼
Authorisation	Deny Permission ▼
Priority	0
Comments	

Conditions Applies To Does Not Apply To

☐ Include Everybody

Individual Users

nobody (nobody) ▼

	Login name	Given name	Other name	Family name
<input type="checkbox"/>	nobody	nobody		

Remove

User Groups

Please select a User Group to add ▼

Selection empty

Property Path to User(s) or User Group(s)

Select a property

Selection empty

Roles

☐ Use Roles

2. Under **Applies To**, untick **Include Everybody**, select **Individual Users** and then *nobody*.
3. Save the *Permission*.

Use this mechanism on any information you do not want sent by e-mail.

Example 7: Hiding Resources And Booking Properties

Sometimes the administrator may want to create *resources* that only some people may access, or define *properties* on a *booking* that some people shouldn't be allowed to see. This is achieved by setting up *permissions* that control who can access what.

To hide some *bookings* from some people:

- Deny them *READ permission* on the *bookings* they should not be able to see.

If you want to hide a *booking property* from some people, then:

- Create a property-level *Booking permission* that denies them *READ* rights on that *property*
- Create a separate *permission* for each *property* you need to hide

If you want to hide a *resource* from someone, here is the list of *Permissions* that need to be set up:

- Deny them *EXISTS permission* on the *resource*
- Deny them *READ permission* on *bookings* for that *resource*
- Deny them *READ permission* on *templates* for that *resource*.

To set up the first permission.

Action	Exists
Data Type	Resource
Condition	Value of self equals Specified value (name of resource)

Self is *property* of type *Biskit* that points to the *Biskit* owning the *property*. By using the **self** *property* rather than the **name** *property* the primary key of the *Biskit* is stored in the *Condition* and not the name, therefore if the name changes the *Permission* will still work.

6.13 Workflows

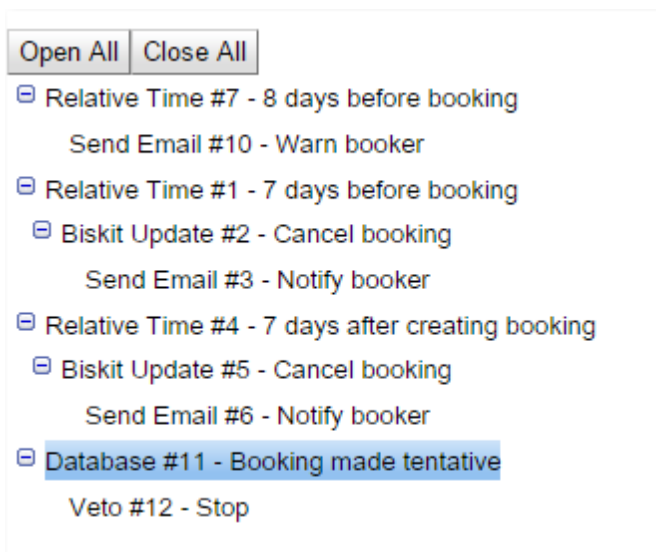
[Workflows](#)⁶⁵⁶ are designed to be very flexible, so that you can automate additional bespoke functionality in **Calpendo**. There are a number of [Workflow Events](#)⁶⁵⁶ that can trigger a *Workflow*, and each [trigger](#)⁶⁵⁶ can then run a number of [Workflow Actions](#)⁶⁵⁶ each of which can have their own child actions and so create the required functionality. Additionally Workflows can be run from menu items (see [Run User Workflow Event](#)⁴⁹⁴), and from buttons associated with a [Biskit Def](#)⁶⁵² created in the [Workflow Editor](#)⁴⁸⁵ from the "Create Workflow Button" button.

6.13.1 How Workflows Work

The [Workflow](#)⁶⁵⁶ environment is really a high level programming environment that is entirely configured from a web browser.

When a [Workflow Event](#)⁶⁵⁶ is triggered, its child [actions](#)³⁶⁵ are executed. Each action can itself have child actions. A workflow is a grouping of events and their actions, which means a single workflow can contain many events. The order in which child *actions* should run can be configured.

Here is an example of the [Tentative Booking](#)²⁷⁸ *Workflow* for **Calpendo**, showing four *Workflow Events* that can be triggered, each having a child *Workflow Action*, and **BiskitUpdate #5 Action** having its own child *Action*.



Both *Workflow Events* and *Workflow Actions* may have [Conditions](#)¹⁰⁷ associated with them to define whether they should be run. If an *Action* is not run due to *Conditions* or fails due to some error then its child *Actions* are not run, although sibling *Actions* would be.

Workflow Header

When a *Workflow* is created the following [properties](#)⁶⁵⁵ are available.

Under the **Details** tab

Property	Value
Name	The name of the <i>Workflow</i>
Enabled	Whether the <i>Workflow</i> is enabled to run or not.
Main Type	The Biskit Type this <i>Workflow</i> is mainly associated with. This is used to determine where in the <i>Workflow</i> tree this <i>Workflow</i> will be displayed. This is only for display purposes and does not affect its functionality.

Details Versioning Comments	
Name	<input type="text"/>
Enabled	true ▼
Main Type	No grouping type

Under the **Versioning** Tab

Property	Value
Version	The version number of the <i>Workflow</i>
Created By	Who created the <i>Workflow</i> .
Created	The date the <i>Workflow</i> was created.
Updated By	Who last updated the <i>Workflow</i> .
Updated	The date of the last update of the <i>Workflow</i> .

Details Versioning Comments	
Version	0
Created By	
Created	
Updated By	
Updated	

Under the **Comments** Tab

Property	Value
Comments	An area to record comments about the <i>Workflow</i>

Workflows and Conditions

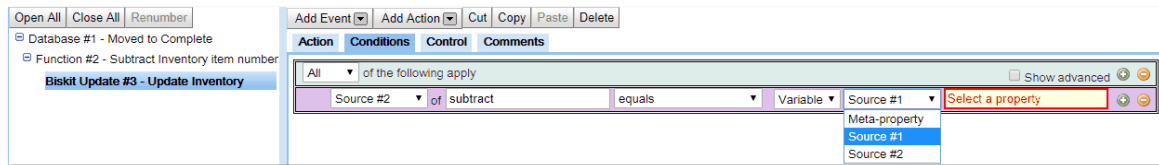
Workflow Events and *Workflow Actions* can both have *Conditions*.
Workflow Events Conditions access a number of *properties* associated with the *Event*.
These are different for each type of *Event*.

All *Events* have the following properties that can be accessed in their *Conditions*:

Example Properties	Notes
executedTime	The time the <i>Event</i> was executed.
triggerTraceString	A list of all the triggers run up until this point.

Each *Event* has its own custom properties that it exposes and are available to its child *actions* and can be used in the *event* conditions. See the page for each *event* for the details.

Workflow Actions can access information on all their ancestor *actions* and the *Event* that triggers that section of the *Workflow*. To do this choose which *Event/Action* will be the **Source** for the *Condition*, and then the appropriate list of *Properties* will be displayed for that **Source**. This allows the *Condition* to compare information from different *Actions/Events*.



All *Actions* have the following properties that can be used in conditions:

Example Properties	Notes
completedOkay	Whether the action Failed or Completed OK
errorMessage	Error message associated with failure
executedTime	The time the <i>Action</i> was executed.
stackTrace	The stack trace if failure occurs
triggerTraceString	A list of all the triggers run up until this point.

Each individual *event* and *action* may make additional information available for their child *actions*. See the documentation for each individual type of *event* or *action* for the details.

Workflows and Permissions

Some **Events** are triggered by users, and others are not. **Database Events** are usually triggered by users, although they can also be triggered by the system itself. In the case of a **Database Event** caused by a person doing something, then a user is "available". If this is a timed event, then there is no user available.

When an **Action** tries to create/update/delete something, there will be a fake/temporary user created. If there is a real user available, and that real user has [Permission](#)⁶⁵⁴ to perform the action, then it's that user whose *permissions* will apply. If there is no real user available, or the real user has no *permission*, then it's the fake user that will be used.

The fake user is endowed with the [User Type](#)⁶⁵⁶ and [Roles](#)⁶⁵⁶ assigned in the **Action**. These are defined on the **Permissions** tab in an *Action*.

However, there are limits to this. If the **admin Role** is added to an **Action**, then only users with the **admin role** will be able to modify the *Workflow* containing that action.

Also, if there is a real user available, then even if the fake user is the one whose *permissions* are used to validate the *Workflow Action*, it will be the real user that will be associated with the audit log entry. Audit log entries also now record the *Workflow* and *Action* that made a change, where performed by a *Workflow*.

Storing Temporary Data During Workflow Execution

If there is a requirement in a *Workflow* to store data temporarily, then you can create temporary variables with an *action* called [Create Variables](#)³⁷¹. You can create any of the variable types available inside the [Bakery](#)⁵³⁷ including their subtypes.

How To Check What Happened When a Workflow Runs

Go to **Admin-->System Events** and search for events around the time the *Workflow* should have run. Every time a *Workflow Action* or *Workflow Event* is run, there's an event created in [System Events](#)²⁰⁴.

Each event and action can be configured so that it does not record a system event when it runs. In that case, there would be nothing in the system events to look at. Some events are, or may be, triggered frequently. If this is true in your case, then they should have system events disabled to avoid excessive data collection. For this reason, [Anonymous Http Events](#)³⁴² have their system events disabled by default.

6.13.2 Workflow Events

There are a number of types of [Workflow Events](#)⁶⁵⁶ available to trigger with a [Workflow](#)⁶⁵⁶.

Event	Description
Anonymous HTTP	Triggers off an HTTP or HTTPS request to a special URL.
Custom Function	Provides an implementation of a function that can be called from a Function Workflow Action ³⁸⁸ .
Database	Triggers off either Create, Update or Delete events of a particular Biskit Def ⁶⁵² .
Privileged Search	Triggered off a request to execute a special type of search that allows access to some aspects of data not otherwise allowed.
Process	Triggers off button presses for a stepped-edit procedure where a biskit is edited across many pages with next and previous buttons to go between the pages.
Relative Time	Triggers at a time relative to a <i>property</i> on a particular <i>Biskit Def</i> .
Reminder	Triggers before a reminder is sent, so modifications may be made to the email.
Timed	Triggers at a particular time. Can be repeated.
User Login	Triggers when a user logs in.
User	Triggers from a menu item of the type <i>Run User Workflow Event Page Menu</i> or a <i>Workflow Button</i> or a <i>dynamic workflow page</i> .

Event types available in Calpendo but not other Exprodo programs:

Event	Description
Booking Rule	Triggers off the running of Rules ⁶⁵² when a booking ⁶⁵² is created or updated so that you can modify the result of a rule or add a rule implemented in a workflow.
Remote Authorisation Request	Triggers when a request is made to authorise a user to use a remote device, which is linked with this system.
Remote User Identification Request	Triggers when a request is made to identify a user using a remote device, that is linked with this system.

All *Event* types have **Event**, **Conditions**, **Control** and **Comments** tabs.

Tab	Description
Event	Define how the event will function.
Conditions	Define here the Conditions ⁶⁵³ required for this <i>Event</i> to be triggered.
Control	Define any controls for this event.
Comments	An area to write any comments about the event and what it is doing.
Output	Shows details about the data output from the event/action that later actions can use.
References	Lists what this action/event references and what references this action/event.

Event	Conditions	Control	Comments	Output	References																
<table border="1"> <tr> <td>Trigger Type</td> <td>Database Workflow Event</td> </tr> <tr> <td>Name</td> <td>Booking costs</td> </tr> <tr> <td>ID</td> <td>68</td> </tr> <tr> <td>Target Type</td> <td>Booking</td> </tr> <tr> <td>Create Events</td> <td>true</td> </tr> <tr> <td>Update Events</td> <td>true</td> </tr> <tr> <td>Delete Events</td> <td>false</td> </tr> <tr> <td>Vetoable Events</td> <td>true</td> </tr> </table>						Trigger Type	Database Workflow Event	Name	Booking costs	ID	68	Target Type	Booking	Create Events	true	Update Events	true	Delete Events	false	Vetoable Events	true
Trigger Type	Database Workflow Event																				
Name	Booking costs																				
ID	68																				
Target Type	Booking																				
Create Events	true																				
Update Events	true																				
Delete Events	false																				
Vetoable Events	true																				

Event Tab	Description
Trigger Type	The type of the <i>Workflow Event</i> .
Name	The user defined name for this event.
ID	The ID number for this event.
Event Specific Information	Defined by the type of Event. For full details for each type see the next section Event Types .
Run Now	Timed and User <i>Workflow Events</i> have a Run Now button to start immediate activation. Very useful for testing purposes.

Event	Conditions	Control	Comments	Output	References						
Enabled		true									
Children to Run		All children									
Record System Events		Record all system events									
Sort Order		0									
Last Ran		26 Sep 2018 13:42									
Ignore events caused by:											
<table border="1"> <tr> <td>Other Workflows</td> <td><input type="checkbox"/></td> </tr> <tr> <td>This Workflow</td> <td><input type="checkbox"/></td> </tr> <tr> <td>This Event</td> <td><input checked="" type="checkbox"/></td> </tr> </table>						Other Workflows	<input type="checkbox"/>	This Workflow	<input type="checkbox"/>	This Event	<input checked="" type="checkbox"/>
Other Workflows	<input type="checkbox"/>										
This Workflow	<input type="checkbox"/>										
This Event	<input checked="" type="checkbox"/>										

Control Tab	Description
Enabled	Whether this <i>Event</i> will run. Disabled <i>Events</i> will be displayed in red in the <i>Workflow</i> definition list.
Children To Run	Whether to run only the first child <i>Workflow Action</i> or all of the child <i>Workflow Actions</i> . The order to run a child <i>Workflow Action</i> is determined by the Sort Order of the <i>Actions</i> .
Record System Events	Choose whether to record system events created by this <i>Workflow Event</i> and its children.
Sort Order	This determines in which order the <i>Workflow Events</i> are fired. If multiple events are triggered at the same time, then the one with the lowest sort order will be executed first.
Last Ran	When this <i>Workflow Event</i> was last run.
Ignore events caused by:	Specify whether this <i>Workflow Event</i> will ignore events created by: <ul style="list-style-type: none"> • Other Workflows • This Workflow • This Event (ticked by default)

6.13.2.1 Anonymous HTTP Workflow Event

This event is triggered whenever an HTTP or HTTPS request is made to a URL beginning *https://your_database/anon/*

When this happens, a *Request* biskit is created that captures information about the HTTP request, then any workflow event matching will be triggered. You may then use the properties stored in the event to respond to the particular URLs that you want to. If there is no suitable

event configured for a particular URL, then the caller will receive a [501 \(Not implemented\) status response](#).

Event Properties

As well as the standard properties that can be set on all events, the following properties can be set on this type of event:

Property	Type	Description
GET Events	Boolean	<i>true</i> if an http GET should trigger this event, and <i>false</i> if we should ignore GET requests.
POST Events	Boolean	<i>true</i> if an http POST should trigger this event, and <i>false</i> if we should ignore POST requests.
PUT Events	Boolean	<i>true</i> if an http PUT should trigger this event, and <i>false</i> if we should ignore PUT events.
DELETE Events	Boolean	<i>true</i> if an http DELETE should trigger this event, and <i>false</i> if we should ignore DELETE requests.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Property	Type	Description
httpMethod	JavaEnum - drop down of: <ul style="list-style-type: none">• GET• DELETE• POST• PUT	Specifies which HTTP method was used.
ipAddress	String	The apparent IP address of the request.
userAgent	String	The user agent string reported by the browser.
parameters	Biskit of type <code>HttpRequestParameters</code> , described below	Defines all the parameters provided in the URL or (for a POST) in the body of the request. See below for details and examples.
response	Biskit of type <code>AnonymousHttpResponse</code>	You can modify properties on this response biskit to control what is sent back to the caller. Failing to configure the response will result in a 501 error status being returned.

`HttpRequestParameters` biskits contain *no* properties by default. Instead, for each parameter specified in the URL, a string-valued property of the same name will be on this biskit with the value of the parameter. For example, if there's an HTTP GET request to

`http://your_database/anon/something?first=hello&second=world` then the event's *response* biskit would contain properties called *first* and *second* with values *hello* and *world* respectively. Since these properties do not exist until the HTTP request is received, you must use a [workflow function action](#)³⁸⁸ to extract it, and this would typically be a function like [getAsString](#)³⁹⁸ or [getAsInt](#)³⁹⁶.

The response biskit, of type `AnonymousHttpResponse`, contains the following properties:

Property	Type	Description
status	Java Enum - drop down of all the known response codes, such as "200 OK" and "404 Not Found". See HTTP Response codes .	The HTTP response code that should be returned. The default is "501 Not implemented". The most common response code is "200 OK".
headers	Biskit of type <code>HttpHeaders</code>	This contains <i>no</i> properties by default. For every header that is on the request, the <i>headers</i> biskit will have a string-valued property of the same name to store the header value.
contentType	String	Specifies the content type of the response, for example "application/json". See MIME Types
body	String	A string that provides the body of the response you want to provide.

6.13.2.2 Booking Rule Workflow Event

When a [booking](#)⁶⁵² is created or updated, Calpendo checks its [Rules](#)⁶⁵². Each time a rule is run, it will check to see if there are any *Booking Rule Workflow Events* to fire. An event will be triggered once when rules are checked, and then once again for every rule. The event provides in its output the rule that it was triggered from. If that rule is *null*, then this is the invocation called at the beginning of checking rules. Otherwise, it's being invoked for a particular rule.

Event Properties

As well as the standard properties that can be set on all events, the following properties can be set on this type of event:

Property	Type	Description
Booking Biskit Type	Biskit of type <code>BiskitDef</code> that must be a subtype of <code>Booking</code>	The subtype of booking that this event should apply to. This defaults to the top-level <code>Booking</code> type, but you can restrict it to only responding to a more specific subtype of <code>Booking</code> .
Rule Biskit Type	Drop down of all available rule types	The subtype of rule that should trigger this event.

Property	Type	Description
Create Events	Boolean	<i>true</i> if this event should be triggered when rules are responding to a request to create a booking, and <i>false</i> if we should ignore booking creation events.
Update Events	Boolean	<i>true</i> if this event should be triggered when rules are responding to a request to update a booking, and <i>false</i> if we should ignore booking update events.
Accepted Events	Boolean	<i>true</i> if this event should be triggered when a rule indicates an acceptance level of <i>Accept</i> , and <i>false</i> if we should ignore booking <i>Accept</i> events.
Rejected Events	Boolean	<i>true</i> if this event should be triggered when a rule indicates an acceptance level of <i>Reject</i> , and <i>false</i> if we should ignore booking <i>Reject</i> events.
Warned Events	Boolean	<i>true</i> if this event should be triggered when a rule indicates an acceptance level of <i>Warn</i> , and <i>false</i> if we should ignore booking <i>Warn</i> events.

Note that you must choose whether the rule should be triggered by a booking being created, a booking being updated, or both. Set at least one of *Create Events* or *Update Events* to *true*, otherwise a *Booking Rule Workflow Event* would never be triggered.

Once you've decided whether to be triggered by a *create* or *update*, then you must also decide whether this event should be triggered by a rule that has decided to accept, reject or warn about a booking. Again, you must set at least one of these to *true*, or else the event would never be triggered.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Type	Notes
oldBooking	Biskit of type <i>Booking</i>	The booking ⁶⁵² as it was before the change that triggered rules ⁶⁵² .
newBooking	Biskit of type <i>Booking</i>	The booking ⁶⁵² as it is after the change that triggered rules ⁶⁵² , and after any changes that may have been made while checking rules ⁶⁵² .
rule	Biskit of type <i>Rule</i>	The rule ⁶⁵² that triggered this particular invocation of the rule event. If this is <i>null</i> , then this is the invocation that happens at the start of checking rules ⁶⁵² . Otherwise, it's being called for a particular rule ⁶⁵² .
realUser	Biskit of type <i>User</i>	The real user that has caused this event to happen. This would normally

Properties	Type	Notes
		be not <i>null</i> , but could be <i>null</i> .
pretendUser	Biskit of type <i>User</i>	The pretend user running this rules ⁶⁵² request. It can be different from <i>realUser</i> when using the rule validator ²⁷⁷ .
output	String	A string produced by the rule ⁶⁵² . This is not normally seen by the end users, but can be useful in the rule validator ²⁷⁷ to see information about what the rule ⁶⁵² did.
update	Boolean	<i>true</i> when rules ⁶⁵² are called due to a booking ⁶⁵² being updated, and <i>false</i> when rules ⁶⁵² are called due to a booking ⁶⁵² being created.
validatorOnly	Boolean	<i>true</i> when rules ⁶⁵² are running from the rule validator ²⁷⁷ .
rulesResult	Biskit of type <i>RulesOutcome</i>	A Biskit ⁶⁵² of type <i>RulesOutcome</i> , defined below.
repeatsAffected	JavaEnum - drop down of: <ul style="list-style-type: none"> • This item • This and later items • All items 	Indicates which repeats are affected by the change. Only relevant for repeat bookings. For non-repeat bookings, this will be <i>null</i> .

This is the content of the *RulesOutcome* biskit:

Properties	Type	Notes
rejectionLevel	Drop down of Accept, Warn, Reject	The result of running the rule.
message	String	The message generated by the rule to be given to the user
ruleName	String	The name of the rule that generated this outcome.
tryBookingRequest	Boolean	<i>true</i> if the rule recommends retrying the booking as a request because it may change the result, and <i>false</i> if there is no recommendation to retry as a request.
tryBookingTentative	Boolean	<i>true</i> if the rule recommends retrying the booking as tentative because it may change the result, and <i>false</i> if there is no recommendation to retry as tentative.

You can modify the properties on the *rulesResult*. Note that you should only set something here if you want the rejection level to be at least as high as the one already set in the *rulesResult*. The order of the rejection levels is **Accept, Warn, Reject**.

If you were to set the rejection level to **Accept** regardless of what was already there, then you could be overwriting previous results and making it seem that a rule that had failed had actually passed.

If multiple booking **Rule** events could be called for the same booking then making sure only the highest level is reported is down to the programmer. (Unlike **Rules** which always set the highest level no matter the order they are run in)

6.13.2.3 Custom Function Workflow Event

This allows the user to specify a user defined function that can be called from a [Function Workflow Action](#)³⁸⁸. It allows the user to define a group of Workflow actions that may occur in a Workflow many times once. Define the Event, setup the input and output variables that will be required, and set up the name. Then call this function from the same or other workflows. In order for the Custom Function to be viewable inside the Function->Workflow the Workflow must be saved. Then the user can continue to edit and use the function.

Also, if a Custom Function is defined as being only useable within a Workflow it is possible to provide an alternative implementation of a global function, even one provided with the system, but in order to do that, the **CustomFunctionWorkflowEvent** must be created with both a matching function name and also matching input and output names as well. In other words, the names of the inputs and outputs are seen as an integral part of the function-matching system.

Event Properties

As well as the standard properties that can be set on all events, the following properties can be set on this type of event:

Property	Description
Function name	The name the function should have. This must be set, and it must be unique.
Display Category	When you create a Function Workflow Action ³⁸⁸ , you can find functions in a drop-down where functions are grouped by <i>category</i> . The "Display Category" is your chance to customise the category used when your custom function is added to this drop-down
Hide From Other Workflows	Indicates whether the custom function should be hidden from other workflows, so it's only usable from within the workflow that defines it. Alternatively, it can be used from any workflow.
Inputs	These are the values that you require when your custom function is called. There is no concept of optional inputs - they must all be provided. You can define them to be of any type, and provide a name for them.
Outputs	These are the values that you will set up as outputs generated by the custom function. As for inputs, you specify their name and type. Anybody calling your function will then be able to access those outputs.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
inputs	A biskit whose properties represent all the inputs defined to exist on the custom function event.
outputs	A biskit whose properties represent all the outputs defined to exist on the custom function event.

6.13.2.4 Database Workflow Event

A Database Workflow Event is triggered in response to a change in the data stored in the database. That means it represents data being created, updated or deleted.

Event Properties

As well as the standard properties that can be set on all events, the following properties can be set on this type of event:

Property	Description
Target Type	The type of biskits whose creation, deletion or update triggers this event.
Create Events	<i>true</i> if an this event should be triggered by the creation of a biskit, and <i>false</i> if we should ignore biskit creation events.
Update Events	<i>true</i> if an this event should be triggered by the update of a biskit, and <i>false</i> if we should ignore biskit update events.
Delete Events	<i>true</i> if an this event should be triggered by the deletion of a biskit, and <i>false</i> if we should ignore biskit deletion events.
Vetoable Events	When a database event occurs, you sometimes want to run inside the database transaction so that you can modify the data before it is saved or so that you can prevent (or veto) the database change. But you sometimes would prefer to be outside the database transaction so that the event is triggered only after the database change has completely finished. Set this option to <i>true</i> if we should run inside the database transaction and so have the option of preventing the transaction from completing, and <i>false</i> if we should run after the transaction has completed.

Vetoable Events

The option for running with vetoable events can be a bit confusing. It's all a question of whether you run *inside* the database transaction or *after* it has completed. Let's explain this with some examples.

Suppose you create a system that records somebody purchasing something, and you use a workflow to keep track of how much money they have. Then a biskit might be created when a purchase is made. But suppose the workflow discovers that the user does not have enough money to buy the goods. Then the workflow should *veto* the change by calling a [Veto Workflow Action](#)⁴⁸⁵. For this to work, the event *must* have been set to work with vetoable events so that it runs *inside* the database transaction. Otherwise, the transaction would complete before you checked whether they had sufficient money.

Now suppose you want to send an confirmation email whenever somebody makes a purchase. If there are multiple workflows that do things with or to the purchase biskit, you might find that you send the email and then something else vetoes the purchase. That would mean the email gives misleading information. So you should make sure the database event that triggers the email is done *without* vetoable events because that means it will only be called *after* the change has been finalised.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
new	The new value of the biskit after the change. <i>old</i> and <i>new</i> are identical for a <i>create</i> and a <i>delete</i> .

Properties	Notes
old	The old value of the biskit before the change.
AuditLog	Information stored in the AuditLog for this change. This records information about who made the change, when, and what the biskit was like after the change.
UserMakingTheChange	Indicates which user made the change.
Crud	<i>Property</i> storing the type of change Create, Update, Delete.

6.13.2.5 Privileged Search Workflow Event

Fill this in

Event Properties

As well as the standard properties that can be set on all events, the following properties can be set on this type of event:

Property	Description
Target Type	The type of biskits whose creation, deletion or update triggers this event.
Create Events	<i>true</i> if an this event should be triggered by the creation of a biskit, and <i>false</i> if we should ignore biskit creation events.
Update Events	<i>true</i> if an this event should be triggered by the update of a biskit, and <i>false</i> if we should ignore biskit update events.
Delete Events	<i>true</i> if an this event should be triggered by the deletion of a biskit, and <i>false</i> if we should ignore biskit deletion events.
Vetoable Events	When a database event occurs, you sometimes want to run inside the database transaction so that you can modify the data before it is saved or so that you can prevent (or veto) the database change. But you sometimes would prefer to be outside the database transaction so that the event is triggered only after the database change has completely finished. Set this option to <i>true</i> if we should run inside the database transaction and so have the option of preventing the transaction from completing, and <i>false</i> if we should run after the transaction has completed.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
new	The new value of the biskit after the change. <i>old</i> and <i>new</i> are identical for a <i>create</i> and a <i>delete</i> .

Properties	Notes
old	The old value of the biskit before the change.
AuditLog	Information stored in the AuditLog for this change. This records information about who made the change, when, and what the biskit was like after the change.
UserMakingTheChange	Indicates which user made the change.
Crud	<i>Property</i> storing the type of change Create, Update, Delete.

6.13.2.6 Process Workflow Event

A process, also known as a stepped-edit process, is the procedure for viewing and/or editing something by a sequence of pages that have next and previous buttons on them so that you can step through the pages. Each page can have a number of buttons, not just *next* and *previous*. Each time a button is pressed, the data being displayed is sent to the server to be saved, and a *Process Workflow Event* is generated. The workflow then has the opportunity to modify which page will be displayed next. See [Processes](#)⁶²² for details on the stepped-edit processes.

Event Properties

As well as the standard properties that can be set on all events, the following properties can be set on this type of event:

Property	Description
Target Type	The <i>BiskitDef</i> of the biskit edited by the process.
Finish events	<i>true</i> if this event should be triggered by the user clicking the <i>Finish</i> button.
Next events	<i>true</i> if this event should be triggered by the user clicking the <i>Next</i> button.
Previous events	<i>true</i> if this event should be triggered by the user clicking the <i>Previous</i> button.
Reset Event	<i>true</i> if this event should be triggered by the user clicking the <i>Reset</i> button.
Help Events	<i>true</i> if this event should be triggered by the user clicking the <i>Help</i> button.
Initial Events	<i>true</i> if this event should be triggered by the initialisation of a stepped edit process.

Properties Available for Child Actions To Use

Property	Type	Description
new old	Biskit whose type will be	When a button is pressed, we are sent the details of the biskit displayed. When the display allowed

Property	Type	Description
	determined at run time	the user to change information, then the values provided by the user will be stored in the new property and the values before the change will be stored in the old property. If this is the first step or the step wasn't an update, then new and old will be identical.
AuditLog	Biskit of type AuditLog	Information stored in the AuditLog for this change.
UserMakingTheChange	ExprodoUser	The user who performed the step, or <i>null</i> if it wasn't a real person.
EntityMakingTheChange	PermissionsUser	The user who performed the step. This might be a real user or a system user. When this is a real user, then EntityMakingTheChange is identical to UserMakingTheChange and it would actually be of type User.
Button	JavaEnum of Next, Previous, Reset, Help, Finish	The type of button pressed that caused this event to be triggered.
Response	Biskit of whose type is described below	The response to send. Modify this to change the next step and possibly add a message to be given to the user.
StepPerformed	Biskit of type Step	The step just performed by the user.

The *Response* property is a biskit with the following properties:

Property	Type	Description
nextStep	Biskit of type Step	The next step to be taken
message	String	A message to display to the user after this step
nextPageToken	String	The page to switch to. This should be the URL of the page that follows the '#'. For example, use 'about' to go to the 'About' page.
biskitToEdit	Biskit	The biskit that should be edited (only used for an initial event when starting the edit process)
messageType	Java Enum of ERROR, WARNING, INFO	Specifies whether the message being displayed is an error, a warning or just information.

A typical reason to use a *Process Workflow Event* is because you want an opportunity to modify some data between steps, or to display a message to a user between steps, or to choose which should be the next step to take. The default course of action is that the next step is decided by the definition of the process (see [Processes](#) ⁶²²), but you can have the choice of the next step be conditional using this event.

To do any of these things, you would use a [Biskit Update Workflow Action](#)³⁷¹ to modify the biskit in the *response* property on the event.

6.13.2.7 Relative Time Workflow Event

Triggers at a time relative to a *property* on a particular *Biskit Def*. For example, it might trigger a set time before a user's account is set to expire or a set time after their last login.

Event	Example Properties	Notes
Relative Time	Biskit	Information about the <i>Biskit</i> that fired the <i>Event</i> .

Event Properties

As well as the standard properties that can be set on all events, the following properties can be set on this type of event:

Property	Description
Relative to PropertyDef	Specifies the property with a date or date/time value to which the event should be relative.
Time Offset	The amount of time before or after the date found in the property when the event should trigger. Use a positive value to trigger <i>after</i> and a negative value to trigger <i>before</i> .

For example, to trigger one day before a user is set to expire, you would create a relative time workflow event with the property set to User.Expiry Date, and set the time offset to minus one day.

If you trigger this event from a property that stores a date property (that is, one without a time component), then you will also be asked for a time of day. This will be the time when the event will trigger on the calculated day.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Property	Description
biskit	The biskit that triggered the event. For example, if you configure an event to trigger the day before a user expires, the <i>biskit</i> will be set to the user.

Business Days

The **Relative Time Event** has a concept of **Business Days**. These are weekdays Monday-Friday, not including Holidays. There is a *Biskit Def* called **Holiday Date** which is used to store all the days defined as being Holidays.

6.13.2.8 Reminder Workflow Event

Some biskits have a built-in reminder mechanism. Depending on which Exprodo program you are using and which modules are loaded, you may have no such biskits, or you may have one or many. When faced with these built-in reminders, a user is typically offered the option of whether or when to have a reminder, but not what the content of the reminder should be. The content is automatically generated without any customisation. However, it will then trigger a *Reminder Workflow Event* which provides an opportunity for customising the content of the reminder, should you wish to do so.

Event Properties

As well as the standard properties that can be set on all events, the following properties can be set on this type of event:

Property	Description
Reminder for BiskitDef	This is the BiskitDef of the biskit whose reminders you wish to customise.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Property	Description
biskit	This is the biskit that is being reminded about.
email	This biskit specifies what will be sent as the reminder. It will be populated with the standard reminder content before the event is triggered so that you only have to modify any parts of it that you want to customise.

The email biskit contains the following properties, any of which you can modify:

Property	Type	Description
subject	String	The subject of the email
content	String	The body of the email (plain text and not HTML formatted)
from	String	The email this appears to come from
replyTo	String	The email address replies should be sent to
to	Set of String	A set of raw email addresses to send the email to
cc	Set of String	A set of raw email addresses to copy the email to

Property	Type	Description
bcc	Set of String	A set of raw email addresses to blind copy the email to
priority	Int	The priority of the email
toUsers	Set of biskits of type ExprodoUser	A set of the users to send the email to
ccUsers	Set of biskits of type ExprodoUser	A set of the users to copy the email to
bccUsers	Set of biskits of type ExprodoUser	A set of the users to blind copy the email to

6.13.2.9 Remote Authorisation Request Workflow Event

If you use a method of capturing actual usage from a remote program, such as CAR, the [Calpendo Activity Recorder](#), then when somebody logs in to a remote computer, and is identified as a particular Calpendo user, we then need to decide whether they are allowed to use the remote system. That's what this event is designed to do: to provide an opportunity to customise the decision about who is allowed to use which instrument.

Event Properties

There are no properties you can set on the event beyond the standard properties that can be set on all events.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Property	Type	Description
user	Biskit of type Exprodo User	The user whose authorisation is being evaluated.
ipAddress	String	The IP address that the request appears to be coming from. Depending on the nature of the networks between the instrument and Calpendo, this might not be the IP address the instrument believes it has, but it could be the IP address of a network router.
declaredIpAddressesCSV	String	A comma-separated list of the IP addresses being declared by the remote system. This will be IP addresses as used by all of its network devices.
declaredIpAddresses	Set of String	A set of the IP addresses being declared by the remote system. This will be IP addresses as used by all of its network devices.
authenticationMethodName	String	The name of the authentication method that was used by the remote user.

Property	Type	Description
hostname	String	The name by which the remote server knows itself.
osDomainName	String	The name of the computer domain used by the remote user.
result	Biskit	This biskit contains the result of the authorisation request, and its properties are the ones you can modify to choose whether a user should be authorised.

The result biskit is the one whose properties you should modify if you use this event. It has only two properties:

Property	Type	Description
authorised	Boolean	<i>true</i> if the user is allowed to use the remote system and <i>false</i> otherwise.
message	String	A message to be given to the remote user.

6.13.2.10 Remote User Identification Workflow Event

If you use a method of capturing actual usage from a remote program, such as CAR, the [Calpendo Activity Recorder](#), then when somebody logs in to a remote computer, a message will be sent to Calpendo with information the user that's logged in. Depending on how the login is performed, Calpendo might receive a Windows user name. We then need to map that to a Calpendo user. The default will look for a local user with the same login name as the remote user. But you may need to do something different.

That's what this workflow is for: it provides an opportunity to identify who is using a remote system by the information we receive. The event provides all the information given to it, and you can then use that to calculate the Calpendo user involved.

Event Properties

There are no properties you can set on the event beyond the standard properties that can be set on all events.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Property	Type	Description
ipAddress	String	The IP address that the request appears to be coming from. Depending on the nature of the networks between the instrument and Calpendo, this might not be the IP address the instrument believes it has, but it could be the IP address of a network router.

Property	Type	Description
declaredIpAddressesCSV	String	A comma-separated list of the IP addresses being declared by the remote system. This will be IP addresses as used by all of its network devices.
declaredIpAddresses	Set of String	A set of the IP addresses being declared by the remote system. This will be IP addresses as used by all of its network devices.
hostname	String	The name by which the remote server knows itself.
osDomainName	String	The name of the computer domain used by the remote user.
ipAddress		The apparent IP address of the request.
loginName	String	The login name used by the user on the remote system.
result	Biskit	A biskit which contains the only part of this event's data that you should modify. The result biskit contains a single property, user which you should set to the user that you have identified.

6.13.2.11 User Workflow Event

A user workflow event is one that is triggered by one of a number of user actions. This can be:

- a workflow button, which is a button designed to run a user workflow event
- a custom menu item designed to run a user workflow event
- a dynamic workflow page, which is an HTML page that can be created with a response to a *User Workflow Event*, can trigger a further *User Workflow Event* when a form on the page is submitted.

Event Properties

There are no properties you can set on the event beyond the standard properties that can be set on all events.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Property	Type	Description
biskit	<p>Biskit of unknown type.</p> <p>You will need to Type Cast the value to some <i>Biskit Type</i> to have access to the properties on the biskit.</p>	<p>When viewing an individual biskit and a workflow button is pressed, biskit is the biskit that was being displayed.</p> <p>When the <i>User Workflow Event</i> is triggered by other means (such as a menu) or from a workflow button associated with a list of biskits, then the value of the biskit property will be <i>null</i>.</p>
biskits	<p>List of Biskits of unknown type.</p> <p>You will need to Type Cast the values to some <i>Biskit Type</i> to have access to the properties on the biskit.</p>	<p>When viewing a list of biskits and a workflow button is pressed, biskits provides all those biskits that were displayed.</p> <p>When the <i>User Workflow Event</i> is triggered by other means (such as a menu) or from a workflow button associated with an individual biskit, then the value of the biskits property will be <i>null</i>.</p>
buttonLabel	String	The label on the button that was pressed to trigger this event.
menu	Biskit of type Run Workflow Event Page Menu ⁴⁹⁴	If this event was triggered by a menu, then this specifies the item the user pressed, including the label on the item.
formContent	<p>Biskit</p> <p>See table below for a definition of the properties on this biskit.</p>	If the event was triggered by a form within a <i>dynamic workflow page</i> , then the formContent property provides the information submitted in the form.
response	<p>Biskit</p> <p>See table below for a definition of the properties on this biskit</p>	This is a biskit that your workflow can modify to determine what happens next to the user's browser. This includes the ability to have the user download a file, display a <i>dynamic workflow page</i> , switch to another page or display a pop-up message.
state	String	<p>This is a string whose content is entirely defined by your workflows. When one <i>User Workflow Event</i> runs, and generates a dynamic workflow page which can trigger a secondary <i>User Workflow Event</i>, you may want to have access in the secondary event to information that was generated in the primary event.</p> <p>To deal with that, the primary event's workflow would set the state property on the primary event's response biskit, and then the secondary event's state would contain the same value.</p>

Property	Type	Description
		You decide on the format of the value to be whatever you need.
userWaiting	Boolean	Indicates whether the user's browser is waiting for a response. If the user's browser is not waiting the a response, then the response property value is ignored by the browser.

The formContent biskit contains the following properties:

Example Properties	Type	Notes
triggerID	String	The HTML id property of the button that submitted the form.
triggerName	String	This is the name property of the button that submitted the form.
state	String	A workflow-defined string that can be used to persist state across calls to the workflow. The form state is whatever value was set in the response we received from the previous call.
one property for each of the form's named items.	String	<p>When an HTML form is submitted, the browser provides property names and values as strings.</p> <p>When this event is triggered by an HTML form in a dynamic workflow page, those form properties are presented as properties on the formContent biskit.</p> <p>For example, if you have a "First Name" element in the pageContent, then the formContent biskit would have a property called "firstName" whose value is a string. If nothing is entered into the text box, the firstName property would be missing, and so fetching its value would return null. Note that checkboxes and radio buttons yield string values of "true" and "false". Use <code>getAsString()</code> to return a string value.</p>

The response biskit contains the following properties:

Properties	Type	Notes
buttons	List of Biskit of type User Workflow Button	A list of buttons that should be displayed, possibly along with message , where each button may trigger another workflow.
downloadableFile	Biskit of type Temporary File	A file that should be downloaded to the user's browser. Only one file can be downloaded per user workflow event execution.
message	String	A message to display to the user. This is only displayed if the user is waiting for a response from the workflow.
messageType	JavaEnum (drop down) of: <ul style="list-style-type: none"> • Error (HTML formatted) • Warning (HTML formatted) • Info message (HTML, auto-disappears) • Info Message (HTML, manually cancelled) • Info Message (Plain text, manually cancelled) 	The type of message to display. This is only used if there is a value for 'message'.
nextPageToken	String	The page to switch to. This should be the part of the URL of

Properties	Type	Notes
		the page that follows the '#'. For example, use 'about' to go to the 'About' page.
nextUserWorkflowEvent	Biskit of type User Workflow Event	The user workflow event to be run when any form within the page content is submitted.
pageContent	String	HTML to display for the next page. If you set this, then it is the HTML content that you would like to be displayed to the user for the next page. This is known as a <i>dynamic workflow page</i> because it is a page whose content is completely defined by a workflow. You can use this to display a form which will cause a secondary <i>User Workflow Event</i> to be run when the form is submitted.
state	String	A workflow-defined string that can be used to persist state across calls to the workflow. Whatever value you store in here will be presented back in any subsequent User Workflow Event triggered.

Running A Workflow From A Menu

In order to run a *Workflow* from a **Menu**, first create a *Workflow* with a **User Workflow Event**. This event will be run when the **Menu** item is selected.

You can then modify the properties of the **Event's response biskit**, for example to have the user's browser download a file, display a dynamic workflow page, display any standard page, or to display a pop-up message, possibly with buttons that can trigger further **User Workflow Events**.

Once the **Workflow** has been created go to the [Menu Editor](#)⁴⁸⁹ and create a **Menu Custom Page** of type "**Run User Workflow Event**". Here you will specify the **User Workflow Event** the menu item will run and whether the user is waiting for the result.

To create **Buttons** for the **response**, use the **Biskit Create Workflow Action** and create a *biskit* of type **User Workflow Button**, do not save to the database, and set up the properties, these are:

Properties	Type	Notes
Await Outcome	Boolean	Will the user wait for the outcome of pressing this button.
Button Label	String	The label on the button.
Button Tooltip	String	Tooltip to be viewed if the button is hovered over.
Enabled	Boolean	Is the button enabled.
State	String	String property to allow passing of information from the original Workflow to the one to be run by this button.
User Workflow Event	Biskit of type User Workflow Event	The Workflow Event to be run when the button is pressed.

Then add the button to the response using **add** (List:Biskit:typeOf:list; list)add(List:Biskit;; list,Biskit:typeOf:list; item). They will appear in the order they have been added to the list.

Running A Workflow From A Button

In order to run a *Workflow* from a **Button**, first create a *Workflow* with a **User Workflow Event**. This event will be run when the **Button** item is pressed. The **Event** contains a **response biskit** (see above for details), into which some information may be put using **Biskit Update Workflow Action**.

Once the **Workflow** has been created in the **Workflow Editor** press the **Create Workflow Button** and fill in the button details:

Properties	Notes
Button Label	The label on the button.
Button Tooltip	Tooltip to be viewed if the button is hovered over.
Target Biskit Type	Which <i>Biskit Type</i> is the button going to be assigned to.
Enabled	Is the button enabled.
Contexts to display button	View Biskit: Whilst a <i>biskit</i> of the <i>Biskit Type</i> is being viewed.
	Update Biskit: Whilst a <i>biskit</i> of the <i>Biskit Type</i> is being edited.
	View Biskit List: When a list of <i>Biskits</i> is being edited.
User Workflow Event	The Workflow Event to be run when the button is pressed.
Await Outcome	Will the user wait for the outcome of pressing this button.
Sort Order	The order the button will be presented in

Use **Type Cast Workflow Action** to convert *biskit* to be the appropriate type in order to see the *properties*. A response with buttons can be created in the same manner as when running a **Workflow** from a **Menu** (see above).

6.13.2.12 User Login Workflow Event

A *User Login Workflow Event* is triggered every time somebody logs in. This provides you the opportunity to run anything you want at that time, and you can also modify the user's environment. You can:

- Modify the menu a user will get so you can ensure users with particular roles/groups will always get the correct menu.
- Run **Veto Workflow Action** to prevent the user from logging in.
- Choose the **Layouts** of particular *Biskits* the user may see. Call **addUserLayout** function passing in a layout they are to use.
- Modify the **Workflow Created Buttons** the user will see.
 - There's a list of buttons **workflowButtons** on the event.
 - You can add to the list, remove from the list, or modify the buttons in the list.

- Modifying buttons makes sense if you want to disable them for some users. If you do this, then you should make sure the **Biskit Update** action when you modify the button is marked as temporary so that the change is not persisted to the database where it could affect other users.

Event Properties

There are no properties you can set on the event beyond the standard properties that can be set on all events.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Property	Type	Notes
layouts	List of biskits of type User Layout	The layouts the user can use, use addUserLayout to add to this list.
settings	Biskit of type UserSettings	The User Settings allowing customisation of the settings at login time.
user	Biskit of type User	The User Biskit allowing user information to be extracted.
workflowButtons	List of biskits whose type is defined below	The list of buttons created by workflows the user can see

The biskits in the **workflowButtons** property look like this:

Property	Type	Notes
buttons	List of biskits of type User Workflow Biskit Type Button	<p>A list of workflow buttons to be given to the user for display at the appropriate place.</p> <p>Workflow buttons are each configured to display whenever you're viewing a biskit of a particular type, or a list of biskits of a particular type.</p> <p>By using a User Login Workflow Event you can customise which workflow buttons a user has available.</p> <p>When the event runs, it will start with the buttons property set to the standard buttons. You can modify this list to add new buttons, remove existing ones, or modify existing ones just for this user's view (eg removing a button for some users or adding a button for some users).</p>

6.13.2.13 Timed Workflow Event

A *Timed Workflow Event* is one that runs at a particular time, and may repeat. For example, it might run on the first day of every month.

Event Properties

As well as the standard properties that can be set on all events, the following properties can be set on this type of event:

Property	Type	Description
When	Date and time	Specifies a date and time when the event should run.
Repeat	Repeat	This specifies if and how the event should repeat.

Properties Available for Child Actions To Use

There are no properties available for use in any child or descendant action beyond the standard properties.

6.13.3 Workflow Actions

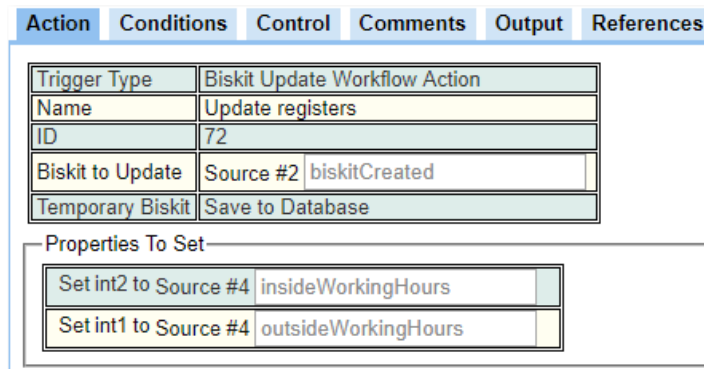
There are a number of [Workflow Actions](#)⁶⁵⁶ available with [Workflows](#).⁶⁵⁶

Action	Description
Biskit Create	Creates a new Biskit ⁶⁵² from a Biskit Def ⁶⁵² .
Biskit Delete	Deletes a <i>Biskit</i> .
Biskit Update	Updates a number of properties on a <i>Biskit</i> .
Create Variables	Creates temporary variables for storage whilst the <i>Workflow</i> runs.
Delay	Creates a delay before continue with the next <i>Action</i> .
Diff	Returns the differences between two <i>Biskits</i> .
Email	Sends an email.
Evaluate Expression	Allows the user to create an expression to be evaluated. Removes the need to have multiple <i>Actions</i> when doing calculations.
Execute System Command	Allows the user to execute a system command direct from the operating system.
Find Text	Looks for regular expressions in a text string and returns all the requested number of times that it occurs. Or look for a text string and replace first or all versions of that string with a different string.
For Each	Pulls information from a list or set to run the following <i>Actions</i> on. Equivalent to a programming For loop.
Function	Acts on a piece of information in a specific way returning the answer.
List Extract	Extracts a <i>property</i> from each <i>Biskit</i> in a list, and returns a list of those <i>properties</i> .

Action	Description
Network Message	Create an network message.
Search	Does a search of the database and returns the result as a list.
Simple	Does nothing, use as a holder, useful if building the equivalent of a programming Case statement.
Sort	Sorts a list into a different order.
Templated Text	Create text to be used in another action.
Type Cast	Checks whether a <i>Biskit</i> has a particular subtype, and it only runs the child Actions if the <i>Biskit</i> does have that subtype. The child Actions can then treat the <i>Biskit</i> as having that subtype, which means there is access to all the properties ⁶⁵⁵ that exist on the subtype.
Veto	Stops the <i>Workflow</i> and writes a message.

Most *Workflow Action* types then have four tabs which are similar to the *Event* tabs, **Action**, **Conditions**, **Control**, **Comments**.

Tab	Description
Action	Define how the action will function.
Conditions	Define here the Conditions ⁶⁵³ required for this <i>Action</i> to be triggered.
Control	Define any controls for this action.
Comments	An area to write any comments about the action and what it is doing.
Output	Shows details about the data output from the event/action that later actions can use.
References	Lists what this action/event references and what references this action/event.



Trigger Type	Biskit Update Workflow Action
Name	Update registers
ID	72
Biskit to Update	Source #2 biskitCreated
Temporary Biskit	Save to Database

Properties To Set

Set int2 to Source #4	insideWorkingHours
Set int1 to Source #4	outsideWorkingHours

Action Tab	Description
Trigger Type	The type of the <i>Workflow Action</i> .
Name	The user defined name for this action.
ID	The ID number for this action.
Action Specific Information	Defined by the type of <i>Action</i> . For full details for each type see the different <i>Action</i> sections below.

Action	Conditions	Control	Comments	Output	References
Enabled		true			
Children to Run		All children			
Record System Events		Record all system events			
Sort Order		0			
Completion Action		Continue			
On Error		Abort			
Last Ran		26 Sep 2018 13:42			

Permissions

Permission User Roles	Admin, User, Guest
Permission User Type	No user type assigned for permissions

Control Tab	Description
Enabled	Whether this <i>Action</i> will run. Disabled <i>Actions</i> will be displayed in red.
Children To Run	Whether to run only the first valid child <i>Workflow Action</i> or all of the valid child <i>Workflow Actions</i> . The order to check for a valid child <i>Workflow Actions</i> is determined by the Sort Order of the <i>Actions</i> .
Record System Events	Choose whether to record system events created by this <i>Action</i> and its children.
Sort Order	This determines in which order the <i>Workflow Actions</i> are run where an <i>Event</i> or <i>Action</i> has multiple children. The higher the Sort Order the later the child is run.
Completion Action	What to do when the action completes. Either Continue , Last Child or Stop .
On Error	What to do if an error occurs, Abort or Continue . Useful if parsing dates or strings, and the Workflow needs to continue even if in the wrong format.
Last Ran	When this <i>Workflow Action</i> was last run.
Permissions	Defines the <i>Permissions</i> this <i>Action</i> will run with. These can be applied by User Role ⁶⁵⁶ or User Type ⁶⁵⁶ .

Action Tab

When accessing the **Action** Tab, at least one of the options available will be switch-able from **Fixed** to **Variable**.

Fixed allows the user to specify a particular item is to be used, if that item could be a *Biskit* then the user may be asked to input the *Biskit Def*, and then provide the individual *Biskit* to be used.

Variable allows the user to pick an object that is output from one of the previous **Actions** or the **Event** from which these **Actions** stem or the **Meta-Properties** available.

Meta-Properties

The **Meta-Property** option allows access to data from the Work Flow as a whole. Certain **Events** will not have data available for all **Meta-Properties**, for example **Timed Event** will only have values in the date **Meta-Property**.

Meta-Property	Description
dataType	The dataType the Work Flow is running under if running under a data type.
date	The date/time the workflow was run.
sessionID	The sessionID of the current user's sessions, if running under a user's session.
user	The current user information if one is available.

6.13.3.1 Biskit Create Workflow Action

Creates a new [Biskit](#)⁶⁵² from the supplied [BiskitDef](#)⁶⁵².

The *Biskit* that is created can be saved to the database or could be a temporary *Biskit* that is only available during the running of this [Workflow](#)⁶⁵⁶. Use temporary *Biskits* to store information that needs to be used by a sibling flow that will be run later, or to convert a *Biskit* to be a [property](#)⁶⁵⁵ of a *Biskit* for such things as a [Property Path](#)⁶⁵⁵ on an email recipient list.

If the *Biskit* to be created is from a *BiskitDef* which is defined on an already created *Biskit* then the new one may be created with the default initial *property* values or may have the values copied from the original *Biskit*.

Finally any of the *properties* on the *Biskit* may be specifically set by the user on creation.

Biskit Type to Create	Variable ▼	Source #10 ▼	biskitCreated.biskitType
Temporary Biskit	Save to database ▼		
Initial Property Values	Copy Biskit Values ▼		

Properties To Set
 There are no entries here
 Add ➕

Returns **biskitCreated** for use with subsequent [Actions](#)⁶⁵⁶ which holds all the properties of the created *Biskit*.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
biskitCreated	Information about the created <i>Biskit</i> .

6.13.3.2 Biskit Delete Workflow Action

Deletes the supplied [Biskit](#)⁶⁵².

Biskit	Fixed ▼	Location	ANC Building
--------	---------	----------	--------------

Returns **biskitAffected** for use with subsequent [Actions](#)⁶⁵⁶ which holds all the properties of the deleted *Biskit*.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
biskitAffected	Information about the deleted <i>Biskit</i> .

6.13.3.3 Biskit Update Workflow Action

Updates the supplied [Biskit](#)⁶⁵² with the list of [properties](#)⁶⁵⁵ to be set and whether the save is temporary or not.

Returns **biskitAffected** for use with subsequent [Actions](#)⁶⁵⁶ which holds all the properties of the updated *Biskit*.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
biskitAffected	Information about the updated <i>Biskit</i> .

6.13.3.4 Create Variables Workflow Action

Creates temporary variables for use by the *Workflow* with the list of [properties](#)⁶⁵⁵ to be created.

Variables of types **Biskit, Boolean, Date, Date Range, Datetime, Double, Int, JavaEnum, List, Long, Set, String, StringEnum**.

Returns **variables** for use with subsequent [Actions](#)⁶⁵⁶ which holds all the variables that have been created.

There is also the option using the edit icon to add a description for each variable.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
changes	Information about changes on the <i>Biskit</i>

6.13.3.5 Delay Workflow Action

Delays the running of the [Workflow](#)⁶⁵⁶ by the number of units set. Units can be anything from seconds to years, including *Business Days* (See [Workflow Events](#)⁶⁴⁰ for a definition of Business Days).

Delay Units	weeks ▼
Delay Number	7

Returns nothing for use with subsequent [Actions](#)⁶⁵⁶.

Properties Available for Child Actions To Use

There are no additional properties available for use in any child or descendant action beyond the standard set.

6.13.3.6 Diff Workflow Action

Returns the difference between two [Biskits](#)⁶⁵² as a text string defining those differences.

First item to compare	Variable ▼	Source #1 ▼	New.supervisor
Second item to compare	Variable ▼	Source #1 ▼	Old.supervisor

Returns **diff** for use with subsequent [Actions](#)⁶⁵⁶ which holds all the differences found.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
diff	The differences found

6.13.3.7 Email Workflow Action

Sets up an email to be sent.

The Recipients Tab

Returns **emailBody**, **emailSubject**, **errorMessage**, **numberOfRecipients** and **users** for use with subsequent [Actions](#)⁶⁵⁶ which holds information about the email.

Individual Users

This lets you specify particular users that should receive the email.

User Types

If you specify one or more *User Types*, then any user with one of those *User Types* will receive the email.

User Groups

Any user in the specified *groups* will receive the email.

Email Addresses

A list of email addresses who will receive the email.

User Roles

Users can have multiple [roles](#)⁶⁵⁶. Select any number of *roles* by ticking the check box next to the *role*, and then choose with the drop-down whether the email is to go to users that have all of the selected *roles*, or users that have any of the selected *roles*.

Property Path

As an abstract concept, the [property path](#)⁶⁵⁵ may at first appear confusing. It is used when the data that is accessed contains a [reference](#)⁶⁵⁵ to a **User**, *User Group* or *User Type*, or else contains a *reference* to something that contains a *reference* to a **User**, *User Group* or *User Type*.

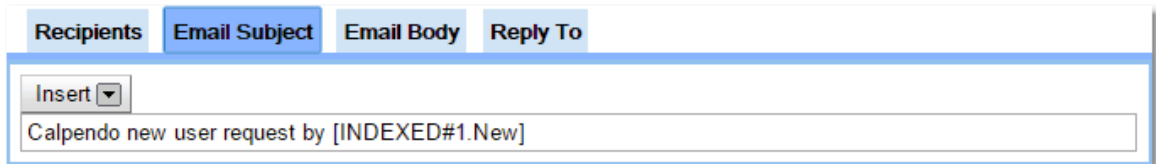
Send To User Performing Action

An **Email Action** is triggered as part of a **Workflow**. The **Workflow** could be triggered by something a particular user did. If that user is required to receive the email, then use [Meta-property](#)⁶⁵⁴ user.

The Email Subject Tab

Insert

This allows the user to insert data into the subject line from any previously run [Workflow Actions](#)⁶⁵⁶.



The screenshot shows a configuration window with four tabs: 'Recipients', 'Email Subject', 'Email Body', and 'Reply To'. The 'Email Subject' tab is active. Inside this tab, there is an 'Insert' button with a dropdown arrow. Below the button is a text input field containing the text 'Calpendo new user request by [INDEXED#1.New]'. The text '[INDEXED#1.New]' is highlighted in red, indicating it is a dynamic insert point.

The **Insert** button that appears in the tab enables dynamic text to be inserted that will be replaced when the email is generated. For example, suppose the email subject is required to be **New Calpendo user request by X** where X should be replaced by the login name of the new user. Use the Insert button to provide all the possible information that could be inserted gathered from previously run *Workflow Actions*.

New And Old Data

This means that they allow the insertion of text from the object that was created, updated or deleted. In the case of an update triggering the email, then the **New** option is linked to the *properties* on the object after the change has taken place and the **Old** option is linked to the *properties* on the object before the change took place.

For date *properties* extract just the date or just the time to give more control over the content of the subject.

When the action triggering the email is either **Create** or **Delete**, then the text inserted by the **New and Old** options will generate identical results.

Metaproperties Date

This inserts the text **[DATE]** and when the email is generated, this inserts the date and time that the email is generated.

The AuditLog Data

Whenever data is created, updated or deleted, there is an **AuditLog** entry created. This option allows the extraction of information from the audit log entry. This is not normally useful, but is required for a special case. When a user registers, there is normally an **Email Action** that lets an administrator know that a user has just registered. That email contains two **URLs** - one that will automatically approve the user, and one that will automatically deny the user. For this to work, **Calpendo** needs to make sure that when the relevant **URLs** are accessed, nobody else has already changed the user's *properties*. So, the **Email Action** uses the audit log information to add the **AuditLog's** identifier. This is a unique identifier for the audit log entry. **Calpendo** can then examine the audit log and check to see if there have been any changes to the user since the time the email was generated; only if there have been no subsequent changes will the user be approved or denied.

The Email Body Tab

Recipients Email Subject **Email Body** Reply To

Insert ▾ ALL URL

Given name: [INDEXED#1.New.givenName]
 Other name: [INDEXED#1.New.otherName]
 Family name: [INDEXED#1.New.familyName]
 Login name: [INDEXED#1.New.userIdentity.loginName]
 Email: [INDEXED#1.New.email]

Click here to accept request:
 [URL]#nurs&status=NORMAL&user=[INDEXED#1.New.id]&log=[INDEXED#1.AuditLog.id]

The body of the email works just like the subject does. The only differences are that there are extra buttons, for additional **magic** text, and the editor allows multi-line text instead of single-line text.

The All Button

ALL generates output that shows every piece of data from all previously run *Workflow Actions*. The format this generates is not particularly user-friendly and can have a lot of data, and so if this is used, be careful about who receives such emails.

The URL Button

This inserts a **URL** to your **Calpendo**. Sometimes, it is required to provide a link in an email to a particular page. This can be done by putting the relevant page information after the main **URL**. For example, if a [Project](#)⁶⁵⁴ has been changed, then set up a link like this:
[URL]/#ba&type=Project&action=view&id=[NEW.id]

In this case, **[URL]** is replaced by the main **URL**, then **#ba&type=Project&action=view** links to a page that is told to display *Projects* and not other *Biskit types*. Finally, the **&id=[NEW.id]** specifies the identifier of the particular *Project* that was changed.

The easy way to work out how any URL like this should be formatted is to first go to the relevant page and see what URL **Calpendo** uses, then replicate that within the **Email Action**, replacing parts as appropriate with dynamic text.

It should be noted that it's not normally useful to insert a **URL** into an email's subject. However, the same **URL...** button is also used for the content of an email.

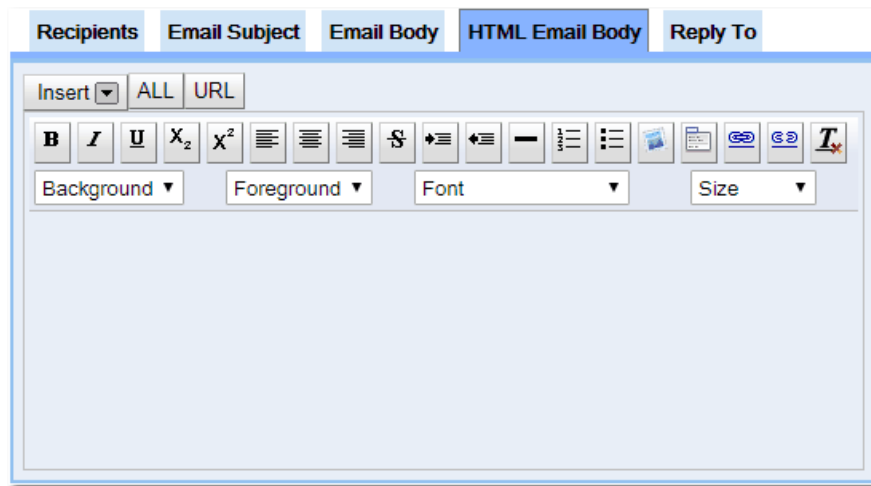
Attachments

When setting up an email in the **Workflow Manager**, if a [Biskit](#)⁶⁵² **Attachment** is referenced within the text, the UI will insert the attachment name into the text of the email body or subject. This will also cause the **Attachment** to be attached to the email as well.

If the path inserted into the email is modified so that instead of selecting the attachment name, the **Attachment** itself is selected.

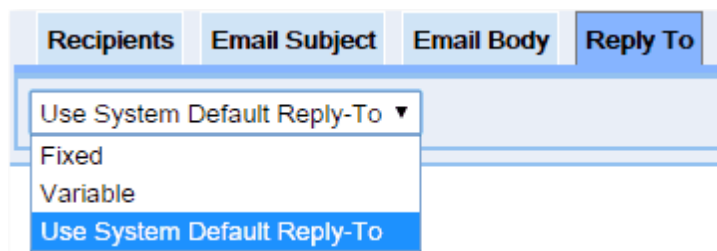
For example, selecting an **Attachment** when building the content of the email will insert a path similar to **index#1.attachment.name**. If the path is modified to **index#1.attachment**, the email still has the **Attachment** attached to it, but no text is added to the email's body or subject.

The HTML Email Body Tab



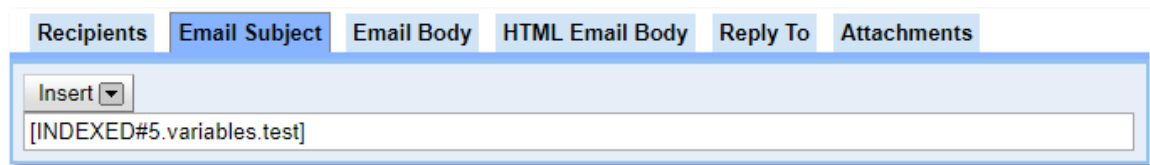
The HTML Email Body of the email works just like the Email Body does. The only differences are that there are extra options, to create the body using HTML rather than straight text.

The Email Reply To Tab



Allows the user to either use the system default for **Reply To** (Global Preferences), or insert their own **Reply To** address.

The Attachments Tab



The screenshot shows a user interface with several tabs: 'Recipients', 'Email Subject', 'Email Body', 'HTML Email Body', 'Reply To', and 'Attachments'. The 'Attachments' tab is currently selected. Below the tabs is a text input field with a dropdown menu labeled 'Insert'. The dropdown menu is open, showing a single option: '[INDEXED#5.variables.test]'.

Allows the user to either use the specify an attachment or list of attachments for the email..

Dealing With Sensitive Data In Emails

If there is data that any **user** shouldn't be able to put into an email because of confidentiality or sensitivity but the **user** still needs to be able to access the information in **Calpendo** then set up specific *Permissions* for data. There is a special user called *nobody* which is used for this purpose. In the **Permissions** page, for a *property* or *Biskit*, create a *Permission* with *Action* set to **Read**, **Authorisation** to **Deny Permission** and [Applies To](#)⁶⁵² to *nobody*, so that the *property* or *Biskit* can still be read by the user within **Calpendo** but cannot be added to an email. Read the [Permissions](#)³¹⁴ section for more details on setting *Permissions*.

If emailing data is a problem create a **Templated Text Workflow Action** (which does not check against *nobody*), to generate the text required and use that as input to the **Email Workflow Action**.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

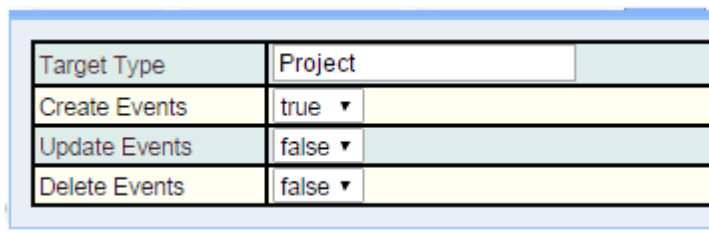
Properties	Notes
emailBody emailSubject emailMessage numberOfRecipients	<i>Properties</i> defining the email information.

6.13.3.7.1 Example Email Workflows

Example 1: Send Email To An Administrator When A Project Request Created

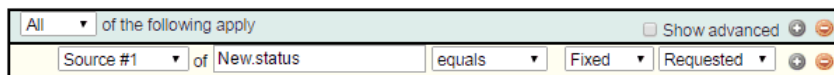
This first example shows how to send an email to an administrator when somebody creates a [Project](#)⁶⁵⁴ whose [status](#)⁶⁵⁴ is **Requested**.

1. First, go to the **Workflow Manager** page and press the **Create** button to create a new [Workflow](#)⁶⁵⁶.
2. Give it a name, like **Project request to admin**
3. Specify the Main Type as **Project**
4. Add a [Database Event](#)⁶⁴⁰, give it a **Name**, set the **Target Type** to be **Project** and **Create Events to True**.



Target Type	Project
Create Events	true ▼
Update Events	false ▼
Delete Events	false ▼

5. Add a [condition](#)⁶⁵³: **Status is Requested**



All ▼	of the following apply	<input type="checkbox"/> Show advanced
Source #1 ▼	of New.status	equals ▼ Fixed ▼ Requested ▼

6. Add an [Email Workflow Action](#)³⁷³, give it a **Name**.
7. Add the recipients you require
8. Enter the email's subject



Recipients	Email Subject	Email Body
Insert ▼ Calpendo new project request		

9. Specify the text of the email. This can reference *properties* of the project created and the user who created it.

The screenshot shows a window with three tabs: "Recipients", "Email Subject", and "Email Body". The "Email Body" tab is selected. Inside the tab, there is a toolbar with "Insert", "ALL", and "URL" buttons. Below the toolbar is a text area containing the text: "A new project request has been made by [INDEXED#1.New.owner.userIdentity]". The text "[INDEXED#1.New.owner.userIdentity]" is underlined in red, indicating a link or a specific data field.

Example 2: Send Email To A New User After They Register

This example shows how to create an **Email Workflow** that will be sent to a new user immediately after they register with **Calpendo**. We need to make sure it gets sent when a new **User** is created, and that the email gets sent to the new user.

1. Create a *Workflow* with a **Database Event**³⁴⁰ that is triggered when a **User** is **Created**, to do this give it a **Name**, set the **Target Type** to be **User** and **Create Events** to **True**.

Target Type	User
Create Events	true
Update Events	false
Delete Events	false

2. Add a *condition*: **Status** is **Requested**

All of the following apply
Source #1 New.status equals Requested

3. Add an **Email Workflow Action**³⁷³, give it a **Name**.
4. On the **Recipients** tab in **Property Paths**, set **Source #? UserMakingTheChange**.

Recipients	Email Subject	Email Body				
<div> <div>Individual Users</div> <div>User Types</div> <div>User Groups</div> <div>Email Addresses</div> <div>User Roles</div> <div>Property Paths</div> </div>	<div>Property Path To User(s), User Type(s), User Group(s) or email addresses</div> <div>Meta-property ▼ Select a property</div> <table border="1"> <thead> <tr> <th>Source</th> <th>Property Path</th> </tr> </thead> <tbody> <tr> <td>Source #1</td> <td>UserMakingTheChange</td> </tr> </tbody> </table> <div> <div>↑</div> <div>Delete</div> </div>		Source	Property Path	Source #1	UserMakingTheChange
Source	Property Path					
Source #1	UserMakingTheChange					

5. On the **Email Subject** tab, enter the text you want to be the subject of the email.
6. On the **Email Body** tab, enter the text you want to be the main content of the email.

Example 3: Send Email To A User When New User Request Denied

This example shows how to send an email to user who has just registered (so the **status** is set to **Requested**) but had their request denied (so the **status** has changed to **Denied**).

1. Create a *Workflow* with a [Database Event](#)³⁴⁰ that is triggered when a **User** is **Updated**, to do this give it a **Name**, set the **Target Type** to be **User** and **Update Events** to **True**.

Target Type	User
Create Events	false ▼
Update Events	true ▼
Delete Events	false ▼

2. Add a *condition*: **Old Value Status is Requested**.
Add a *condition*: **New Value Status is Denied**.

All ▼	of the following apply					Show advanced
Source #1 ▼	of	New.status	equals ▼	Fixed ▼	Denied ▼	
Source #1 ▼	of	Old.status	equals ▼	Fixed ▼	Requested ▼	

3. Add an [Email Workflow Action](#)³⁷³, give it a **Name**.
4. On the **Recipients** tab in **Property Paths**, set **Source #?** to be **New**;

Recipients	Email Subject	Email Body				
Individual Users User Types User Groups Email Addresses User Roles Property Paths	Property Path To User(s), User Type(s), User Group(s) or email addresses Meta-property ▼ Select a property <table border="1"> <thead> <tr> <th>Source</th> <th>Property Path</th> </tr> </thead> <tbody> <tr> <td>Source #1</td> <td>New</td> </tr> </tbody> </table> ↑ Delete		Source	Property Path	Source #1	New
Source	Property Path					
Source #1	New					

5. Enter the email's **Subject**.
6. Enter the email's main content

Example 4: Send An Email When A Booking Is Cancelled

This example shows how you can do quite complicated things by adding *conditions* to achieve the desired result. We're going to create an **Email Workflow** that will tell interested people when there's a cancellation on a particular *resource*⁶⁵⁵. The purpose of this email is to enable *resources* to be better utilised by filling in gaps in the *bookings*⁶⁵². However, if the *booking* being cancelled is too far into the future, there's little point telling people about it, so we will make sure the *booking* is no more than 21 days into the future. Also, if a *booking* has only just been made, then sending out an email is likely to be viewed as spam. So we will make sure the *booking* was created before 1 hour ago.

For this last requirement to work, we rely on the fact that **Calpendo** stores the date and time a *booking* was created on the **Booking**.

1. Create a *Workflow* with a **Database Event**³⁴⁰ that is triggered when a **Booking** is **Updated**, to do this give it a **Name**, set the **Target Type** to be **Booking** and **Update Events** to **True**.
2. Add a *condition*: **Old Value** not equal to **Cancelled**
 Add a *condition*: **New Value** equals **Cancelled**
 Add a *condition*: **New Value** of **dateRange.start** earlier than **now plus 21 days to the day**
 Add a *condition*: **Old Value** of **created** earlier than **now minus 1 hour with accuracy of a minute**

All of the following apply		Show advanced	
Source #1	of	Old.status	not equal to Fixed Cancelled
Source #1	of	New.status	equals Fixed Cancelled
Source #1	of	Old.dateRange.start	earlier than now plus 21 days to the day
Source #1	of	Old.created	earlier than now minus 1 hours to the minute

3. Add an **Email Workflow Action**³⁷³, give it a **Name**.
4. Add the recipients you require.
5. Enter the email's **Subject** and **Body**. Text should include the name of the *resource* and the time of the *booking*.

Recipients Email Subject Email Body

Insert [v]

Booking has been cancelled for [INDEXED#1.New.resource.name] on [INDEXED#1.Old.dateRange.start.date]

6. Enter the email's body

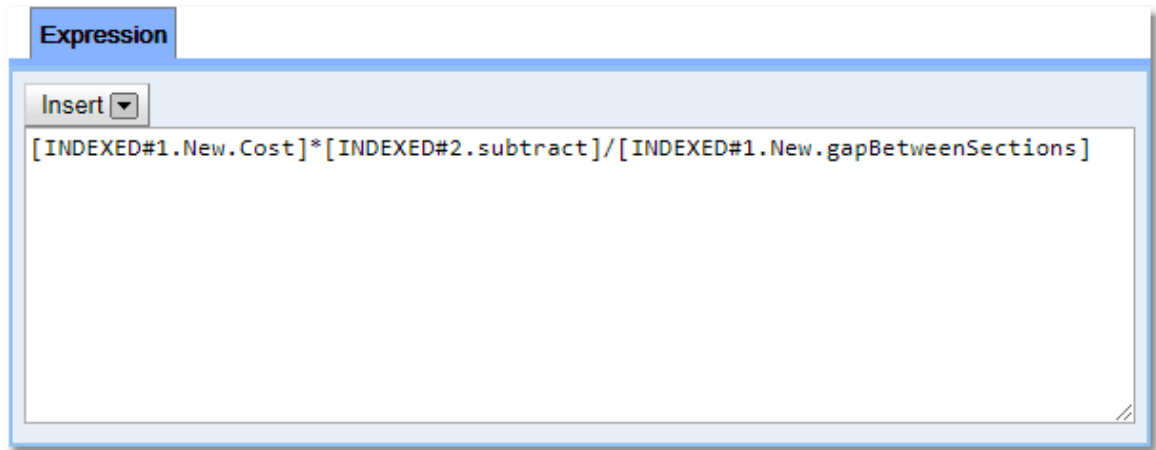
Recipients Email Subject Email Body

Insert [v] ALL URL

Booking cancelled by [INDEXED#1.UserMakingTheChange.self.userIdentity] at [INDEXED#1.New.cancelled]
 The booking was for [INDEXED#1.New.project.name] owned by [INDEXED#1.New.project.owner.userIdentity]

6.13.3.8 Evaluate Expression Workflow Action

Allows the user to type in an expression, including text, numbers and variables which will then be evaluated.



The screenshot shows a workflow action window titled 'Expression'. Inside, there is a text area containing the expression: `[INDEXED#1.New.Cost]*[INDEXED#2.subtract]/[INDEXED#1.New.gapBetweenSections]`. Above the text area is a button labeled 'Insert' with a dropdown arrow.

There is also support for logical combinations of booleans.

Therefore the following expressions can be used:

Logical AND:

a && b
a AND b
a and b

Logical OR:

a || b
a OR b
a or b

Logical Exclusive OR:

a ^ b
a XOR b
z xor b

Logical NOR:

a NOR b
a nor b

Returns **value** for use with subsequent [Actions](#)⁶⁵⁶ which holds all the value of the expression in the appropriate type (could be double, integer, text, long, date etc). Also included in value is a property valueSet which specifies whether a value was set.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
value	The value returned by the expression.


6.13.3.9 Execute System Command Workflow Action

Allows the user to run a system command. The user passes in the name of the executable, the expected exit value if succeeded, the directory it will be found, the timeout to be used, and can choose to define the names for Standard Output and Standard Error, or allow the system to generate random names, and any arguments that are required for the system executable.

Executable	Null ▼	
Expected Exit Value	Null ▼	
Working Directory	Null ▼	
Command Timeout	Infinite timeout ▼	
Standard Output File Name	Generate random name ▼	
Standard Error File Name	Generate random name ▼	

Command arguments

There are no entries here

Add 

Returns **exitValue** and **stderr**, **stdout** for use with subsequent [Actions](#) ⁶⁵⁶.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
exitValue	The value the system command gives on exit.
stderr	Any error output.
stdout	Any standard output.

6.13.3.10 Find Text Workflow Action

Checks the supplied **Input String** for the defined **Search** string, which can be a **Regular Expression**, or **Plain Text** using the **Flags**, or **Case sensitivity** to define the search, and either find up to **Number of Groups** sets of characters that match or replace the first or all occurrences found, by the **Replacement String**.

Using **Regular Expressions** to extract a number of **Groups** of characters from a string. This can be set up to extract **Once** or **Repeatedly**, the **Repeatedly** is useful for extracting information from a multi-line **csv** file and returns a list of [Biskits](#)⁶⁵² each holding the appropriate number of **Groups** for each line. See [Pattern Matching](#)¹¹² in [Conditions](#)¹⁰⁷ for more information on valid **Regular Expressions**.

Input String	Fixed ▾	
Search For	Fixed ▾	
Use Regular Expressions	Use regular expressions ▾	
Flags	No flags selected	
Mode	Extract text from input repeatedly ▾	
Number of Groups	Extract text from input once	
	Extract text from input repeatedly	
	Replace first occurrence in input	
	Replace all occurrences in input	

Using **Plain text** or **Regular Expressions** to find text in the input string and **Replace First occurrence** or **Replace all occurrences**.

Input String	Fixed ▾	
Search For	Fixed ▾	
Use Regular Expressions	Use plain text ▾	
Case sensitivity	Case insensitive ▾	
Mode	Replace all occurrences in input ▾	
Replacement String	Replace first occurrence in input	
	Replace all occurrences in input	

Returns whether the text **matches**, a list of *Biskits* called **Repeats**, a number of **groups#** of characters requested to be found and **outputString** (the string once replacements are made), for use with subsequent [Actions](#)⁶⁵⁶.

Capturing Groups

Capturing groups are numbered by counting their opening parentheses from left to right. In the expression ((A)(B(C))), for example, there are four such groups:

- 1 ((A) (B (C)))
- 2 (A)
- 3 (B (C))
- 4 (C)

Group zero always stands for the entire expression.

Capturing groups are so named because, during a match, each subsequence of the input sequence that matches such a group is saved.

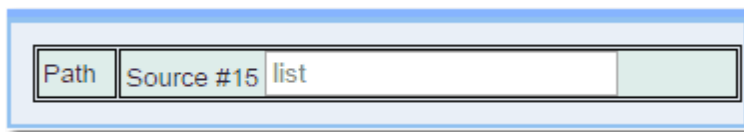
Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
matches	Whether the expression matches or not.
group#	Each text grouping that has been found.
outputString	The changed string when replace used.

6.13.3.11 For Each Workflow Action

Takes as input a list or set supplied by another [Workflow Action](#)⁶⁵⁶ and runs its child [Actions](#)⁶⁵⁶ once for each item in the list.



Returns **biskit**, which holds information about the Biskit pulled from the list and **index** which is the position in the list of the [Biskit](#)⁶⁵² for use with subsequent [Actions](#).

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
biskit	The <i>biskit</i> pulled from the list. This will be Null if a non- <i>biskit</i> list

Properties	Notes
	is involved.
index	Where in the list the data is.
value	The data pulled from the list. This will also hold the <i>biskit</i> information if its a list of <i>biskits</i> .

6.13.3.12 Function Workflow Action

Function [Workflow Actions](#)⁶⁵⁶ take input from previous *Actions*, and implement the requested **Function** on the input(s) returning data which can be used by another *Action*.

There are nine types of **Functions**.

Function Type	Description
Biskit	Works on a Biskit ⁶⁵² , such as adding or removing from a Group
Calendar	Used to provide Calendar information.
Conversion	Convert from one format to another.
Date & Time	Do calculations with time and dates.
File	Provide functions to deal with files
List/Set	Provide functions to work with Lists/Sets .
Logical	Provide functions to implement logical operands.
Mathematical	Provide standard Mathematical functions.
Miscellaneous	Those that do not fit elsewhere
Network	Provide TCP functions.
Text	To do simple text functions.

6.13.3.12.1 Biskit Function Type

Function	Description
addToGroup	Adds an object to its appropriate Group .
biskitDef	Returns the <i>BiskitDef</i> for the type of a given <i>Biskit</i> .
copy	Copies the properties from one <i>Biskit</i> to another. May return the new <i>Biskit</i> .
enumerateBiskitProperties	Returns a list of Biskits ⁶⁵² each of which describe a property in the <i>Biskit</i> .
expandRepeats	Expands the repeats for a <i>Biskit</i>
find	Returns a list of Biskits ⁶⁵² of a particular BiskitDef ⁶⁵² with the input id's (DB primary key).
formEncode	Returns a form encoded version of the input <i>Biskit</i> .
getAsBiskit	Returns the value of any <i>property</i> as a <i>Biskit</i> .
getAsBiskitList	Returns a property of a given name from a biskit, and returns its value as a list of biskits.
getAsBoolean	Returns the value of any property ⁶⁵⁵ as a Boolean.
getAsDate	Returns the value of any <i>property</i> as a Date.
getAsDateList	Returns the values of a property as a list of Dates. Only works on list or set properties.
getAsDateTime	Returns the value of any <i>property</i> as a DateTime.
getAsDouble	Returns the value of any <i>property</i> as a Double.
getAsDoubleList	Returns the values of a property as a list of Doubles. Only works on list or set properties.
getAsInt	Returns the value of any <i>property</i> as a Int.
getAsIntList	Returns the values of a property as a list of Integers. Only works on list or set properties.
getAsLong	Returns the value of any <i>property</i> as a Long.
getAsLongList	Returns the values of a property as a list of Longs. Only works on list or set properties.
getAsString	Returns the value of any <i>property</i> as a String.
getAsStringList	Returns the values of a property as a list of Strings. Only works on list or set properties.
hasProperty	Returns True if the property exists on the <i>Biskit</i> .
identity	Returns the <i>Biskit</i> passed in unchanged.
mapBiskitProperties	Returns a new <i>Biskit</i> as a copy of a given <i>Biskit</i> .
printBiskit	Returns a formatted version of the <i>Biskit</i> as a String
propertyDef	Returns a <i>PropertyDef</i> from a <i>BiskitDef</i> and a path to the <i>property</i> .
removeFromGroup	Removes a User from a User Group .
setProperty	Sets the value of a <i>property</i> .

addToGroup

The user provides the **Group** of the appropriate type and the **User, Project** or **Resource** to be added to that **Group**.. Nothing is returned.

biskitDef

The user provides the *Biskit* or type of *Biskit*. Returned is the *BiskitDef*.

copy

To copy a *Biskit* the user provides the *Biskit* to be copied and the *Biskit* to be updated. Nothing is returned.

The alternative is to create a *Biskit* of a specific type *BiskitDef* and optionally save it to the database. This returns the *Biskit* that was created.

preferUSDateFormat: true if US formatted are preferred when trying to parse any date property and false if European formatted dates preferred.

format: the Date and Time pattern to be used when converting a property from a string.

Note: If the workflow copies to a pre-existing biskit that has previously been saved to the database, then the copy will also cause an update to be sent to the database. If, on the other hand, the workflow either copies to a pre-existing biskit that has never been saved to the database, or doesn't copy to a pre-existing biskit, then the copy is not saved to the database.

enumerateBiskitProperties

Function	enumerateBiskitProperties
Arguments	<div> <div>(List:Biskit:BiskitPropertyBiskit; value)enumerateBiskitProperties(Biskit.; biskit,Enum:PropertyType; type) ▼</div> <div>(List:Biskit:BiskitPropertyBiskit; value)enumerateBiskitProperties(Biskit.; biskit)</div> <div>(List:Biskit:BiskitPropertyBiskit; value)enumerateBiskitProperties(Biskit.; biskit,Enum:PropertyType; type)</div> </div>
Biskit.; biskit	Null ▼
Enum:PropertyType; type	Null ▼

The user provides the *Biskit* to be searched and optionally the *Property Type* to be found. Returned is a list of *Biskits* that match the search each of which describes the properties in a given biskit.

expandRepeats

Function	expandRepeats
Arguments	<div>20 functions</div> <div>(List:Biskit.typeOf:repeatingBiskit; list)expandRepeats(Biskit:Calpendo.Booking; repeatingBiskit.Date from,Date to)</div>
Biskit:Calpendo.Booking; repeatingBiskit	Null ▼
Date from	Null ▼
Date to	Null ▼

The user provides the *Biskit* to be expanded and the dates defining the period to be expanded over, or the start date and the number of days to be included in the expansion or null to use the standard as set in Global Preferences. Returned is the list of expanded repeats, or a list of just the original if the original did not repeat

find

Function	find
Arguments	<div>(List:Biskit.typeOf:bd; biskits)find(Biskit:BiskitDef; bd,List:Int primaryKeys) ▼</div>
Biskit:BiskitDef; bd	Null ▼
List:Int primaryKeys	Null ▼

The user provides the *BiskitDef* to be searched and the list of database ids (primary keys) that will be used for the search. Returned is a list of *Biskits* that match the search.

formEncode

Function	formEncode (String formEncoded)formEncode(Biskit:: biskit) ▼		
Arguments	<table border="1"> <tr> <td>Biskit:: biskit</td> <td>Null ▼</td> </tr> </table>	Biskit:: biskit	Null ▼
Biskit:: biskit	Null ▼		

The user provides the *Biskit* to be encoded. Returned is a form encoded version of the *Biskit*.

Encodes the *properties* of a *Biskit* in a suitable format for use as the body of a response to an **HTTP** event that expects a form **POST**. Ignored are all properties that are of type **Set**, **List**, **User defined** or **DateRange**

getAsBiskit

Function	getAsBiskit (Biskit:: biskit)getAsBiskit(Biskit:: biskit,String propName) ▼ (Biskit:: biskit)getAsBiskit(Biskit:: biskit,String propName) (Biskit::typeOf:bd; biskit)getAsBiskit(Biskit:: biskit,String propName,Biskit:BiskitDef; bd)				
Argument	<table border="1"> <tr> <td>Biskit:: biskit</td> <td>Null ▼</td> </tr> <tr> <td>String propName</td> <td>Null ▼</td> </tr> </table>	Biskit:: biskit	Null ▼	String propName	Null ▼
Biskit:: biskit	Null ▼				
String propName	Null ▼				

The user provides the *Biskit* and the name of the *property* whose value is to be returned. Returned is the value of the *property* as a **Biskit**, if it is a *Biskit*, and null otherwise. The user can optionally specify the expected *BiskitDef*, if the *Biskit* is not of this type then null is returned.

getAsBiskitList

Function	getAsBiskitList (List:Biskit:: value)getAsBiskitList(Biskit:: biskit,String propName) ▼ (List:Biskit:: value)getAsBiskitList(Biskit:: biskit,String propName) (List:Biskit::typeOf:bd; value)getAsBiskitList(Biskit:: biskit,String propName,Biskit:BiskitDef; bd)				
Argument	<table border="1"> <tr> <td>Biskit:: biskit</td> <td>Null ▼</td> </tr> <tr> <td>String propName</td> <td>Null ▼</td> </tr> </table>	Biskit:: biskit	Null ▼	String propName	Null ▼
Biskit:: biskit	Null ▼				
String propName	Null ▼				

The user provides the *Biskit* and the name of the *property* whose value is to be returned. Returns a property of a given name from a *Biskit*, and returns its value as a list of *Biskits*. If the original property is not a list or set property, then return null. Otherwise, check if each element in the source collection is a **Biskit**. For each element, if it is a *Biskit* its value is used, otherwise null is used in its place.

getAsBoolean

Function	getAsBoolean	(Boolean boolean)getAsBoolean(Biskit:: biskit,String propName) ▼
Arguments		
Biskit:: biskit	Null ▼	
String propName	Null ▼	

The user provides the *Biskit* and the name of the *property* (**Boolean**, **Int**, **Long** or **Double**) whose value is to be returned. Returned is the value of the *property* as a **Boolean**, if such conversion is possible, and null otherwise. Numeric properties are converted with zero meaning false and anything else non-null meaning true.

getAsDate

Function	getAsDate	(Date date)getAsDate(Biskit:: biskit,String propName,Boolean preferUSDateFormat) ▼
Arguments		
Biskit:: biskit		
String propName	Null ▼	
Boolean preferUSDateFormat	Null ▼	

The user provides the *Biskit* and the name of the *property* (**Date** or **DateTime**) whose value is to be returned. Returned is the value of the *property* as a **Date**, if such conversion is possible (will attempt to convert from Long and String values), and null otherwise.

preferUSDateFormat: true if US formatted are preferred when trying to parse the date and false if European formatted dates preferred.

format: the Date and Time pattern to be used when converting from a string.

getAsDateList

Function	getAsDateList
Arguments	(List:DateTime value)getAsDateList(Biskit::; biskit,String propName,Boolean preferUSDateFormat) ▼ (List:Date value)getAsDateList(Biskit::; biskit,String propName) (List:Date value)getAsDateList(Biskit::; biskit,String propName,Boolean preferUSDateFormat) (List:Date value)getAsDateList(Biskit::; biskit,String propName,String format) (List:DateTime value)getAsDateList(Biskit::; biskit,String propName) (List:DateTime value)getAsDateList(Biskit::; biskit,String propName,Boolean preferUSDateFormat) (List:DateTime value)getAsDateList(Biskit::; biskit,String propName,String format)

The user provides the *Biskit* and the name of the *property*. Returns the value of the *property* as a list of **Date** or **DateTime**, if such conversion is possible (will attempt to convert from Long and String values), and null otherwise. This only works on List or Set properties other types cause a null to be returned.

preferUSDateFormat: true if US formatted are preferred when trying to parse the date and false if European formatted dates preferred.

format: the Date and Time pattern to be used when converting from a string.

getAsDateTime

Function	getAsDateTime
Argument	(DateTime datetime)getAsDateTime(Biskit::; biskit,String propName) ▼ (DateTime datetime)getAsDateTime(Biskit::; biskit,String propName) (DateTime datetime)getAsDateTime(Biskit::; biskit,String propName,Boolean preferUSDateFormat) (DateTime datetime)getAsDateTime(Biskit::; biskit,String propName,String format)
Biskit::; biskit	
String propName	Null ▼

The user provides the *Biskit* and the name of the *property* (**Date** or **DateTime**) whose value is to be returned. Returned is the value of the *property* as a **DateTime**, if such conversion is possible (will attempt to convert from Long and String values), and null otherwise.

preferUSDateFormat: true if US formatted are preferred when trying to parse the date and false if European formatted dates preferred.

format: the Date and Time pattern to be used when converting from a string.

getAsDouble

Function	getAsDouble	(Double double)getAsDouble(Biskit:: biskit,String propName) ▼
Arguments		
Biskit:: biskit	Null ▼	
String propName	Null ▼	

The user provides the *Biskit* and the name of the *property* (**Int**, **Long** or **Double**) whose value is to be returned. Returned is the value of the *property* as a **Double**, if such conversion is possible, and null otherwise.

getAsDoubleList

Function	getAsDoubleList	(List:Double value)getAsDoubleList(Biskit:: biskit,String propName) ▼
Arguments		
Biskit:: biskit	Null ▼	
String propName	Null ▼	

The user provides the *Biskit* and the name of the *property* (**Int**, **Long** or **Double**) whose value is to be returned. Returned is list of values of the *property* as a **Double**, if such conversion is possible, and null otherwise. This only works on List or Set properties other types cause a null to be returned.

getAsInt

Function	getAsInt (Int int)getAsInt(Biskit:: biskit,String propName) ▼
Arguments	
Biskit:: biskit	Null ▼
String propName	Null ▼

The user provides the *Biskit* and the name of the *property* (**Int**, **Long** or **Double**) whose value is to be returned. Returned is the value of the *property* as an **Int**, if such conversion is possible, and null otherwise.

Long and **Double** values will be used as-is (truncated to an integer if required) if they are within the standard integer range. Otherwise they will be returned as either the maximum integer value or minimum integer value, as appropriate. The minimum and maximum integer values are -2,147,483,648 and 2,147,483,647 respectively. That is, -2^{31} and $2^{31} - 1$.

getAsIntList

Function	getAsIntList (List: Int value)getAsIntList(Biskit:: biskit,String propName) ▼
Arguments	
Biskit:: biskit	Null ▼
String propName	Null ▼

The user provides the *Biskit* and the name of the *property* (**Int**, **Long** or **Double**) whose value is to be returned. Returned is list of values of the *property* as an **Int**, if such conversion is possible, and null otherwise. This only works on List or Set properties other types cause a null to be returned

Long and **Double** values will be used as-is (truncated to an integer if required) if they are within the standard integer range. Otherwise they will be returned as either the maximum integer value or minimum integer value, as appropriate. The minimum and maximum integer values are -2,147,483,648 and 2,147,483,647 respectively. That is, -2^{31} and $2^{31} - 1$.

getAsLong

Function	getAsLong (Long long)getAsLong(Biskit:: biskit,String propName) ▼
Arguments	
Biskit:: biskit	Null ▼
String propName	Null ▼

The user provides the *Biskit* and the name of the *property* (**Int**, **Long** or **Double**) whose value is to be returned. Returned is the value of the *property* as a **Long**, if such conversion is possible, and null otherwise.

Integer and **Double** values will be used as-is (truncated to an integer if required) if they are within the standard integer range. Otherwise they will be returned as either the maximum long integer value or minimum long integer value, as appropriate. The minimum and maximum integer values are -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807 respectively. That is, -2^{63} and $2^{63} - 1$.

getAsLongList

Function	getAsLongList (List:Long value)getAsLongList(Biskit:: biskit,String propName) ▼
Arguments	
Biskit:: biskit	Null ▼
String propName	Null ▼

The user provides the *Biskit* and the name of the *property* (**Int**, **Long** or **Double**) whose value is to be returned. Returned is list of values of the *property* as an **Long**, if such conversion is possible, and null otherwise. This only works on List or Set properties other types cause a null to be returned

Integer and **Double** values will be used as-is (truncated to an long if required) if they are within the standard long integer range. Otherwise they will be returned as either the maximum long value or minimum long value, as appropriate. The minimum and maximum long integer values are -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807 respectively. That is, -2^{63} and $2^{63} - 1$

getAsString

Function	getAsString (String value)getAsString(Biskit.; biskit,String propName) ▼
Arguments	
Biskit.; biskit	Null ▼
String propName	Null ▼

The user provides the *Biskit* and the name of the *property* whose value is to be returned. Returned is the value of the *property* as a **String**.

getAsStringList

Function	getAsStringList (List:String value)getAsStringList(Biskit.; biskit,String propName) ▼
Arguments	
Biskit.; biskit	Null ▼
String propName	Null ▼

The user provides the *Biskit* and the name of the *property* whose value is to be returned. Returned is list of values of the *property* as a **String**. This only works on List or Set properties other types cause a null to be returned

hasProperty

Function	hasProperty (Boolean hasProperty)hasProperty(Biskit.; biskit,String propName) ▼
Arguments	
Biskit.; biskit	Null ▼
String propName	Null ▼

The user provides the *Biskit* and the name of the *property*. Returned is **True** if the *property* exists on the *Biskit* even if its value is Null.

identity

Function	mapBiskitProperties	(Biskit:: biskit)mapBiskitProperties(Biskit:: biskit,String prefixRemove,String prefixAdd,String suffixRemove,String suffixAdd) ▼
Arguments		
Biskit:: biskit	Null ▼	
String prefixRemove	Null ▼	
String prefixAdd	Null ▼	
String suffixRemove	Null ▼	
String suffixAdd	Null ▼	

The user provides the *Biskit*. Returned is the *Biskit*.

mapBiskitProperties

Function	mapBiskitProperties	(Biskit:: biskit)mapBiskitProperties(Biskit:: biskit,String prefixRemove,String prefixAdd,String suffixRemove,String suffixAdd) ▼
Arguments		
Biskit:: biskit	Null ▼	
String prefixRemove	Null ▼	
String prefixAdd	Null ▼	
String suffixRemove	Null ▼	
String suffixAdd	Null ▼	

The user provides the *Biskit*. Returned is the *Biskit*.

The user provides a prefix and suffix to add and to remove. The prefix/suffix is removed from the existing property names if they match the property names found, and then the prefix/suffix is prepended or appended as appropriate.

If any prefix or suffix is null, then it is treated the same as if it were the empty string.

For example, there is a *Biskit* with properties of names X_1, X_2 and X_3, then setting the removal prefix to X and the addition prefix to Y would result in a *Biskit* with properties Y_1, Y_2 and Y_3.

printBiskit

Function	printBiskit	(String printedBiskit)printBiskit(Biskit:; biskit,Boolean deep) ▾
Arguments		
Biskit:; biskit	Null ▾	
Boolean deep	Null ▾	

The user provides the *Biskit*, and whether to recursively output the properties on nested *Biskits*. Returned is the formatted *Biskit* as a string

propertyDef

Function	propertyDef	2 functions
		(Biskit:PropertyDef, pd)propertyDef(Biskit:BiskitDef; rootBD,String path) ▾
		(Biskit:PropertyDef, pd)propertyDef(Biskit:BiskitDef; rootBD,String path)
		(Biskit:PropertyDef, pd)propertyDef(String rootBiskitType,String path)
Arguments		
Biskit:BiskitDef; rootBD	Null ▾	
String path	Null ▾	

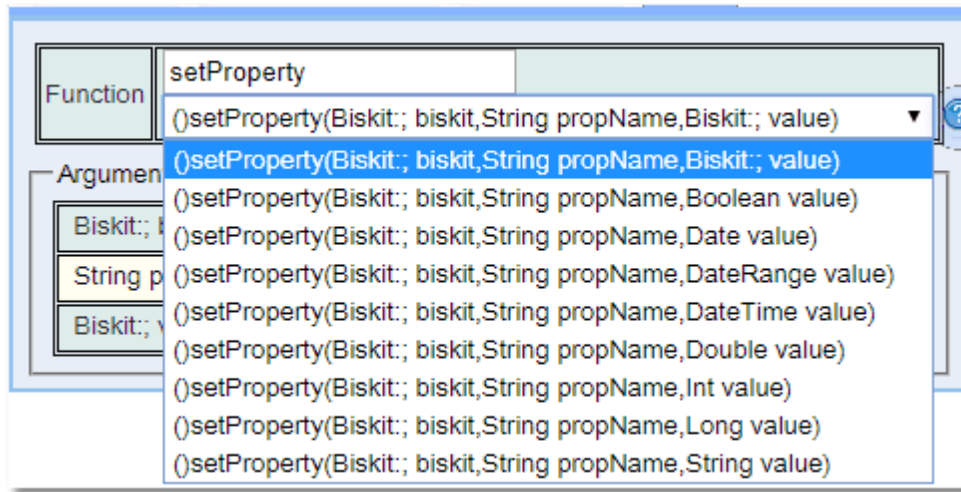
The user provides the *Biskit*, and a path to the *PropertyDef* required. Returned is the *PropertyDef*.

removeFromGroup

Function	removeFromGroup	
		()removeFromGroup(Biskit:Calpendo.ProjectGroup; projectGroup,Biskit:Calpendo.Project; project) ▾
		()removeFromGroup(Biskit:Calpendo.ProjectGroup; projectGroup,Biskit:Calpendo.Project; project)
		()removeFromGroup(Biskit:Calpendo.ResourceGroup; resourceGroup,Biskit:Calpendo.Resource; resource)
		()removeFromGroup(Biskit:UserGroup; userGroup,Biskit:ExprodoUser; user)
Argument		
Biskit:C		
Biskit:Calpendo.Project; project	Null ▾	

The user provides the **Group** of the appropriate type and the **User, Project or Resource** to be removed from that **Group**.. Nothing is returned.

setProperty



The user provides the *Biskit* and the name of the *property* whose value is to be updated as well as the new value. Nothing is returned.

6.13.3.12.2 Calendar Function Type

Function	Description
bookingsWeeklyView	Allows the displaying of a Calendar to a web page without the need to login. This is done by generating HTML representing bookings for several days across a number of resources.
createCalendarInvite	Creates an ICS file.
getBookableProjects	Finds the list of projects that a user is allowed to associate with a booking of a particular resource or list of resources.
getBookingTemplates	Finds the parts of a Template that covers any part of the booking or a particular time.
getUsableBookings	Gets a list of bookings that may be associated with a users use of a resource. Typically this would be associated with a ResourceUsage during a workflow that gets actual usage information.

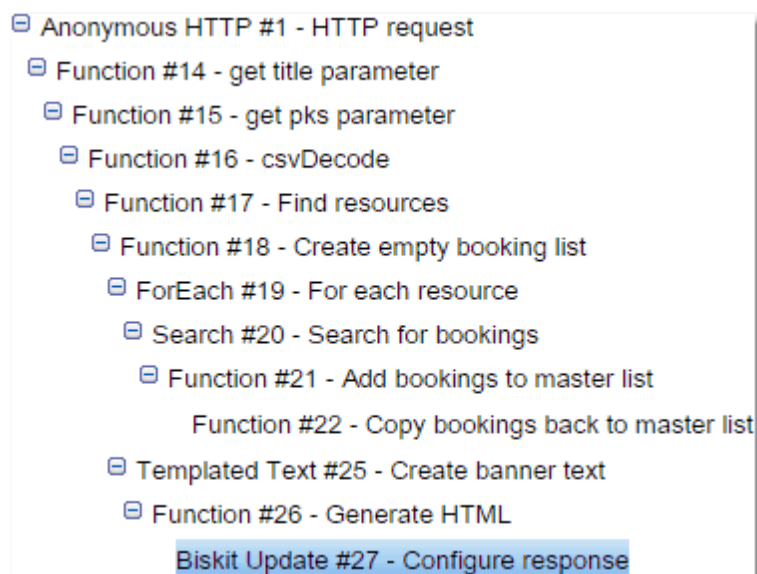
bookingsWeeklyView

Function	bookingsWeeklyView (String html)bookingsWeeklyView(List:Biskit:Calpendo.Booking; bookings,List:Biskit:Calpendo.Resource; resources,Biskit:Calpendo.CalpendoUser; permissionsUser,Biskit:Calpendo.CalpendoUser; styleUser,String title,String heading,Int numDays,Int startHour,Int endHour,Boolean missOutWeekends,Int resourceLabelHeight,Int bannerHeight,Int bookingTitleHeight,Int headingHeight)																												
Arguments	<table> <tr><td>List:Biskit:Calpendo.Booking; bookings</td><td>Null</td></tr> <tr><td>List:Biskit:Calpendo.Resource; resources</td><td>Null</td></tr> <tr><td>Biskit:Calpendo.CalpendoUser; permissionsUser</td><td>Null</td></tr> <tr><td>Biskit:Calpendo.CalpendoUser; styleUser</td><td>Null</td></tr> <tr><td>String title</td><td>Null</td></tr> <tr><td>String heading</td><td>Null</td></tr> <tr><td>Int numDays</td><td>Null</td></tr> <tr><td>Int startHour</td><td>Null</td></tr> <tr><td>Int endHour</td><td>Null</td></tr> <tr><td>Boolean missOutWeekends</td><td>Null</td></tr> <tr><td>Int resourceLabelHeight</td><td>Null</td></tr> <tr><td>Int bannerHeight</td><td>Null</td></tr> <tr><td>Int bookingTitleHeight</td><td>Null</td></tr> <tr><td>Int headingHeight</td><td>Null</td></tr> </table>	List:Biskit:Calpendo.Booking; bookings	Null	List:Biskit:Calpendo.Resource; resources	Null	Biskit:Calpendo.CalpendoUser; permissionsUser	Null	Biskit:Calpendo.CalpendoUser; styleUser	Null	String title	Null	String heading	Null	Int numDays	Null	Int startHour	Null	Int endHour	Null	Boolean missOutWeekends	Null	Int resourceLabelHeight	Null	Int bannerHeight	Null	Int bookingTitleHeight	Null	Int headingHeight	Null
List:Biskit:Calpendo.Booking; bookings	Null																												
List:Biskit:Calpendo.Resource; resources	Null																												
Biskit:Calpendo.CalpendoUser; permissionsUser	Null																												
Biskit:Calpendo.CalpendoUser; styleUser	Null																												
String title	Null																												
String heading	Null																												
Int numDays	Null																												
Int startHour	Null																												
Int endHour	Null																												
Boolean missOutWeekends	Null																												
Int resourceLabelHeight	Null																												
Int bannerHeight	Null																												
Int bookingTitleHeight	Null																												
Int headingHeight	Null																												

Parameter	Description
bookings	A list of the bookings ⁶⁵² to be displayed.
resource	A list of the resources ⁶⁵⁵ to be displayed.
permissionsUser	The User whose permissions ⁶⁵⁴ level will be used to decide what can be displayed.
styleUser	The User whose CSS style will be used for the Calendar display.
title	The string to be displayed on the web pages tab.
heading	The heading to be displayed above the Calendar .

Parameter	Description
numDays	How many days to display.
startHour	The hour the days display will start at.
endHour	The hour the days display will finish at.
missOutWeekends	Whether to display weekends or not.
resourceLabelHeight	The height of the label with the <i>resource</i> name at the top of the column.
bannerHeight	The height of the banner.
bookingTitleHeight	The height of the area displaying the <i>booking</i> title information.
headingHeight	The height of the header area above the Calendar .

Returns the **HTML** to display the **Calendar** as a web page.



Here is an example of the [Workflow](#)⁶⁵⁶ to create the information required to display the Calendar. **Function #26** is the **bookingsWeeklyView** function where all the information is collated and the **HTML** generated.

createCalendarInvite

Function	createCalendarInvite
	1 functions
	(Biskit.Attachment; calendarInvite)createCalendarInvite(Biskit:Calpendo Booking; booking Biskit:Calpendo.CalpendoUser; permissionsUser) ▼
Arguments	
Biskit:Calpendo.Booking; booking	Variable ▼ Source #156 ▼ biskit
Biskit:Calpendo.CalpendoUser; permissionsUser	Fixed ▼ admin (admin) ▼

Takes as input a *Biskit* of **Type Booking** and a user whose permissions are used to determine output. Returns the calendar invitation (ics file) as an attachment.

getBookableProjects

Function	getBookableProjects (List:Biskit:Calpendo.Project; projects,Int numberOfProjects)getBookableProjects(Biskit:Calpendo.CalpendoUser; user,List:Biskit:Calpendo.Resource; resources)
Argument	(List:Biskit:Calpendo.Project; projects,Int numberOfProjects)getBookableProjects(Biskit:Calpendo.CalpendoUser; user,Biskit:Calpendo.Resource; resource) (List:Biskit:Calpendo.Project; projects,Int numberOfProjects)getBookableProjects(Biskit:Calpendo.CalpendoUser; user,List:Biskit:Calpendo.Resource; resources)
Biskit:Calpendo.CalpendoUser; user	Null
List:Biskit:Calpendo.Resource; resources	Null

Takes as input the User and the *Resource* or a list of *Resources*. Returns a list of [Projects](#)⁶⁵⁴, and the number of *Projects* in the list.

getBookingTemplates

Function	getBookingTemplates 2 functions (List:Biskit:Calpendo.TemplateTAM; templateTAMs)getBookingTemplates(Biskit:Calpendo.Resource; resource,Biskit:Calpendo.CalpendoUser; booker,Biskit:Calpendo.Project; project,DateTime start,DateTime finish) (List:Biskit:Calpendo.TemplateTAM; templateTAMs)getBookingTemplates(Biskit:Calpendo.Booking; booking)
Arguments	(List:Biskit:Calpendo.TemplateTAM; templateTAMs)getBookingTemplates(Biskit:Calpendo.Resource; resource,Biskit:Calpendo.CalpendoUser; booker,Biskit:Calpendo.Project; project,DateTime start,DateTime finish) (List:Biskit:Calpendo.TemplateTAM; templateTAMs)getBookingTemplates(Biskit:Calpendo.Booking; booking)
Biskit:Calpendo.Resource; resource	Null
Biskit:Calpendo.CalpendoUser; booker	Null
Biskit:Calpendo.Project; project	Null
DateTime start	Null
DateTime finish	Null

Finds the parts of a [Time Template](#)⁶⁵⁶ that covers any part of a [Booking](#)⁶⁵² or a particular time period. Returned is one **TemplateTAM** for each *Time Template* that is found to cover part of the *Booking* or the particular time period. Each **TemplateTAM** stores a **Targeted user/project**, **Acceptability** and **Message** for a **Template**. Input is the *Booking* to check, or the time period, with the resource, project and booker, output is a list of the **TemplateTAM**'s covering the *Booking*.

getUsableBookings


Function	getUsableBookings (List:Biskit:Calpendo.Booking; bookings,Biskit:Calpendo.Booking; mostLikelyBooking,Int numberOfBookings)getUsableBookings(Biskit:Calpendo.CalpendoUser; user,Biskit:Calpendo.Resource; resource)
Arguments	Biskit:Calpendo.CalpendoUser; user Null Biskit:Calpendo.Resource; resource Null

Takes as input the User and the *Resource*. Returns a list of *Bookings* that could be associated with the **ResourceUsage**, the most likely *Booking* and how many *Bookings* returned in the list.

6.13.3.12.3 Conversion Function Type

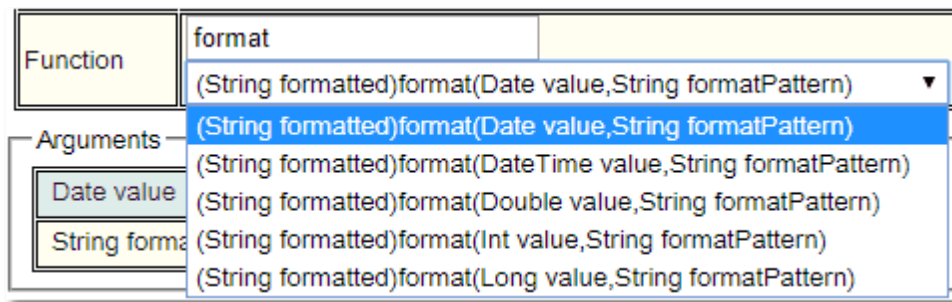
Function	Description
convertUnits	Converts a number from one unit to another.
format	Formats a value as a string.
fromJson	Converts a JSON formatted string to a Bisikit ⁶⁵²
prettyPrintJson	Takes JSON thats difficult to read and makes it easier to read.
toDate	Converts the input value to a value of type Date .
toDateTime	Converts the input into a DateTime .
toDouble	Converts the input to a Double .
toInt	Converts the input to an Integer .
toJson	Converts a biskit to JSON
toLong	Converts the input to a Long .
toPDF	Converts a string containing XSL FO to a PDF attachment and also to a temporary file.
toPDF_from_docx	Converts a Word document to a PDF document.
toPDF_from_odt	Converts an Open Office document to a PDF document.
toString	Converts the input into a String .

convertUnits

Trigger Type	Function Workflow Action						
Name	New						
ID	0						
Function	<div>convertUnits</div> <div>2 functions</div> <div>  (Double converted)convertUnits(Double number,Biskit:Unit; sourceUnit,Biskit:Unit; targetUnit) ▼ </div>						
— Arguments — <table border="1"> <tbody> <tr> <td>Double number</td> <td>Null ▼</td> </tr> <tr> <td>Biskit:Unit; sourceUnit</td> <td>Null ▼</td> </tr> <tr> <td>Biskit:Unit; targetUnit</td> <td>Null ▼</td> </tr> </tbody> </table>		Double number	Null ▼	Biskit:Unit; sourceUnit	Null ▼	Biskit:Unit; targetUnit	Null ▼
Double number	Null ▼						
Biskit:Unit; sourceUnit	Null ▼						
Biskit:Unit; targetUnit	Null ▼						

The user provides the number to be converted to a different unit, the current unit and the unit to be converted. The source and target units must be of the same type (eg both weights or both lengths etc). The output is the converted value.

format



The user the *property* to be formatted and the pattern to be used for the output. Returned is the formatted string.

The format used is of this general syntax for numeric types (int, long and double):

```
%[flags][width][.precision]conversion
```

The optional *flags* is a set of characters that modify the output format. The set of valid flags depends on *conversion*.

The optional *width* is a non-negative decimal integer indicating the minimum number of characters to be written to the output.

The optional *precision* is a non-negative decimal integer usually used to restrict the number of characters. The specific behavior depends on *conversion*.

The required *conversion* is a character indicating how the value should be formatted. The set of valid conversions depends on the value's data type.

The format used for date/time formatting is of this general syntax:

```
%[flags][width]conversion
```

The *optional* flags and *width* are defined as above.

The required *conversion* is a two character sequence. The first character is 't' or 'T'. The second character indicates the format to be used.

Conversions

Conversions are divided into the following categories:

- General - may be applied to any value type
- Numeric
 - ☐ Integer and Long
 - ☐ Double
- Date/Time - may be applied to types which are capable of encoding a date or time: Long, Date and Datetime.

- Percent - produces a literal '%'
- Line Separator - produces the platform-specific line separator

The following table summarizes the supported conversions. Conversions denoted by an upper-case character (i.e. 'B', 'H', 'S', 'C', 'X', 'E', 'G', 'A', and 'T') are the same as those for the corresponding lower-case conversion characters except that the result is converted to upper case according to the rules of the prevailing locale.

Conversion	Value Category	Description
'b', 'B'	general	If the value is null, then the result is "false". Otherwise, the result is "true".
'h', 'H'	general	If the value is null, then the result is "null". Otherwise, the result is obtained by converting the value's hash code to hexadecimal. For integer and long values, this means the original value is converted to hexadecimal.
's', 'S'	general	If the value is null, then the result is "null". Otherwise, the result is obtained by a simple conversion to a string.
'd'	Int, Long	The result is formatted as a decimal integer
'o'	Int, Long	The result is formatted as an octal integer
'x', 'X'	Int, Long	The result is formatted as a hexadecimal integer
'e', 'E'	Double	The result is formatted as a decimal number in computerized scientific notation
'f'	Double	The result is formatted as a decimal number
'g', 'G'	Double	The result is formatted using computerized scientific notation or decimal format, depending on the precision and the value after rounding.
'a', 'A'	Double	The result is formatted as a hexadecimal floating-point number with a significand and an exponent
't', 'T'	date/time	Prefix for date and time conversion characters. See Date/Time Conversions.
'%'	percent	The result is a literal '%'
'n'	line separator	The result is the platform-specific line separator

Any characters not explicitly defined as conversions are illegal and are reserved for future extensions.

Date/Time Conversions

The following two tables show the conversion suffix characters for date and time values using the 't' and 'T' conversions. First, conversion characters used for formatting times:

'H'	Hour of the day for the 24-hour clock, formatted as two digits with a leading zero as necessary i.e. 00 - 23.
'I'	Hour for the 12-hour clock, formatted as two digits with a leading zero as necessary, i.e. 01 - 12.
'k'	Hour of the day for the 24-hour clock, i.e. 0 - 23.
'l'	Hour for the 12-hour clock, i.e. 1 - 12.

'M'	Minute within the hour formatted as two digits with a leading zero as necessary, i.e. 00 - 59.
'S'	Seconds within the minute, formatted as two digits with a leading zero as necessary, i.e. 00 - 60 ("60" is a special value required to support leap seconds).
'L'	Millisecond within the second formatted as three digits with leading zeros as necessary, i.e. 000 - 999.
'N'	Nanosecond within the second, formatted as nine digits with leading zeros as necessary, i.e. 000000000 - 999999999.
'p'	Locale-specific morning or afternoon marker in lower case, e.g. "am" or "pm". Use of the conversion prefix 'T' forces this output to upper case.
'z'	RFC 822 style numeric time zone offset from GMT, e.g. -0800. This value will be adjusted as necessary for Daylight Saving Time. For Long, Date and Date-time, the time zone used is the default time zone for the server.
'Z'	A string representing the abbreviation for the time zone. This value will be adjusted as necessary for Daylight Saving Time. For Long, Date and Date-time, the time zone used is the default time zone for the server.
's'	Seconds since the beginning of the epoch starting at 1 January 1970 00:00:00 UTC.
'Q'	Milliseconds since the beginning of the epoch starting at 1 January 1970 00:00:00 UTC.

and these are the conversion characters used for formatting dates:

'B'	Locale-specific full month name, e.g. "January", "February".
'b'	Locale-specific abbreviated month name, e.g. "Jan", "Feb".
'h'	Same as 'b'.
'A'	Locale-specific full name of the day of the week, e.g. "Sunday", "Monday"
'a'	Locale-specific short name of the day of the week, e.g. "Sun", "Mon"
'C'	Four-digit year divided by 100, formatted as two digits with leading zero as necessary, i.e. 00 - 99
'Y'	Year, formatted as at least four digits with leading zeros as necessary, e.g. 0092 equals 92 CE for the Gregorian calendar.
'y'	Last two digits of the year, formatted with leading zeros as necessary, i.e. 00 - 99.
'J'	Day of year, formatted as three digits with leading zeros as necessary, e.g. 001 - 366 for the Gregorian calendar.
'm'	Month, formatted as two digits with leading zeros as necessary, i.e. 01 - 13.
'd'	Day of month, formatted as two digits with leading zeros as necessary, i.e. 01 - 31
'e'	Day of month, formatted as two digits, i.e. 1 - 31.

The following conversion characters are used for formatting common date/time compositions.

'R'	Time formatted for the 24-hour clock as "%tH:%tM"
'T'	Time formatted for the 24-hour clock as "%tH:%tM:%tS".

'r'	Time formatted for the 12-hour clock as "%tI:%tM:%tS %Tp". The location of the morning or afternoon marker ('%Tp') may be locale-dependent.
'D'	Date formatted as "%tm/%td/%ty".
'F'	ISO 8601 complete date formatted as "%tY-%tm-%td".
'c'	Date and time formatted as "%ta %tb %td %tT %tZ %tY", e.g. "Sun Jul 20 16:17:00 EDT 1969".

Any characters not explicitly defined as date/time conversion suffixes are illegal and are reserved for future extensions.

Flags

The following table summarizes the supported flags. *y* means the flag is supported for the indicated value types.

Flag	General	Int/Long	Double	Date/Time	Description
'.'	y	y	y	y	The result will be left-justified.
'#'	y	y ²	y		The result should use a conversion-dependent alternate form
'+'		y ³	y		The result will always include a sign
' '		y ³	y		The result will include a leading space for positive values
'0'		y	y		The result will be zero-padded
'.'		y ¹	y ⁴		The result will include locale-specific grouping separators
'('		y ³	y ⁴		The result will enclose negative numbers in parentheses

1. For 'd' conversion only.
2. For 'o', 'x', and 'X' conversions only.
3. For 'd' applied to Integer and Long.
4. For 'e', 'E', 'f', 'g', and 'G' conversions only.

Any characters not explicitly defined as flags are illegal and are reserved for future extensions.

Width

The *width* is the minimum number of characters to be written to the output. For the line separator conversion, *width* is not applicable; if it is provided, an exception will be thrown.

Precision

For general value types, *precision* is the maximum number of characters to be written to the output.

For integer, long, and date/time value types and the percent and line separator conversions, *precision* is not applicable; if precision is provided, an exception will be thrown. %[flags]
[width]conversion

fromJson

The user provides the **JSON** to be converted. Optionally the user could also provide the name of the type of *Biskit* to be created, if this is used the conversion will use knowledge of the target type being created instead of just a generic conversion. The output is a *Biskit* version of the **JSON** string.

prettyPrintJson

The user provides the ugly **JSON** to be prettified. The output is a prettier **JSON** string.

toDate

The screenshot shows a configuration window for the 'toDate' function. It has a 'Function' dropdown set to 'toDate' and an 'Argument' dropdown with three options: '(Date out)toDate(DateTime in)', '(Date out)toDate(Long in)', and '(Date out)toDate(DateTime in)'. The 'Argument' dropdown is currently open, showing these options. Below the dropdowns, there is a 'DateTime in' input field with a 'Null' value and a dropdown arrow.

The user provides the data to be converted which could be either a **DateTime** or a **Long** (number of seconds that have elapsed since January 1, 1970). The output is a **Date**.

toDateTime

The screenshot shows a configuration window for the 'toDatetime' function. It has a 'Function' dropdown set to 'toDatetime' and an 'Argument' dropdown with three options: '(DateTime out)toDatetime(Date in)', '(DateTime out)toDatetime(DateTime in)', and '(DateTime out)toDatetime(Long in)'. The 'Argument' dropdown is currently open, showing these options. Below the dropdowns, there is a 'Date in' input field with a 'Null' value and a dropdown arrow.

The user provides the data to be converted which could be either a **Date** or a **Long** (number of seconds that have elapsed since January 1, 1970). The output is a **DateTime**.

toDouble

The screenshot shows a configuration window for the 'toDouble' function. It has a 'Function' dropdown set to 'toDouble' and an 'Argument' dropdown with four options: '(Double out)toDouble(Int in)', '(Double out)toDouble(Long in)', '(Double out)toDouble(String in)', and '(Double out)toDouble(Double in)'. The 'Argument' dropdown is currently open, showing these options. Below the dropdowns, there is an 'Int in' input field.

The user provides the data to be converted which could be either an **Int**, a **Long** or a **String**. The output is a **Double**.

toInt

Function	toInt
Argument	(Int out)toInt(Double in) ▼
Double	(Int out)toInt(Long in)
	(Int out)toInt(String in)

The user provides the data to be converted which could be either a **String**, a **Long** or a **Double**. The output is a **Integer**.

toJson

Function	toJson
Arguments	(String json)toJson(Biskit.; biskit, Boolean dropNullProperties) ▼
	(String json)toJson(Biskit.; biskit)
	(String json)toJson(Biskit.; biskit, Boolean dropNullProperties)
Biskit.; biskit	Null ▼
Boolean dropNullProperties	Null ▼

The user provides the *biskit* to be converted. Optionally the user defines whether the JSON doesn't include properties of value null. The output is the **JSON** string.

toLong

Function	toInt
Argument	(Int out)toInt(Double in) ▼
Double	(Int out)toInt(Long in)
	(Int out)toInt(String in)

The user provides the data to be converted which could be either a **String**, an **Int** or a **Double**. The output is a **Long**.

toPDF

Function	toPDF
Arguments	3 functions (Biskit.Attachment, pdfAttachment Biskit.TemporaryFile, pdfTemporaryFile)toPDF(String xsf, String outputFilename, String title, String author, String creator, String keywords, Int targetResolution) (Biskit.Attachment, pdfAttachment Biskit.TemporaryFile, pdfTemporaryFile)toPDF(String xsf) (Biskit.Attachment, pdfAttachment Biskit.TemporaryFile, pdfTemporaryFile)toPDF(String xsf, String outputFilename)
String xsf	(Biskit.Attachment, pdfAttachment Biskit.TemporaryFile, pdfTemporaryFile)toPDF(String xsf, String outputFilename, String title, String author, String creator, String keywords, Int targetResolution)
String outputFilename	Null
String title	Null
String author	Null
String creator	Null
String keywords	Null
Int targetResolution	Null

The user provides a **String** holding the **XSL FO** data and optionally, the filename for the attachment, a title for the **PDF** file, the authors name, the creator, keywords (all to be stored in the metadadta) and resolution of the **PDF** file. The output is the **PDF** file as an attachment and a temporary file.

Some XSL FO tutorials:

- <http://www.herongyang.com/XSL-FO/>
- <http://w3schools.sinsixx.com/xslfo/default.asp.htm>
- <https://www.webucator.com/tutorial/learn-xsl-fo/index.cfm>
- <https://www.antennahouse.com/comprehensive-xsl-fo-tutorials-and-samples-collection/>
- <https://www.qctutorials.com/learning/xslfo/index.html>
- <https://www.ibm.com/developerworks/library/x-xslfo2app/?ca=dnt-46h>
- <http://www.cafeconleche.org/books/bible3/chapters/ch16.html>
- <http://www.renderx.com/tutorial.html>

The standard is described here:

- <https://www.w3.org/TR/xsl/>

Barcodes

For PDFs to include barcodes, see the XSL FO workflow example mentioned above for a "Hello World" type example. For examples of all the types of barcode available, see <http://barcode4j.sourceforge.net/examples.html> ⁴⁰⁵

The format of the XML required inside the XSL FO file is described at <http://barcode4j.sourceforge.net/2.1/barcode-xml.html> ⁴⁰⁵

For details on the different types of barcode available, see <https://github.com/lindell/JsBarcode/wiki> ⁴⁰⁵

Note that the PDF barcodes (generated by Barcode4j) have a different set of barcode types from those available in the browser (generated by JsBarcode). In particular, the PDF barcodes can include 2D barcodes in QR format whereas those in the browser cannot.

toPDF_from_docx

Function	toPDF_from_docx (Biskit:Attachment; pdf)toPDF_from_docx(Biskit:File; template,Biskit:: biskitToConvert) ▼
Arguments	
Biskit:File; template	Null ▼
Biskit:: biskitToConvert	Null ▼

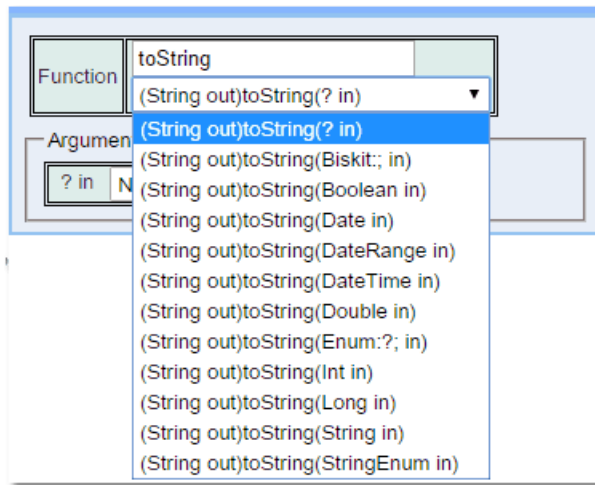
The user provides **Docx** file as a template and the [Biskit](#)⁶⁵² to provide the information for the template. Output is the **PDF** file as an **Attachment**. When creating the Template in **Word** use **<alt>i** followed by **f** to bring up the Field option. The **Field name** required is **MERGEFIELD**, once selected fill the property name into the **Field name** box starting with a b. or biskit and encased in \${ }. If you copy and paste a merge field you must remember to right click and **Edit Field** in order to change the **Field name**.

toPDF_from_odt

Function	toPDF_from_odt (Biskit:Attachment; pdf)toPDF_from_odt(Biskit:File; template,Biskit:: biskitToConvert) ▼
Arguments	
Biskit:File; template	Null ▼
Biskit:: biskitToConvert	Null ▼

The user provides **odt** file as a template and the [Biskit](#)⁶⁵² to provide the information for the template. Output is the **PDF** file as an **Attachment**. When creating the Template in ODT, use Insert->Field->More Fields (Ctrl+F2), choose Input Field, Insert and then type in the field required starting with a \${b. or \${biskit. and ending with a }

toString



The user provides the data to be converted which could be one of a large number of options, please see the image for the complete list. The output is a **String**.

6.13.3.12.4 Date & Time Function Type

Function	Description
addBusinessDays	Adds a given number of business days to a DateTime or a Date .
addDays	Adds a number of days to a DateTime or a Date .
addHours	Adds a number of hours to a DateTime .
addMinutes	Adds a number of minutes to a DateTime .
calculateWorkingHours	Returns how many minutes are within, and outside working hours for a particular Date Range .
createDate	Returns a Date when given the year, month, day of month.
createDateTime	Returns a DateTime when given the year, month, day of month, hour and minute.
createDateRange	Returns a DateRange when given two DateTime 's.
dayOfMonth	Returns the day of the month of a Date or a Date Time .
dayOfWeek	Returns the day of the week of a Date or a Date Time .
hour	Returns the hour of a Date Time .
intersect	Returns how many minutes the two DateRanges , four DateTimes or DateRange and two DateTime 's intersect by.
isBusinessDay	Returns whether a day in a DateTime or a Date is a business day.
isRangeOutsideWorkingHours	Returns whether a Date Range is outside the specified working hours.
millisecond	Returns the millisecond of a DateTime .
minute	Returns the minute of a DateTime .
minuteOfDay	Returns the minute of the day of a DateTime .
month	Returns the month of a DateTime or a Date .
monthZeroBased	Returns the month of a DateTime or a Date with January as 0.
parseDate	Returns the parsed Date .
parseDateTime	Returns the parsed DateTime .
second	Returns the second of a DateTime .
setDayOfMonth	Sets the dayOfMonth in a DateTime or a Date .
setHour	Sets the hour in a DateTime .
setMillisecond	Sets the millisecond in a DateTime .
setMinute	Sets the minute in a DateTime .
setMinuteOfDay	Sets the minuteOfDay in a DateTime .
setMonth	Sets the month in a DateTime or a Date .

Function	Description
setMonthZeroBased	Sets the month based on January being 0, in a DateTime or a Date .
setSecond	Sets the second in a DateTime .
setYear	Sets the year in a DateTime or a Date .
subtractBusinessDays	Returns the Date after subtracting a number of business days.
subtractDays	Subtracts a number of days to a DateTime or a Date .
subtractHours	Subtracts a number of hours to a DateTime .
subtractMinutes	Subtracts a number of minutes to a DateTime .
timeDiffBusinessDays	Returns how many business days there are between two Dates DateTime s
timeDiffDays	Returns the difference in days between two DateTime s or two Dates
timeDiffHours	Returns the difference in hours between two DateTime s
timeDiffMilliseconds	Returns the difference in milliseconds between two DateTime s
timeDiffMinutes	Returns the difference in minutes between two DateTime s
timeDiffSeconds	Returns the difference in seconds between two DateTime s
weekOfYear	Returns the week of the year of a Date or a DateTime .
year	Returns the year of a Date or a DateTime .

addBusinessDays

The screenshot shows a configuration window for the **addBusinessDays** function. It includes a 'Function' dropdown menu, an 'Argument' dropdown menu, and two input fields for arguments: 'Date date' and 'Int offset'. Both are currently set to 'Null'.

Allows the addition of a number of business days (as **Integer**) to either a **DateTime** or a **Date**. Returns the new **DateTime** or **Date**.

addDays

Function	addDays
Argument	<div> <div>(Date addDays)addDays(Date dateTime,Double number) ▼</div> <div>(Date addDays)addDays(Date dateTime,Double number)</div> <div>(Date addDays)addDays(Date dateTime,Int number)</div> <div>(DateTime addDays)addDays(DateTime dateTime,Double number)</div> <div>(DateTime addDays)addDays(DateTime dateTime,Int number)</div> </div>
Date da	
Double	

Allows the addition of a number of days (as **Integer** or **Double**) to either a **DateTime** or a **Date**. Returns the new **DateTime** or **Date**.

addHours

Function	addHours
Argument	<div> <div>(DateTime addHours)addHours(DateTime dateTime,Double number) ▼</div> <div>(DateTime addHours)addHours(DateTime dateTime,Double number)</div> <div>(DateTime addHours)addHours(DateTime dateTime,Int number)</div> </div>
DateTime dateTime	Null ▼
Double number	Null ▼

Allows the addition of a number of hours (as **Integer** or **Double**) to a **DateTime**. Returns the new **DateTime**.

addMinutes

Function	addMinutes
Argument	<div> <div>(DateTime addMinutes)addMinutes(DateTime dateTime,Double number) ▼</div> <div>(DateTime addMinutes)addMinutes(DateTime dateTime,Double number)</div> <div>(DateTime addMinutes)addMinutes(DateTime dateTime,Int number)</div> </div>
DateTime dateTime	Null ▼
Double number	Null ▼

Allows the addition of a number of minutes (as **Integer** or **Double**) to a **DateTime**. Returns the new **DateTime**.

calculateWorkingHours

Function	calculateWorkingHours (Int insideWorkingHours,Int outsideWorkingHours)calculateWorkingHours(Int startWorkingMinuteOfDay,Int finishWorkingMinuteOfDay,DateTime rangeStart,DateTime rangeFinish)
Arguments	
Int startWorkingMinuteOfDay	Null
Int finishWorkingMinuteOfDay	Null
DateTime rangeStart	Null
DateTime rangeFinish	Null

Calculates how many minutes of a **Date Range** are in and out of working hours.

Takes **Integer startWorkingMinuteOfDay**: the start minute of the working day. For example, 0 for midnight, 480 for 8am and 510 for 8.30am.

Takes **Integer finishWorkingMinuteOfDay**: the finish minute of the day. For example, 1220 for 5pm, 1280 for 6pm and 1440 for midnight.

Takes **Date Time rangeStart**: the start of the period that you want to do the calculation for.

Takes **Date Time rangeFinish**: the finish of the period that you want to do the calculation for.

Returns how many minutes are inside working hours and how many minutes are outside working hours . Outside working hours is defined by the weekend, and holidays, as defined by **Holiday Date [Biskits](#)**⁶⁵², or outside the standard working hours as indicated by **[startWorkingMinuteOfDay, finishWorkingMinuteOfDay]**.

createDate

Function	createDate (Date createDate)createDate(Int year,Int month,Int dayOfMonth)
Arguments	
Int year	Null
Int month	Null
Int dayOfMonth	Null

Takes a year, month, day of month and returns the **Date**.

createDateTime

Function	createDateTime (DateTime createDateTime)createDateTime(Int year,Int month,Int dayOfMonth,Int hour,Int minute) (DateTime createDateTime)createDateTime(Int year,Int month,Int dayOfMonth,Int hour,Int minute) (DateTime createDateTime)createDateTime(Int year,Int month,Int dayOfMonth,Int hour,Int minute,Int second)
Arguments	
Int year	Null
Int month	Null
Int dayOfMonth	Null
Int hour	Null
Int minute	Null

Takes a year, month, day of month, hour, minute and optionally second, and returns the **DateTime**.

createDateRange

Function	createDateRange
Arguments	2 functions ? (DateTime createDateRange)createDateRange(DateTime start,DateTime finish) ▾ (DateTime createDateRange)createDateRange() (DateTime createDateRange)createDateRange(DateTime start,DateTime finish)
DateTime start	Null ▾
DateTime finish	Null ▾

Takes two **DateTime**'s or optionally null, and returns the **DateRange**. If the null option is used the **DateRange** returned is from now until December 31st 2999.

dayOfMonth

Function	dayOfMonth
Argument	(Int dayOfMonth)dayOfMonth(Date date) ▾ (Int dayOfMonth)dayOfMonth(Date date) (Int dayOfMonth)dayOfMonth(DateTime dateTime)
Date date	Null ▾

Takes a **Date** or **Date Time** and returns the day of the month.

dayOfWeek

Function	dayOfWeek
Argument	(Int dayOfWeek)dayOfWeek(DateTime dateTime) ▾ (Int dayOfWeek)dayOfWeek(Date date) (Int dayOfWeek)dayOfWeek(DateTime dateTime)
DateTime dateTime	Null ▾

Takes a **Date** or **Date Time** and returns the day of the week.

hour

The screenshot shows a configuration window for the 'hour' function. It has a 'Function' section with a text box containing 'hour' and a dropdown menu showing '(Int hour)hour(DateTime dateTime)'. Below this is an 'Arguments' section with a text box containing 'DateTime dateTime' and a dropdown menu showing 'Null'.

Takes a **Date Time** and returns the hour.

intersect

The screenshot shows a configuration window for the 'intersect' function. It has a 'Function' section with a text box containing 'intersect' and a dropdown menu showing '4 functions'. Below this is an 'Arguments' section with a text box containing 'DateTime dateTime' and a dropdown menu showing 'Null'.

Calculates how many minutes two **DateRange**'s, four **DateTime**'s or a combination of a **DateRange** and two **DateTime**'s overlap by. Returns the number of minutes as an **Integer**.

isBusinessDay

The screenshot shows a configuration window for the 'isBusinessDay' function. It has a 'Function' section with a text box containing 'isBusinessDay' and a dropdown menu showing '(Boolean isBusinessDay)isBusinessDay(Date date)'. Below this is an 'Argument' section with a text box containing 'Date date' and a dropdown menu showing 'Null'.

Takes a **Date** or **Date Time** and returns whether it is a business day or not.

isRangeOutsideWorkingHours

Function	isRangeOutsideWorkingHours (Boolean isRangeOutsideBusinessHours)isRangeOutsideWorkingHours(Int startWorkingMinuteOfDay,Int finishWorkingMinuteOfDay,DateTime rangeStart,DateTime rangeFinish) ▼		
Arguments			
Int startWorkingMinuteOfDay	Null	▼	
Int finishWorkingMinuteOfDay	Null	▼	
DateTime rangeStart	Null	▼	
DateTime rangeFinish	Null	▼	

Checks whether any of the period between two times is outside working hours.

Takes **Integer startWorkingMinuteOfDay**: the start minute of the working day. For example, 0 for midnight, 480 for 8am and 510 for 8.30am.

Takes **Integer finishWorkingMinuteOfDay**: the finish minute of the day. For example, 1220 for 5pm, 1280 for 6pm and 1440 for midnight.

Takes **Date Time rangeStart**: the start of the period that you want to check for being outside working hours.

Takes **Date Time rangeFinish**: the finish of the period that you want to check for being outside working hours.

Returns true when the range [**rangeStart**, **rangeFinish**] covers any period that is not inside working hours. That means a weekend, holiday, as defined by **Holiday Date** [Biskits](#)⁶⁵², or outside the standard working hours as indicated by [**startWorkingMinuteOfDay**, **finishWorkingMinuteOfDay**].

millisecond

Function	millisecond (Int millisecond)millisecond(DateTime dateTime) ▼		
Arguments			
DateTime dateTime	Null	▼	

Takes a **Date Time** and returns the millisecond.

minute

Function	minute (Int minute)minute(DateTime dateTime) ▼		
Arguments			
DateTime dateTime	Null	▼	

Takes a **Date Time** and returns the minute.

minuteOfDay

Function	minuteOfDay	(Int minuteOfDay)minuteOfDay(DateTime dateTime) ▾		
Arguments	<table border="1"><tr><td>DateTime dateTime</td><td>Null ▾</td></tr></table>		DateTime dateTime	Null ▾
DateTime dateTime	Null ▾			

Takes a **Date Time** and returns the minute of the day.

month

Function	month	(Int month)month(Date date) ▾
Argument	(Int month)month(DateTime dateTime)	
Date date	Null ▾	

Takes a **Date** or **Date Time** and returns the month.

monthZeroBased

Function	monthZeroBased	(Int monthZeroBased)monthZeroBased(Date date) ▾
Argument	(Int monthZeroBased)monthZeroBased(DateTime dateTime)	
Date date	Null ▾	

Takes a **Date** or **Date Time** and returns the month with January as 0.

parseDate

Function	parseDate
Argument	(Date date)parseDate(String dateString) (Date date)parseDate(String dateString, Boolean preferUSDateFormat) (Date date)parseDate(String dateString, String dateFormat) (Date date)parseDate(String dateString, String dateFormat)

Takes a **String** and returns the parsed **Date**. The user can provide a date format to assist the parsing, or specify that its US date format.

parseDateTime

Function	parseDateTime
Argument	(DateTime dateTime)parseDateTime(String dateString) (DateTime dateTime)parseDateTime(String dateString, Boolean preferUSDateFormat) (DateTime dateTime)parseDateTime(String dateString, Boolean preferUSDateFormat) (DateTime dateTime)parseDateTime(String dateString, String dateFormat)

Takes a **String** and returns the parsed **DateTime**. The user can provide a date format to assist the parsing, or specify that its US date format.

second

Function	second
Argument	(Int second)second(DateTime dateTime)
Argument	Null

Takes a **Date Time** and returns the second.

setDayOfMonth

Function	setDayOfMonth	(Date date)setDayOfMonth(Date date,Int setDayOfMonth) ▾
Argument	(Date date)setDayOfMonth(Date date,Int setDayOfMonth)	
	Date date	Null ▾
	Int setDayOfMonth	Null ▾

Takes a **Date** or **DateTime** and the day of the month as an **Integer** and returns the new **Date** or **Date Time**.

setHour

Function	setHour	(DateTime dateTime)setHour(DateTime dateTime,Int setHour) ▾
Arguments		
	DateTime dateTime	Null ▾
	Int setHour	Null ▾

Takes a **Date Time** and the hour as an **Integer** and returns the new **Date Time**.

setMillisecond

Function	setMillisecond	(DateTime dateTime)setMillisecond(DateTime dateTime,Int setMillisecond) ▾
Arguments		
	DateTime dateTime	Null ▾
	Int setMillisecond	Null ▾

Takes a **Date Time** and the millisecond as an **Integer** and returns the new **Date Time**.

setMinute

Function	setMinute	
	(DateTime dateTime)setMinute(DateTime dateTime,Int setMinute) ▾	
Arguments		
DateTime dateTime	Null ▾	
Int setMinute	Null ▾	

Takes a **Date Time** and the minute as an **Integer** and returns the new **Date Time**.

setMinuteOfDay

Function	setMinuteOfDay	
	(DateTime dateTime)setMinuteOfDay(DateTime dateTime,Int setMinuteOfDay) ▾	
Arguments		
DateTime dateTime	Null ▾	
Int setMinuteOfDay	Null ▾	

Takes a **Date Time** and the minute as an **Integer** and returns the new **Date Time**.

setMonth

Function	setMonth	
	(Date date)setMonth(Date date,Int setMonth) ▾	
Argument	(Date date)setMonth(Date date,Int setMonth)	
	(DateTime dateTime)setMonth(DateTime dateTime,Int setMonth)	
Date date	Null ▾	
Int setMonth	Null ▾	

Takes a **Date** or **DateTime** and the month as an **Integer** and returns the new **Date** or **Date Time**.

setMonthZeroBased

Function	setMonthZeroBased	(Date date)setMonthZeroBased(Date date,Int setMonthZeroBased) ▾
Argument	(Date date)setMonthZeroBased(Date date,Int setMonthZeroBased)	
	(DateTime dateTime)setMonthZeroBased(DateTime dateTime,Int setMonthZeroBased)	
Date date	Null ▾	
Int setMonthZeroBased	Null ▾	

Takes a **Date** or **DateTime** and the month as an **Integer** where January is 0, returns the new **Date** or **Date Time**.

setSecond

Function	setSecond	(DateTime dateTime)setSecond(DateTime dateTime,Int setSecond) ▾
Arguments		
DateTime dateTime	Null ▾	
Int setSecond	Null ▾	

Takes a **Date Time** and the second as an **Integer** and returns the new **Date Time**.

setYear

Function	setYear	(Date date)setYear(Date date,Int setYear) ▾
Argument	(Date date)setYear(Date date,Int setYear)	
	(DateTime dateTime)setYear(DateTime dateTime,Int setYear)	
Date date	Null ▾	
Int setYear	Null ▾	

Takes a **Date** or **DateTime** and the year as an **Integer** and returns the new **Date** or **Date Time**.

subtractBusinessDays

Function	subtractBusinessDays
Arguments	<div> <div>(Date result)subtractBusinessDays(Date date,Int offset)</div> <div>(Date result)subtractBusinessDays(Date date,Int offset)</div> <div>(DateTime result)subtractBusinessDays(DateTime date,Int offset)</div> </div>
Date date	Null
Int offset	Null

Allows the subtraction of a number of business days (as **Integer**) from a **DateTime** or a **Date**. Returns the new **DateTime** or **Date**.

subtractDays

Function	subtractDays
Argument	<div> <div>(Date subtractDays)subtractDays(Date dateTime,Double number)</div> <div>(Date subtractDays)subtractDays(Date dateTime,Double number)</div> <div>(Date subtractDays)subtractDays(Date dateTime,Int number)</div> <div>(DateTime subtractDays)subtractDays(DateTime dateTime,Double number)</div> <div>(DateTime subtractDays)subtractDays(DateTime dateTime,Int number)</div> </div>
Date dateTime	
Double number	

Allows the subtraction of a number of days (as **Integer** or **Double**) from a **DateTime** or a **Date**. Returns the new **DateTime** or **Date**.

subtractHours

Function	subtractHours
Argument	<div> <div>(DateTime subtractHours)subtractHours(DateTime dateTime,Double number)</div> <div>(DateTime subtractHours)subtractHours(DateTime dateTime,Double number)</div> <div>(DateTime subtractHours)subtractHours(DateTime dateTime,Int number)</div> </div>
DateTime dateTime	Null
Double number	Null

Allows the subtraction of a number of hours (as **Integer** or **Double**) from a **DateTime**. Returns the new **DateTime**.

subtractMinutes

Function	subtractMinutes
Argument	(DateTime subtractMinutes)subtractMinutes(DateTime dateTime,Double number) ▼ (DateTime subtractMinutes)subtractMinutes(DateTime dateTime,Double number) (DateTime subtractMinutes)subtractMinutes(DateTime dateTime,Int number)
DateTime dateTime	Null ▼
Double number	Null ▼

Allows the subtraction of a number of minutes (as **Integer** or **Double**) from a **DateTime**. Returns the new **DateTime**.

timeDiffBusinessDays

Function	timeDiffBusinessDays
Arguments	(Double businessDaysDifference)timeDiffBusinessDays(Date firstDate,Date secondDate) ▼ (Double businessDaysDifference)timeDiffBusinessDays(Date firstDate,Date secondDate) (Double businessDaysDifference)timeDiffBusinessDays(DateTime firstDate,DateTime secondDate) (Int businessDaysDifference)timeDiffBusinessDays(Date firstDate,Date secondDate) (Int businessDaysDifference)timeDiffBusinessDays(DateTime firstDate,DateTime secondDate)
Date firstDate	
Date secondDate	

Calculates the difference in business days between two **DateTimes** or **Dates**. Returns the difference in days as an **Integer** or a **Double**

timeDiffDays

Function	timeDiffDays
Arguments	(Double timeDiffDays)timeDiffDays(Date a,Date b) ▼ (Double timeDiffDays)timeDiffDays(Date a,Date b) (Double timeDiffDays)timeDiffDays(DateTime a,DateTime b) (Int timeDiffDays)timeDiffDays(Date a,Date b) (Int timeDiffDays)timeDiffDays(DateTime a,DateTime b)
Date a	Null
Date b	Null

Calculates the difference in days between two **DateTimes** or **Dates**. Returns the difference in days as an **Integer** or a **Double**.

timeDiffHours

Function	timeDiffHours (Int timeDiffHours)timeDiffHours(DateTime a,DateTime b) ▼
Arguments	
DateTime a	Null ▼
DateTime b	Null ▼

Calculates the difference in hours between two **DateTimes**. Returns the difference in hours as an **Integer**.

timeDiffMilliseconds

Function	timeDiffMilliseconds 1 functions ? (Long timeDiffMilliseconds)timeDiffMilliseconds(DateTime a,DateTime b) ▼
Arguments	
DateTime a	Null ▼
DateTime b	Null ▼

Calculates the difference in milliseconds between two **DateTimes**. Returns the difference in milliseconds as an **Integer**.

timeDiffMinutes

Function	timeDiffMinutes 1 functions ? (Int timeDiffMinutes)timeDiffMinutes(DateTime a,DateTime b) ▼
Arguments	
DateTime a	Null ▼
DateTime b	Null ▼

Calculates the difference in minutes between two **DateTimes**. Returns the difference in minutes as an **Integer**.

timeDiffSeconds

Function	timeDiffSeconds
	1 functions
	(Long timeDiffSeconds)timeDiffSeconds(DateTime a,DateTime b) ▼
Arguments	
DateTime a	Null ▼
DateTime b	Null ▼

Calculates the difference in seconds between two **DateTimes**. Returns the difference in seconds as an **Integer**.

weekOfYear

Function	weekOfYear
	(Int weekOfYear)weekOfYear(DateTime dateTime) ▼
Arguments	
DateTime dateTime	Null ▼

Takes a **Date** or **Date Time** and returns the week of the year.

year

Function	year
	(Int year)year(Date date) ▼
Argument	(Int year)year(Date date)
	(Int year)year(DateTime dateTime)
Date date	Null ▼

Takes a **Date** or **Date Time** and returns the year.

6.13.3.12.5 File Function Type

Function	Description
createAttachment	Creates an attachment from an OS file.
createFile	Creates a file from an OS file.
createOSFile	Creates a temporary file with empty content.
createTemporary Directory	Creates a temporary directory.
createTemporary File	Creates a temporary file from an OS file.
getOSFileInfo	Generates information about an OS file.
listOSFiles	Lists the contents of an OS directory.
loadOSFile	Loads the contents of an OS file into a String .

createAttachment

Function	createAttachment
Arguments	(Biskit.Attachment; attachment)createAttachment(String filePath) (Biskit.Attachment; attachment)createAttachment(String filePath) (Biskit.Attachment; attachment)createAttachment(String filePath,String attachmentName)
	String filePath Null

Takes the path name to an OS file and optionally the name of the attachment to be created. Returns a *Biskit* of type **Attachment**.

createFile

Function	createFile
Arguments	(Biskit.Attachment; file)createFile(Biskit.Directory; parentDir,String filePath)
	Biskit.Directory; parentDir Null
	String filePath Null

Takes the name of the directory the file should be in and the full path name to an OS file. Returns a *Biskit* of type **Attachment**.

createOSFile

Function	createOSFile
Arguments	<div> <div>(Biskit:OSFileInfo; info)createOSFile(String prefix,String suffix,Biskit:Attachment; attachment)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String name)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String name,Biskit:Attachment; attachment)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String name,Biskit:File; file)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String name,String fileContent)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String parentDirPath,String prefix,String suffix,Biskit:Attachment; attachment)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String parentDirPath,String prefix,String suffix,Biskit:File; file)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String parentDirPath,String prefix,String suffix,String fileContent)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String prefix,String suffix,Biskit:Attachment; attachment)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String prefix,String suffix,Biskit:File; file)</div> <div>(Biskit:OSFileInfo; info)createOSFile(String prefix,String suffix,String fileContent)</div> </div>

Takes information to define the file to be created. This could be the name of the file, the name of the directory for the file, or a suffix and prefix to create a randomly named file. Also may be defined are where the contents are to come from, (Attachment, Biskit of type File, String). Returns a *Biskit* describing the file that has been created.

createTemporaryDirectory

Function	createTemporaryDirectory
Arguments	<div> <div>(Biskit:OSFileInfo; info)createTemporaryDirectory(String prefix,String parentDirPath)</div> <div>(Biskit:OSFileInfo; info)createTemporaryDirectory(String prefix)</div> <div>(Biskit:OSFileInfo; info)createTemporaryDirectory(String prefix,String parentDirPath)</div> </div>
String prefix	Null
String parentDirPath	Null

Creates a temporary directory that will automatically be deleted at the end of handling the current Workflow. Takes the prefix to apply to the name of the created directory and optionally the parent directory of the one to be created. Returns a *Biskit* describing the directory that has been created.

createTemporaryFile

Function	createTemporaryFile
Arguments	<div> <div>(Biskit:TemporaryFile; tempFile)createTemporaryFile(String filePath,String prefix,String suffix)</div> </div>
String filePath	Variable Source #22 info.path
String prefix	Null
String suffix	Fixed .pdf

Creates a temporary file from an OS file, that will automatically be deleted at the end of handling the current Workflow. Takes the **filePath** or **attachment** that points at the file to be converted to a temporary file. Also the prefix and suffix of the temporary files name. Returns a *Biskit* describing the **temporaryFile** that has been created.

getOSFileInfo

Function	getOSFileInfo (Biskit:OSFileInfo; info)getOSFileInfo(String path) ▼		
Arguments	<table border="1"><tr><td>String path</td><td>Null ▼</td></tr></table>	String path	Null ▼
String path	Null ▼		

Takes the path to the file to be examined. Returns a *Biskit* describing the file of type **OSFileInfo**.

listOSFiles

Function	listOSFiles (List:Biskit:OSFileInfo; files)listOSFiles(String dirPath) ▼		
Arguments	<table border="1"><tr><td>String dirPath</td><td>Null ▼</td></tr></table>	String dirPath	Null ▼
String dirPath	Null ▼		

Takes the path to the directory to be examined. Returns a list of *Biskits* describing the files of type **OSFileInfo**.

loadOSFile

Function	loadOSFile (String contents)loadOSFile(String filePath) ▼		
Arguments	<table border="1"><tr><td>String filePath</td><td>Null ▼</td></tr></table>	String filePath	Null ▼
String filePath	Null ▼		

Takes the path to the file to be loaded. Returns the contents of the file in a **String**.

6.13.3.12.6 List/Set Function Type

Function	Description
add	Takes two lists of <i>Biskits</i> ⁶⁵² and returns the combined list and the size of that list.
add	Add an item to a list of the same type of item.
addUserToProject	Adds a user to a <i>Project</i> .
contains	Check whether a list contains a particular element.
copy	Copies a list of <i>Biskits</i> to another list.
createList	Creates a list of the appropriate type.
createNumericSequence	Generates a list of numeric values using values from a linear sequence, which may optionally repeat.
exclusiveUnion	Takes two lists of <i>Biskits</i> and returns the list of <i>Biskits</i> only found in one of lists, and the size of that list.
first	Takes a list of objects and returns the first one.
get	Returns the requested object from the list of objects. 0 is the index to the first object in the list.
intersect	Takes two lists of <i>Biskits</i> and returns the list of <i>Biskits</i> found in both lists, and the size of that list.
last	Takes a list of objects and returns the last one.
max	Returns the maximum value in a list.
min	Returns the minimum value in a list.
randomListOrder	Produces a new list in a random order.
remove	Removes an item from a list
removeDuplicates	Takes a list of <i>Biskits</i> and returns a Set with any duplicates removed.
removeUserFromProject	Removes a user from the given <i>Project</i> .
reverseListOrder	Reverses the order of the items in a List or Set.
size	Returns the size of the input list.
subtract	Returns a list of <i>Biskits</i> holding those <i>Biskits</i> in the first input list that are not in the second input list.
subList	Returns a list that is a sub list of the original. 0 is the index to the first object in the list.
sum	Returns the sum of the values of a particular <i>property</i> ⁶⁵⁵ on the <i>Biskits</i> stored in the list.
union	Takes two lists and returns the combined list with duplicates removed.

add

Function	add (List:Biskit;; add,Int size)add(List:Biskit;; a,List:Biskit;; b) ▼
Arguments	
List:Biskit;; a	Null ▼
List:Biskit;; b	Null ▼

Takes the two input lists of *Biskits* and adds them together including duplicates. Returns the combined list and the size of that list.

Function	add (List:Boolean list)add(List:Boolean list,Boolean item) ▼
Arguments	
List:Boolean	(Double out)add(Double a,Double b)
Boolean item	(Int add)add(Int a,Int b)
	(List:Biskit;; add,Int size)add(List:Biskit;; a,List:Biskit;; b)
	(List:Biskit::typeOf:list; list)add(List:Biskit;; list,Biskit::typeOf:list; item)
	(List:Boolean list)add(List:Boolean list,Boolean item)
	(List:Date list)add(List:Date list,Date item)
	(List:DateRange list)add(List:DateRange list,DateRange item)
	(List:DateTime list)add(List:DateTime list,DateTime item)
	(List:Double::typeOf:list; list)add(List:Double list,Double::typeOf:list; item)
	(List:Int::typeOf:list; list)add(List:Int list,Int::typeOf:list; item)
	(List:Long list)add(List:Long list,Long item)
	(List:String::typeOf:list; list)add(List:String list,String::typeOf:list; item)

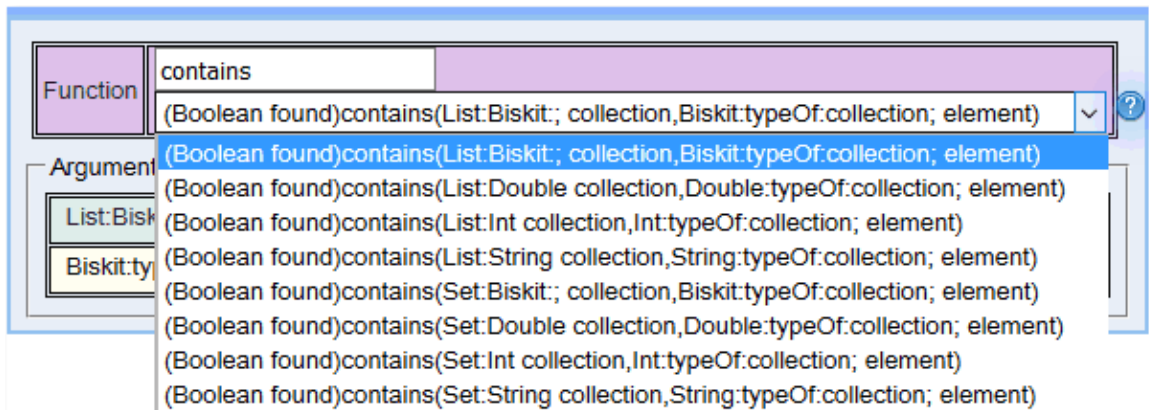
Takes a List of a particular type and an item of that type and adds the item to the List. Returns the new List.

addUserToProject

Function	addUserToProject (Boolean added)addUserToProject(Biskit:Calpendo.CalpendoUser; user,Biskit:Calpendo.Project; project) ▼
Arguments	
Biskit:Calpendo.CalpendoUser; user	Null ▼
Biskit:Calpendo.Project; project	Null ▼

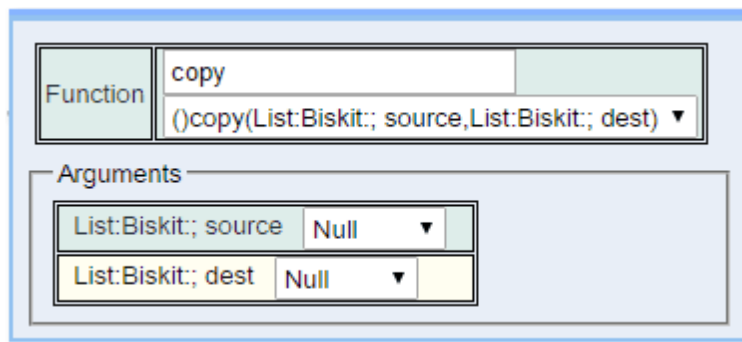
Takes the as input a the *User* and the *Project*. Returns **True** if the *User* was added to the *Project*.

contains



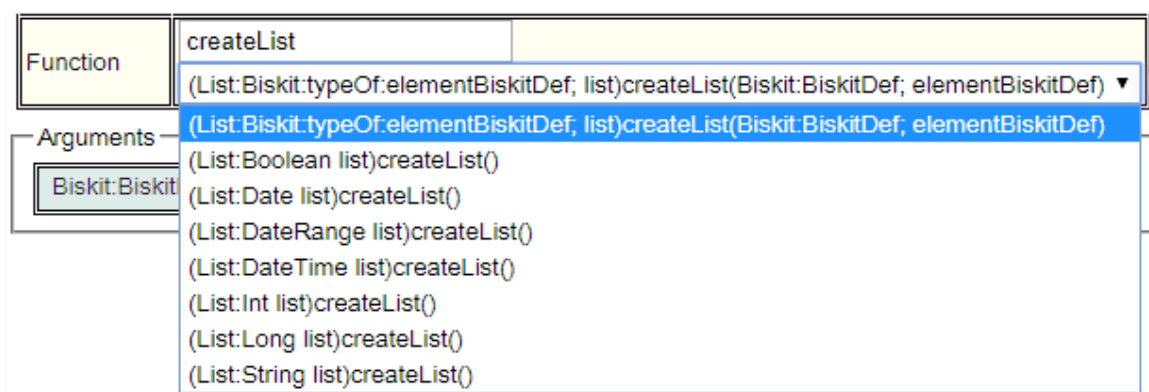
Takes the as input a list of *Biskits* and a *biskit* the same type as the list and returns True if the Biskit is in the list.

copy



Takes an input list of *Biskits* and copies it to the destination list. Nothing is returned.

createList



Creates a list of the appropriate type. For a list of *Biskits* the type of *Biskit* must be input. Returned is the new empty list.

createNumericSequence

Function	createNumericSequence
	(List:Double list)createNumericSequence(Double initialValue,Double increment,Int repeatSize,Int totalSize) ▼
Argument	(List:Double list)createNumericSequence(Double initialValue,Double increment,Int repeatSize,Int totalSize)
	(List:Int list)createNumericSequence(Int initialValue,Int increment,Int repeatSize,Int totalSize)
	(List:Long list)createNumericSequence(Long initialValue,Long increment,Int repeatSize,Int totalSize)
Double	
Double increment	Null ▼
Int repeatSize	Null ▼
Int totalSize	Null ▼

Takes as input the Initial Value of the sequence, the increment, the repeat size and the total size of the list to be created. These can be **Integers**, **Doubles** or **Longs**. Returns the completed list.

For example, by specifying **initialValue** = 6, **increment** = 2, **repeatSize** = 0, **totalSize** = 10, the output will be a list containing [6, 8, 10, 12, 14, 16, 18, 20, 22, 24].

If the last example is modified by setting **repeatSize** = 3, then the output will be a list containing [6, 8, 10, 6, 8, 10, 6, 8, 10, 6].

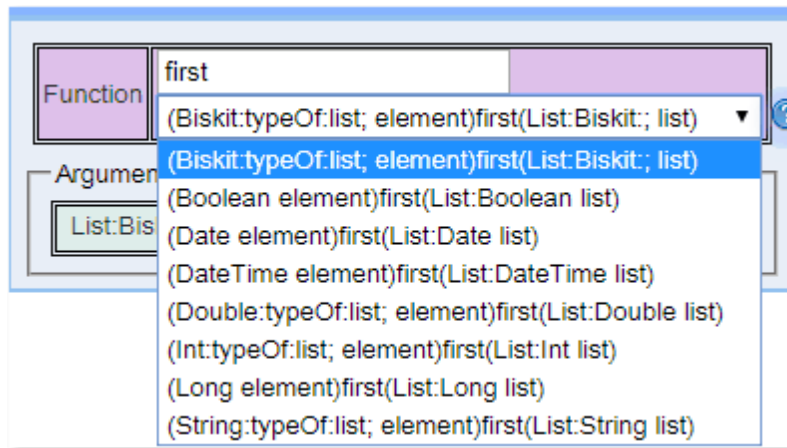
If the last example is modified again by setting **repeatSize** = 1, then the output will be a list containing [6, 6, 6, 6, 6, 6, 6, 6, 6, 6].

exclusiveUnion

Function	exclusiveUnion
	(List:Biskit;; exclusiveUnion,Int size)exclusiveUnion(List:Biskit;; a,List:Biskit;; b) ▼
Arguments	
List:Biskit;; a	Null ▼
List:Biskit;; b	Null ▼

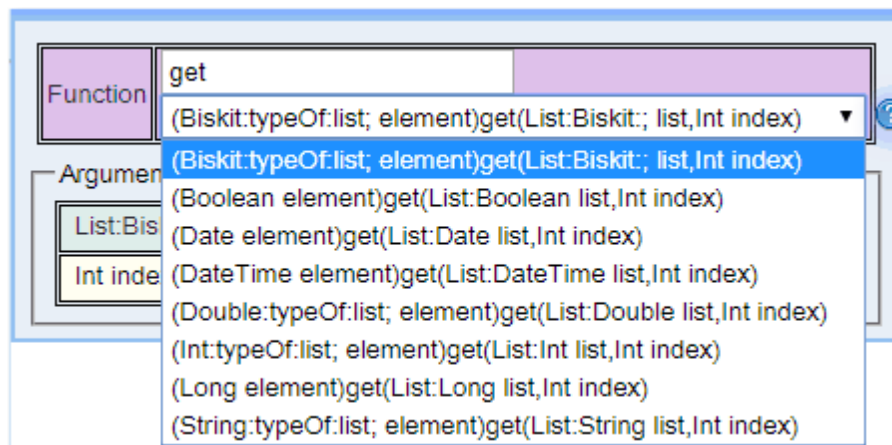
Takes the two input lists of *Biskits* and finds all those that are only in one of the lists. Returns the list of unique *Biskits* and the size of that list.

first



Takes as input a list of objects. Returns the first object in the list, which is at index 0.

get



Takes as input a list of objects and the position in the list of the object to be returned, indexing starts at 0. Returns the object.

intersect

Function	intersect
	(List:Biskit; intersect,Int size)intersect(List:Biskit; a,List:Biskit; b) ▼
Arguments	
List:Biskit; a	Null ▼
List:Biskit; b	Null ▼

Takes the two input lists of *Biskits* and finds all those that are in both lists. Returns the list of intersecting *Biskits* and the size of that list.

last

Function	last
	(Biskit:typeof: list; element)last(List:Biskit; list) ▼
Argument	(Biskit:typeof: list; element)last(List:Biskit; list)
List:Bis	(Boolean element)last(List:Boolean list)
	(Date element)last(List:Date list)
	(DateTime element)last(List:DateTime list)
	(Double:typeof: list; element)last(List:Double list)
	(Int:typeof: list; element)last(List:Int list)
	(Long element)last(List:Long list)
	(String:typeof: list; element)last(List:String list)

Takes as input a list of objects. Returns the last object in the list.

max

Function	max
	8 functions
	(Date element)max(List:Date list) ▼
Argument	(Date element)max(List:Date list)
List:Date list	(DateTime element)max(List:DateTime list)
	(Double out)max(Double a,Double b)
	(Double:typeof: list; element)max(List:Double list)
	(Int max)max(Int a,Int b)
	(Int:typeof: list; element)max(List:Int list)
	(Long element)max(List:Long list)
	(Long max)max(Long a,Long b)

Takes as input a list of objects. Returns the maximum value in the list or Null.

min

Function	min
Arguments	<div> <div>8 functions</div> <div> <div>(Date element)min(List:Date list)</div> <div>(Date element)min(List:Date list)</div> <div>(DateTime element)min(List:DateTime list)</div> <div>(Double out)min(Double a,Double b)</div> <div>(Double:typeOf:list; element)min(List:Double list)</div> <div>(Int min)min(Int a,Int b)</div> <div>(Int:typeOf:list; element)min(List:Int list)</div> <div>(Long element)min(List:Long list)</div> <div>(Long min)min(Long a,Long b)</div> </div> </div>

Takes as input a list of objects. Returns the minimum value in the list or Null.

randomListOrder

Add Event	
Trigger Type	
Name	
Enabled	
Children to	
Record Sys	
Sort Order	
Completion	
Last Ran	
ID	
Conditions	
Function	<div> <div>(List:Biskit;; randomList)randomListOrder(List:Biskit;; incomingList)</div> <div>(List:Biskit;; randomList)randomListOrder(Set:Biskit;; incomingList)</div> <div>(List:Boolean randomList)randomListOrder(List:Boolean incomingList)</div> <div>(List:Boolean randomList)randomListOrder(Set:Boolean incomingList)</div> <div>(List:Date randomList)randomListOrder(List:Date incomingList)</div> <div>(List:Date randomList)randomListOrder(Set:Date incomingList)</div> <div>(List:DateRange randomList)randomListOrder(List:DateRange incomingList)</div> <div>(List:DateRange randomList)randomListOrder(Set:DateRange incomingList)</div> <div>(List:DateTime randomList)randomListOrder(List:DateTime incomingList)</div> <div>(List:DateTime randomList)randomListOrder(Set:DateTime incomingList)</div> <div>(List:Double randomList)randomListOrder(List:Double incomingList)</div> <div>(List:Double randomList)randomListOrder(Set:Double incomingList)</div> <div>(List:Enum:?: randomList)randomListOrder(List:Enum:?: incomingList)</div> <div>(List:Enum:?: randomList)randomListOrder(Set:Enum:?: incomingList)</div> <div>(List:Int randomList)randomListOrder(List:Int incomingList)</div> <div>(List:Int randomList)randomListOrder(Set:Int incomingList)</div> <div>(List:Long randomList)randomListOrder(List:Long incomingList)</div> <div>(List:Long randomList)randomListOrder(Set:Long incomingList)</div> <div>(List:String randomList)randomListOrder(List:String incomingList)</div> <div>(List:String randomList)randomListOrder(Set:String incomingList)</div> <div>(List:Biskit;; randomList)randomListOrder(List:Biskit;; incomingList)</div> </div>
Arguments	<div> <div>List:Biskit;; incomingList</div> <div>Null</div> </div>

Takes as input the list to be randomized. The input can be **Set** or a List. There are options to deal with many different data types. Returns the created the list in a random order.

remove

Function	remove
Arguments	<div> <div>(List:Biskit.typeOf:list; list)remove(List:Biskit;; list,Biskit.typeOf:list; item)</div> <div>(List:Biskit.typeOf:list; list)remove(List:Biskit;; list,Biskit.typeOf:list; item)</div> <div>(List:Boolean list)remove(List:Boolean list,Boolean item)</div> <div>(List:Date list)remove(List:Date list,Date item)</div> <div>(List:DateRange list)remove(List:DateRange list,DateRange item)</div> <div>(List:DateTime list)remove(List:DateTime list,DateTime item)</div> <div>(List:Double.typeOf:list; list)remove(List:Double list,Double.typeOf:list; item)</div> <div>(List:Int.typeOf:list; list)remove(List:Int list,Int.typeOf:list; item)</div> <div>(List:Long list)remove(List:Long list,Long item)</div> <div>(List:String.typeOf:list; list)remove(List:String list,String.typeOf:list; item)</div> </div>

Removes an item form a list, leaving the original list unchanged. Returns a list with just the item in it.

removeDuplicates

Function	removeDuplicates
Argument	<div> <div>(List:Int removeDuplicates)removeDuplicates(List:Int a)</div> <div>(List:Biskit.typeOf:a; removeDuplicates)removeDuplicates(List:Biskit;; a)</div> <div>(List:Double removeDuplicates)removeDuplicates(List:Double a)</div> <div>(List:Int removeDuplicates)removeDuplicates(List:Int a)</div> <div>(List:Long removeDuplicates)removeDuplicates(List:Long a)</div> <div>(List:String removeDuplicates)removeDuplicates(List:String a)</div> </div>

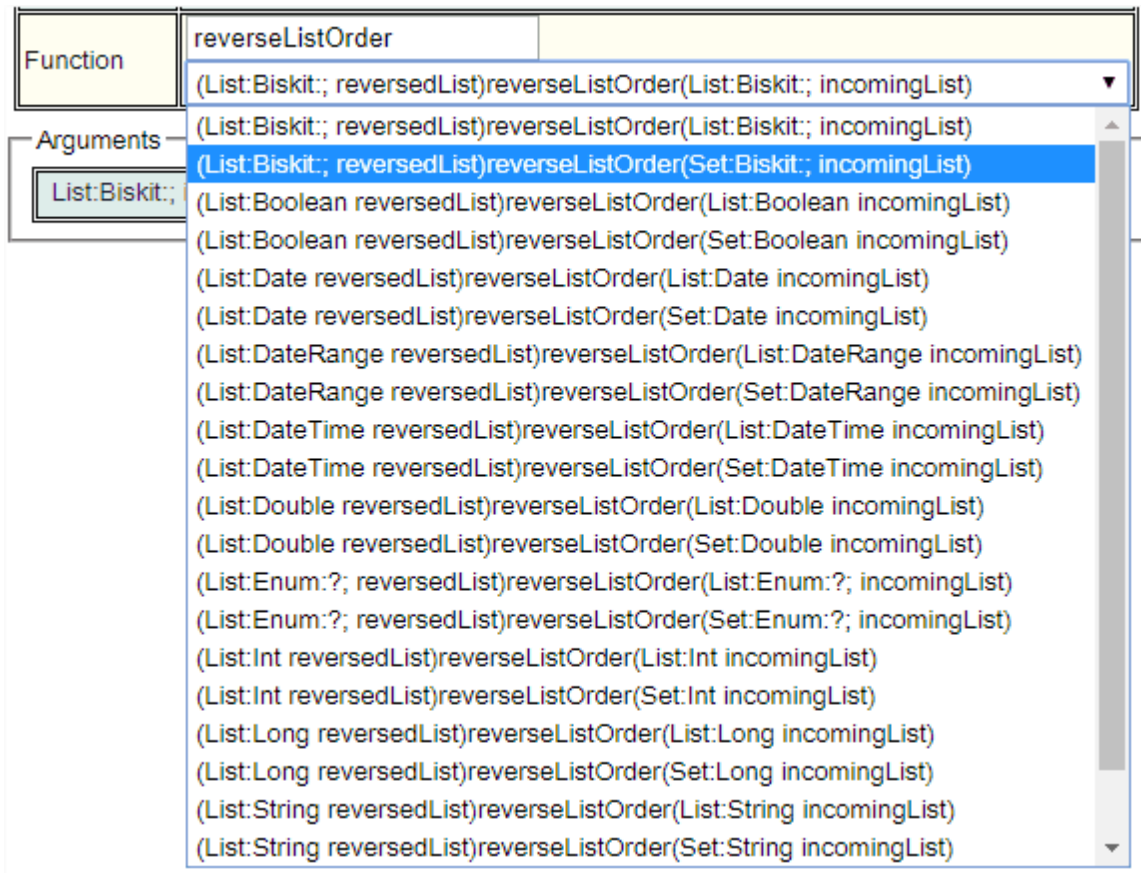
Takes as input a List of *items*. Returns a List with any duplicate *items* removed.

removeUserFromProject

Function	removeUserFromProject
Arguments	<div> <div>(Boolean removed)removeUserFromProject(Biskit:Calpendo.CalpendoUser; user,Biskit:Calpendo.Project; project)</div> <div> <div>Biskit:Calpendo.CalpendoUser; user</div> <div>Null</div> </div> <div> <div>Biskit:Calpendo.Project; project</div> <div>Null</div> </div> </div>

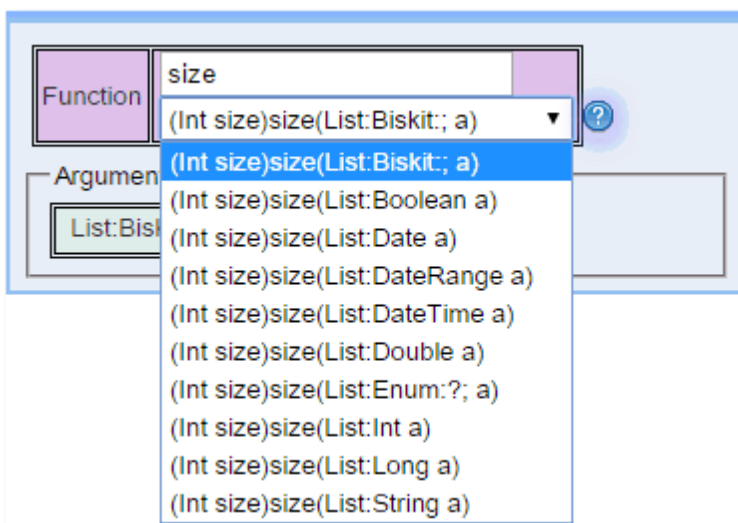
Takes as input a the *User* and the *Project*. Returns **True** if the *User* is removed from the *Project*.

reverseListOrder



Takes as input a List or Set. Returns the reversed List. Sets will be converted to a List.

size



Takes as input a list of *items*. Returns the size of the list.

subtract

Function	subtract (List:Biskit:: subtract,Int size)subtract(List:Biskit:: a,List:Biskit:: b) ▼
Arguments	
List:Biskit:: a	Null ▼
List:Biskit:: b	Null ▼

Takes the two input lists of *Biskits* and subtracts the second list from the first one. Returns a list of those *Biskits* in the first list not found in the second list and the size of the new list.

subList

Function	subList 16 functions
Arguments	(List Biskit typeOf: list; subList Biskit typeOf: list; first Biskit typeOf: list; last Biskit: size)subList(List Biskit:: list,Int fromIndex,Int toIndex) (List Biskit typeOf: list; subList Biskit typeOf: list; first Biskit typeOf: list; last Biskit: size)subList(List Biskit:: list,Int fromIndex) (List Biskit typeOf: list; subList Biskit typeOf: list; first Biskit typeOf: list; last Biskit: size)subList(List Biskit:: list,Int fromIndex,Int toIndex) (List Boolean subList,Boolean first,Boolean last,Boolean size)subList(List Boolean list,Int fromIndex) (List Boolean subList,Boolean first,Boolean last,Boolean size)subList(List Boolean list,Int fromIndex,Int toIndex) (List Date subList,Date first,Date last,Date size)subList(List Date list,Int fromIndex) (List Date subList,Date first,Date last,Date size)subList(List Date list,Int fromIndex,Int toIndex) (List DateTime subList,DateTime first,DateTime last,DateTime size)subList(List DateTime list,Int fromIndex) (List DateTime subList,DateTime first,DateTime last,DateTime size)subList(List DateTime list,Int fromIndex,Int toIndex) (List Double typeOf: list; subList Double typeOf: list; first Double typeOf: list; last Double size)subList(List Double list,Int fromIndex) (List Double typeOf: list; subList Double typeOf: list; first Double typeOf: list; last Double size)subList(List Double list,Int fromIndex,Int toIndex) (List Int typeOf: list; subList Int typeOf: list; first Int typeOf: list; last Int size)subList(List Int list,Int fromIndex) (List Int typeOf: list; subList Int typeOf: list; first Int typeOf: list; last Int size)subList(List Int list,Int fromIndex,Int toIndex) (List Long subList,Long first,Long last,Long size)subList(List Long list,Int fromIndex) (List Long subList,Long first,Long last,Long size)subList(List Long list,Int fromIndex,Int toIndex) (List String typeOf: list; subList String typeOf: list; first String typeOf: list; last String size)subList(List String list,Int fromIndex) (List String typeOf: list; subList String typeOf: list; first String typeOf: list; last String size)subList(List String list,Int fromIndex,Int toIndex)

Takes as input a list of *Biskits*, *fromIndex* (where the list should start) and optionally *toIndex* (where the list should finish). Returns a list of those *Biskits* in the list from *fromIndex* inclusive to *toIndex* exclusive. If *fromIndex* is 0 it starts at the beginning of the list, if *toIndex* is not present then it goes to the end of the list. If *fromIndex* equals *toIndex* then null is returned. 0 is the index to the first object in the list.

sum

Function	sum 4 functions
Arguments	(Double sum)sum(List:Biskit:: list,String path) ▼ (Double sum)sum(List:Biskit:: list,String path) (Double sum)sum(List:Double list) (Int sum)sum(List:Biskit:: list,String path) (Int sum)sum(List:Int list)

Takes as input a list of *Biskits* and the name of the *property* whose values are to be added up or a list of **Doubles**. Returns the sum of the values of the *property* or the list items as a **Double**. This will work on both **Double** and **Integer** value *properties* and lists.

union

Trigger Type	Function Workflow Action	
Name	New	
ID	0	
Function	union	
Arguments	<div> <div>(List:Biskit; union,Int size)union(List:Biskit; a,List:Biskit; b)</div> <div>(List:Biskit; union,Int size)union(List:Biskit; a,List:Biskit; b)</div> <div>(List:Boolean union,Int size)union(List:Boolean a,List:Boolean b)</div> <div>(List:Biskit; union,Int size)union(List:Biskit; a,List:Biskit; b)</div> <div>(List>Date union,Int size)union(List>Date a,List>Date b)</div> <div>(List:DateRange union,Int size)union(List:DateRange a,List:DateRange b)</div> <div>(List:DateTime union,Int size)union(List:DateTime a,List:DateTime b)</div> <div>(List:Double union,Int size)union(List:Double a,List:Double b)</div> <div>(List:Int union,Int size)union(List:Int a,List:Int b)</div> <div>(List:Long union,Int size)union(List:Long a,List:Long b)</div> <div>(List:String union,Int size)union(List:String a,List:String b)</div> <div>(List:StringEnum union,Int size)union(List:StringEnum a,List:StringEnum b)</div> </div>	

Takes the two input lists of *Biskits* and adds them together removing any duplicates. Returns the new list and the size of the list.

6.13.3.12.7 Logical Function Type

Function	Description
and	Returns TRUE when all values in a list of booleans are TRUE , and null if the list is empty.
if	Returns values depending on a boolean value.
nand	Returns TRUE when not all values in a list of booleans are , and null if the list is empty.
nor	Returns TRUE when none of the values in a list of booleans are TRUE , and null if the list is empty.
or	Returns TRUE when any values in a list of booleans are TRUE , and null if the list is empty.

and

Function: and
(Boolean element)and(List:Boolean list) ▾

Arguments:
List:Boolean list Null ▾

Takes as input a list of Booleans, returns the logical **and** of the list.

if

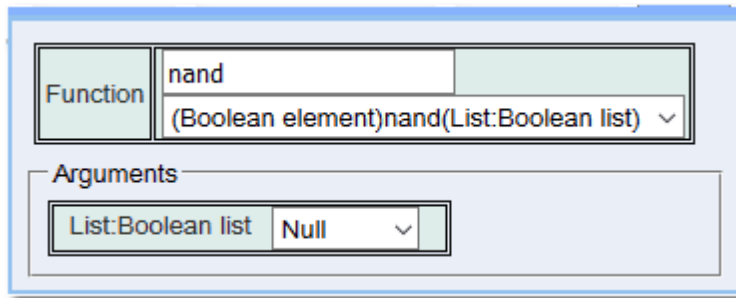
Function: if
(Biskit:typeof:value1; value)if(Boolean boolean,Biskit:; value1,Biskit:; value2) ▾

Arguments:
Boolean boolean
Biskit:; value1
Biskit:; value2

Conditional expressions in dropdown:
 (Boolean boolean, Boolean value1, Boolean value2)
 (Date value)if(Boolean boolean, Date value1, Date value2)
 (DateRange value)if(Boolean boolean, DateRange value1, DateRange value2)
 (DateTime value)if(Boolean boolean, DateTime value1, DateTime value2)
 (Double:typeof:value1; value)if(Boolean boolean, Double value1, Double value2)
 (Enum:typeof:value1; value)if(Boolean boolean, Enum:?: value1, Enum:?: value2)
 (Int:typeof:value1; value)if(Boolean boolean, Int value1, Int value2)
 (List:? value)if(Boolean boolean, List:? value1, List:? value2)
 (Long value)if(Boolean boolean, Long value1, Long value2)
 (Set:? value)if(Boolean boolean, Set:? value1, Set:? value2)
 (String:typeof:value1; value)if(Boolean boolean, String value1, String value2)
 (StringEnum value)if(Boolean boolean, StringEnum value1, StringEnum value2)

Takes as input a Boolean and the values to be returned if **True** or **False**, returns the appropriate , value1 if **True**, value2 if **False**.

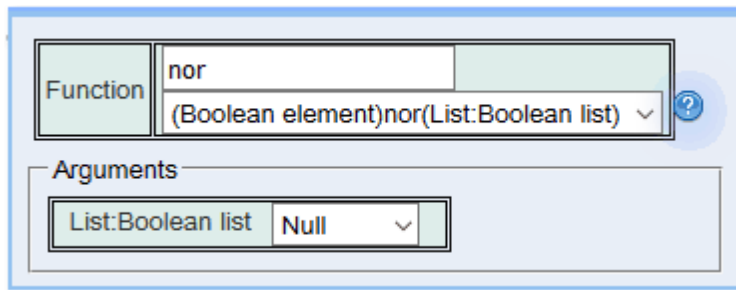
nand



The screenshot shows a configuration window for the 'nand' function. It has a light blue border. Inside, there's a 'Function' section with a text box containing 'nand' and a dropdown menu showing '(Boolean element)nand(List:Boolean list)'. Below this is an 'Arguments' section with a text box containing 'List:Boolean list' and a dropdown menu showing 'Null'.

Takes as input a list of Booleans, returns the logical **nand** of the list.

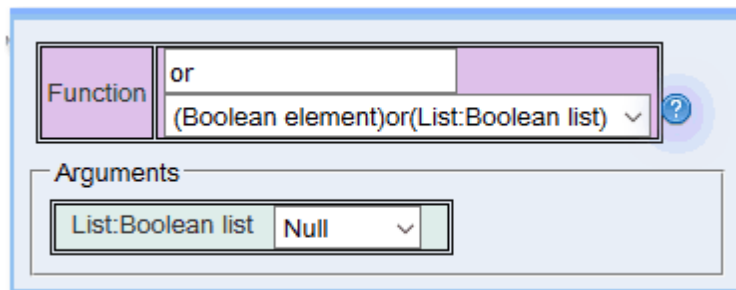
nor



The screenshot shows a configuration window for the 'nor' function. It has a light blue border. Inside, there's a 'Function' section with a text box containing 'nor' and a dropdown menu showing '(Boolean element)nor(List:Boolean list)'. Below this is an 'Arguments' section with a text box containing 'List:Boolean list' and a dropdown menu showing 'Null'.

Takes as input a list of Booleans, returns the logical **nor** of the list.

or



The screenshot shows a configuration window for the 'or' function. It has a light blue border. Inside, there's a 'Function' section with a text box containing 'or' and a dropdown menu showing '(Boolean element)or(List:Boolean list)'. Below this is an 'Arguments' section with a text box containing 'List:Boolean list' and a dropdown menu showing 'Null'.

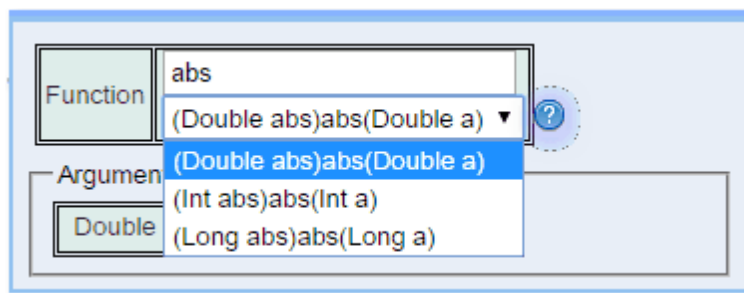
Takes as input a list of Booleans, returns the logical **or** of the list.

6.13.3.12.8 Mathematical Function Type

Function	Description
abs	Returns the absolute value of the input number.
acos	Returns the arc cosine of a value; the returned angle is in the range 0.0 through pi.
add	Adds the two inputs together and returns the result.
asin	Returns the arc sine of a value; the returned angle is in the range -pi/2 through pi/2.
atan	Returns the arc tangent of a value; the returned angle is in the range -pi/2 through pi/2.
bitwiseAnd	Performs a bitwise operation on the input. Returns the result.
ceil	Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.
cos	Returns the trigonometric cosine of an angle.
cosh	Returns the hyperbolic cosine of a double value. The hyperbolic cosine of x is defined to be $(e^x + e^{-x})/2$ where e is Euler's number.
cubeRoot	Returns the cube root of a number.
divide	Returns the result of dividing one number by another.
exp	Returns Euler's number e raised to the power of a double value.
expMinusOne	Returns $(e^x) - 1$. Note that for values of x near 0, the exact sum of $\expm1(x) + 1$ is much closer to the true result of e^x than $\exp(x)$.
floor	Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.
isNull	Returns 'value' as supplied if it is not null, otherwise returns 'nullReplacementValue'.
log	Returns the natural logarithm (base e) of a double value.
log10	Returns the base 10 logarithm of a double value.
max	Returns the largest value from two numbers.
min	Returns the smallest value (closest to negative infinity) from two numbers.
multiply	Returns the result of multiplying two numbers together.
pow	Returns the value of the first argument raised to the power of the second argument.
random	Returns a double or Integer value between the limits provided or 0.0-1.0 if no limits. The user can specify the limit range using a combination of Integers and Doubles .
remainder	Returns the remainder that you get when dividing one number by another a whole number of times.

Function	Description
round	Returns the double value that is closest in value to the argument and is equal to a mathematical integer. If two double values that are mathematical integers are equally close, the result is the integer value that is even.
sin	Returns the trigonometric sine of an angle.
sinh	Returns the hyperbolic sine of a double value. The hyperbolic sine of x is defined to be $(e^x - e^{-x})/2$ where e is Euler's number.
sqrt	Returns the correctly rounded positive square root of a double value.
subtract	Returns the value calculated when subtracting the second argument from the first.
tan	Returns the trigonometric tangent of an angle.
tanh	Returns the hyperbolic tangent of a double value. The hyperbolic tangent of x is defined to be $(e^x - e^{-x})/(e^x + e^{-x})$, in other words, $\sinh(x)/\cosh(x)$. Note that the absolute value of the exact tanh is always less than 1.
toDegrees	Converts an angle measured in radians to an approximately equivalent angle measured in degrees. The conversion from radians to degrees is generally inexact; users should not expect $\cos(\text{toRadians}(90.0))$ to exactly equal 0.0.
toRadians	Converts an angle measured in degrees to an approximately equivalent angle measured in radians. The conversion from degrees to radians is generally inexact.

abs



Returns the absolute value of a number. If the argument is not negative, the argument is returned unchanged. If the argument is negative, the negation of the argument is returned. Takes a **Double**, **Integer** or **Long** as input and returns the same type.

Special cases:

- If the argument is positive zero or negative zero, the result is positive zero.
- If the argument is infinite, the result is positive infinity.
- If the argument is **NaN**, the result is **NaN**.

acos

Function	acos (Double acos)acos(Double a) ▼
Arguments	
Double a	Null ▼

Returns the arc cosine of a value; the returned angle is in the range 0.0 through pi.

Special case:

- If the argument is **NaN** or its absolute value is greater than 1, then the result is **NaN**.

add

Function	add (Double out)add(Double a,Double b) ▼
Argument	(Double out)add(Double a,Double b)
Double	(Int add)add(Int a,Int b)
Double	(List:Biskit; add,Int size)add(List:Biskit; a,List:Biskit; b)
Double	(Long add)add(Long a,Long b)
Double	(String add)add(String a,String b)

Adds the two input values together, takes **Doubles**, **Integers** or **Longs**. Returns the result of the same type as the inputs.

(The **List:Biskit** option is described in the [List/Set Function Type](#)⁴³⁵ section, and the **String** option is described in [Text Function Type](#)⁴⁷⁴)

asin

Function	asin (Double asin)asin(Double a) ▼
Arguments	
Double a	Null ▼

Returns the arc sine of a value; the returned angle is in the range -pi/2 through pi/2.

Special cases:

- If the argument is **NaN** or its absolute value is greater than 1, then the result is .
- If the argument is zero, then the result is a zero with the same sign as the argument.

atan

The screenshot shows a configuration window for the 'atan' function. It has a 'Function' section with a text box containing 'atan' and a dropdown menu showing '(Double atan)atan(Double a)'. Below this is an 'Arguments' section with a text box containing 'Double a' and a dropdown menu showing 'Null'.

Returns the arc tangent of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$.

Special cases:

- If the argument is **NaN**, then the result is **NaN**.
- If the argument is zero, then the result is a zero with the same sign as the argument.

bitwiseAnd

The screenshot shows a configuration window for the 'bitwiseAnd' function. It has a 'Function' section with a text box containing 'bitwiseAnd' and a dropdown menu showing '(Int bitwiseAnd)bitwiseAnd(Int a)'. Below this is an 'Argument' section with a text box containing 'Int a' and a dropdown menu showing 'Null'. A dropdown menu is also visible, showing three options: '(Int bitwiseAnd)bitwiseAnd(Int a)', '(Int bitwiseAnd)bitwiseAnd(Int a)', and '(Int bitwiseAnd)bitwiseAnd(Int a,Int b)'. The first two options are highlighted in blue.

- The first option performs a bitwise **NOT** operation on an integer. This means that every binary digit that was a 0 is changed to a 1, and every 1 is changed to a 0. Returns the result.
- The second option performs a bitwise **EXCLUSIVE OR** operation on two integers. Each binary digit will be set to a 1 if the corresponding bits in the arguments are different from each other. Returns the result.

ceil

Function	ceil (Double ceil)ceil(Double a) ▼
Arguments	
Double a	Null ▼

Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.

Special cases:

- If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- If the argument is **NaN** or an infinity or positive zero or negative zero, then the result is the same as the argument.
- If the argument value is less than zero but greater than -1.0, then the result is negative zero.
- Note that the value of **ceil**(x) is exactly the value of **-floor**(-x).

COS

Function	cos (Double cos)cos(Double a) ▼
Arguments	
Double a	Null ▼

Returns the trigonometric cosine of an angle.

Special case:

- If the argument is **NaN** or an infinity, then the result is **NaN**.

cosh

Function	cosh (Double cosh)cosh(Double a) ▼
Arguments	
Double a	Null ▼

Returns the trigonometric cosine of an angle.

Special cases: If the argument is **NaN** or an infinity, then the result is **NaN**.

cubeRoot

Function	cubeRoot (Double cubeRoot)cubeRoot(Double a) ▼
Arguments	
Double a	Null ▼

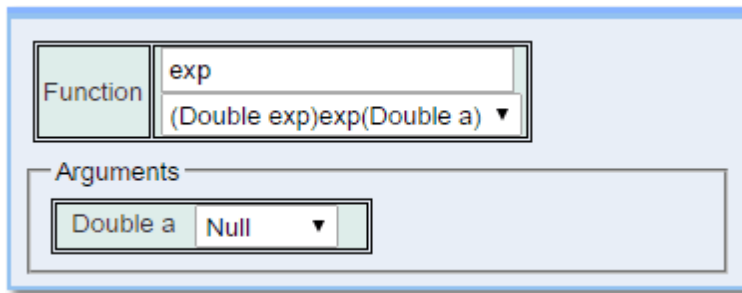
Returns the cube root of a number.

divide

Function	divide (Double out)divide(Double a,Double b) ▼ (Double out)divide(Double a,Double b) (Int divide)divide(Int a,Int b) (Long divide)divide(Long a,Long b)
Argument	
Double	
Double b	Null ▼

Returns the result of dividing one number by another. The inputs can be **Double**, **Integer** or **Long**. Returns the result of the same type as the inputs. When dividing by an Integer if the result is not an Integer it will go closer to zero. ie. 0.5 goes to 0 -0.5 goes to 0.

exp



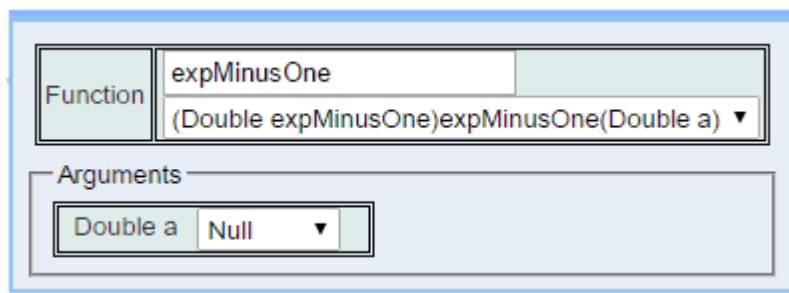
The image shows a function configuration window for the 'exp' function. It has a light blue background and a thin blue border. At the top, there is a 'Function' label in a green box, followed by a text field containing 'exp' and a dropdown menu showing '(Double exp)exp(Double a)' with a downward arrow. Below this is an 'Arguments' label in a green box, followed by a text field containing 'Double a' and a dropdown menu showing 'Null' with a downward arrow.

Returns Euler's number e raised to the power of a double value.

Special cases:

- If the argument is **NaN**, the result is **NaN**.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is negative infinity, then the result is positive zero.

expMinusOne



The image shows a function configuration window for the 'expMinusOne' function. It has a light blue background and a thin blue border. At the top, there is a 'Function' label in a green box, followed by a text field containing 'expMinusOne' and a dropdown menu showing '(Double expMinusOne)expMinusOne(Double a)' with a downward arrow. Below this is an 'Arguments' label in a green box, followed by a text field containing 'Double a' and a dropdown menu showing 'Null' with a downward arrow.

Returns $(e^x) - 1$. Note that for values of x near 0, the exact sum of $\text{expm1}(x) + 1$ is much closer to the true result of e^x than $\text{exp}(x)$.

Special cases:

- If the argument is **NaN**, the result is **NaN**.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is negative infinity, then the result is -1.0.
- If the argument is zero, then the result is a zero with the same sign as the argument.

floor

Function	floor (Double floor)floor(Double a) ▼		
Arguments	<table border="1"><tr><td>Double a</td><td>Null ▼</td></tr></table>	Double a	Null ▼
Double a	Null ▼		

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.

Special cases:

- If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- If the argument is **NaN** or an infinity or positive zero or negative zero, then the result is the same as the argument.

isNull

Function	isNull (Double out)isNull(Double value,Double nullReplacementValue) ▼ (Double out)isNull(Double value,Double nullReplacementValue) (Int isNull)isNull(Int value,Int nullReplacementValue) (Long isNull)isNull(Long value,Long nullReplacementValue)			
Argument	<table border="1"><tr><td>Double</td><td>Double nullReplacementValue</td><td>Null ▼</td></tr></table>	Double	Double nullReplacementValue	Null ▼
Double	Double nullReplacementValue	Null ▼		

Returns '**value**' as supplied if it is not **null**, otherwise returns '**nullReplacementValue**'. The input can be a **Double**, **Integer** or **Long**.

log

The screenshot shows a configuration window for the 'log' function. It has a 'Function' section with a text box containing 'log' and a dropdown menu showing '(Double log)log(Double a)'. Below this is an 'Arguments' section with a text box containing 'Double a' and a dropdown menu showing 'Null'.

Returns the natural logarithm (base e) of a double value.

Special cases:

- If the argument is **NaN** or less than zero, then the result is .
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is positive zero or negative zero, then the result is negative infinity.

log10

The screenshot shows a configuration window for the 'log10' function. It has a 'Function' section with a text box containing 'log10' and a dropdown menu showing '(Double log10)log10(Double a)'. Below this is an 'Arguments' section with a text box containing 'Double a' and a dropdown menu showing 'Null'.

Returns the base 10 logarithm of a double value.

Special cases:

- If the argument is **NaN** or less than zero, then the result is **NaN**.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is positive zero or negative zero, then the result is negative infinity.
- If the argument is equal to 10^n for integer n , then the result is n .

max

Function	max
Argument	(Double out)max(Double a,Double b) ▼ (Double out)max(Double a,Double b) (Int max)max(Int a,Int b) (Long max)max(Long a,Long b)
Double a	Null ▼
Double b	Null ▼

Returns the largest value from two numbers. The inputs can be **Doubles**, **Integers** or **Longs**. Returns the result of the same type as the inputs.

min

Function	min
Argument	(Double out)min(Double a,Double b) ▼ (Double out)min(Double a,Double b) (Int min)min(Int a,Int b) (Long min)min(Long a,Long b)
Double a	Null ▼
Double b	Null ▼

Returns the smallest value (closest to negative infinity) from two numbers. The inputs can be **Doubles**, **Integers** or **Longs**. Returns the result of the same type as the inputs.

multiply

Function	multiply
Argument	(Double out)multiply(Double a,Double b) ▼ (Double out)multiply(Double a,Double b) (Int multiply)multiply(Int a,Int b) (Long multiply)multiply(Long a,Long b)
Double a	Null ▼
Double b	Null ▼

Returns the result of multiplying two numbers together. The inputs can be **Doubles**, **Integers** or **Longs**. Returns the result of the same type as the inputs.

pow

The screenshot shows a configuration window for the 'pow' function. It has a 'Function' section with a text field containing 'pow' and a dropdown menu showing '(Double out)pow(Double a,Double b)'. Below this is an 'Arguments' section with two rows. The first row is 'Double a' with a dropdown menu set to 'Null'. The second row is 'Double b' with a dropdown menu set to 'Null'.

Returns the value of the first argument raised to the power of the second argument.

Special cases:

- If the second argument is positive or negative zero, then the result is 1.0.
- If the second argument is 1.0, then the result is the same as the first argument.
- If the second argument is **NaN**, then the result is **NaN**.
- If the first argument is **NaN** and the second argument is non zero, then the result is **NaN**.
- If
 - the absolute value of the first argument is greater than 1 and the second argument is positive infinity, or
 - the absolute value of the first argument is less than 1 and the second argument is negative infinity,

then the result is positive infinity.

- If
 - the absolute value of the first argument is greater than 1 and the second argument is negative infinity, or
 - the absolute value of the first argument is less than 1 and the second argument is positive infinity,

then the result is positive zero.

- If the absolute value of the first argument equals 1 and the second argument is infinite, then the result is **NaN**.
- If
 - the first argument is positive zero and the second argument is greater than zero, or
 - the first argument is positive infinity and the second argument is less than zero,

then the result is positive zero.

- If

- the first argument is positive zero and the second argument is less than zero, or
- the first argument is positive infinity and the second argument is greater than zero,

then the result is positive infinity.

- If

- the first argument is negative zero and the second argument is greater than zero but not a finite odd integer, or
- the first argument is negative infinity and the second argument is less than zero but not a finite odd integer,

then the result is positive zero.

- If

- the first argument is negative zero and the second argument is a positive finite odd integer, or
- the first argument is negative infinity and the second argument is a negative finite odd integer,

then the result is negative zero.

- If

- the first argument is negative zero and the second argument is less than zero but not a finite odd integer, or
- the first argument is negative infinity and the second argument is greater than zero but not a finite odd integer,

then the result is positive infinity.

- if

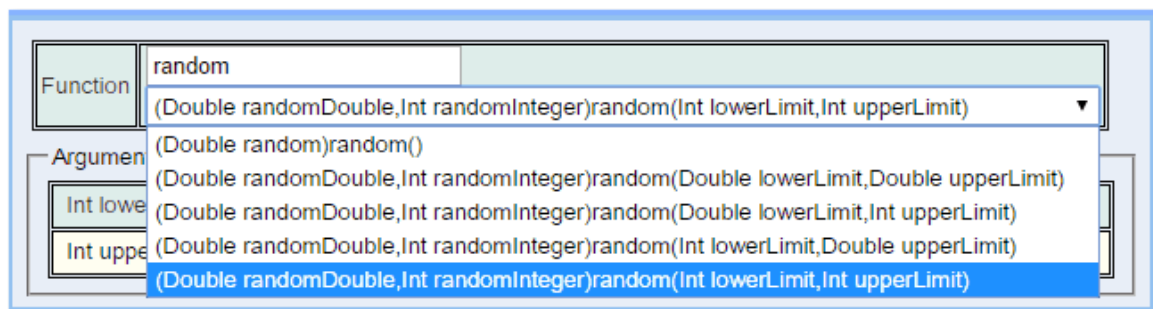
- the first argument is negative zero and the second argument is a negative finite odd integer, or
- the first argument is negative infinity and the second argument is a positive finite odd integer,

then the result is negative infinity.

- If the first argument is finite and less than zero
 - if the second argument is a finite even integer, the result is equal to the result of raising the absolute value of the first argument to the power of the second argument
 - if the second argument is a finite odd integer, the result is equal to the negative of the result of raising the absolute value of the first argument to the power of the second argument
 - if the second argument is finite and not an integer, then the result is NaN.
 - If both arguments are integers, then the result is exactly equal to the mathematical result of raising the first argument to the power of the second argument if that result can in fact be represented exactly as a double value.

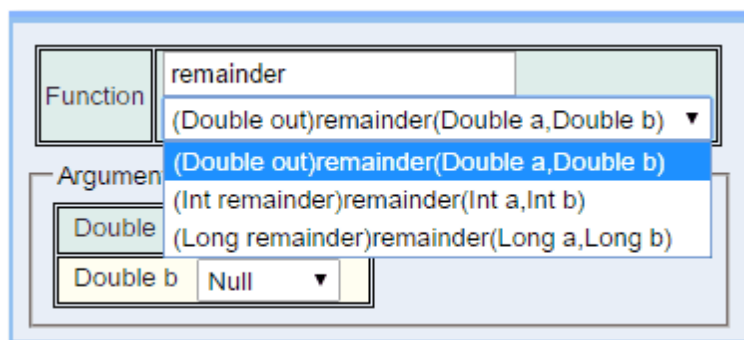
In the foregoing descriptions, a floating-point value is considered to be an integer if and only if it is finite and a fixed point of the method **ceil** or, equivalently, a fixed point of the method **floor**. A value is a fixed point of a one-argument method if and only if the result of applying the method to the value is equal to the value.

random



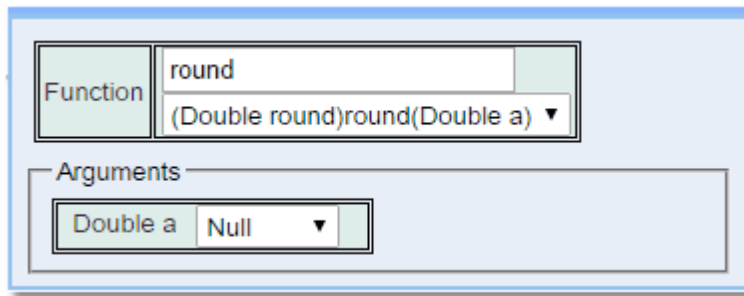
Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudo-randomly with (approximately) uniform distribution from that range. Alternatively the range can be defined and the returned random number can be either double or integer. The integer value returned is rounded close to zero. So a double value of 1.5 is rounded to 1 and -1.5 is rounded to -1.

remainder



Returns the remainder that you get when dividing one number by another a whole number of times. The input can be a **Doubles**, **Integers** or **Longs**. Returns the result of the same type as the inputs.

round



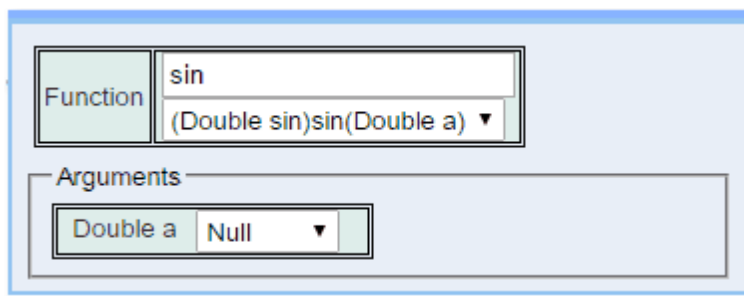
The image shows a configuration window for the 'round' function. It has a 'Function' section with a text box containing 'round' and a dropdown menu showing '(Double round)round(Double a)'. Below this is an 'Arguments' section with a text box containing 'Double a' and a dropdown menu showing 'Null'.

Returns the double value that is closest in value to the argument and is equal to a mathematical integer. If two double values that are mathematical integers are equally close, the result is the integer value that is even.

Special cases:

- If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- If the argument is **NaN** or an infinity or positive zero or negative zero, then the result is the same as the argument.

sin



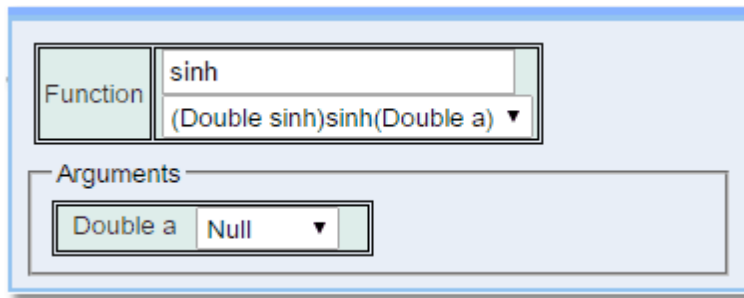
The image shows a configuration window for the 'sin' function. It has a 'Function' section with a text box containing 'sin' and a dropdown menu showing '(Double sin)sin(Double a)'. Below this is an 'Arguments' section with a text box containing 'Double a' and a dropdown menu showing 'Null'.

Returns the trigonometric sine of an angle.

Special cases:

- If the argument is **NaN** or an infinity, then the result is **NaN**.
- If the argument is zero, then the result is a zero with the same sign as the argument.

sinh



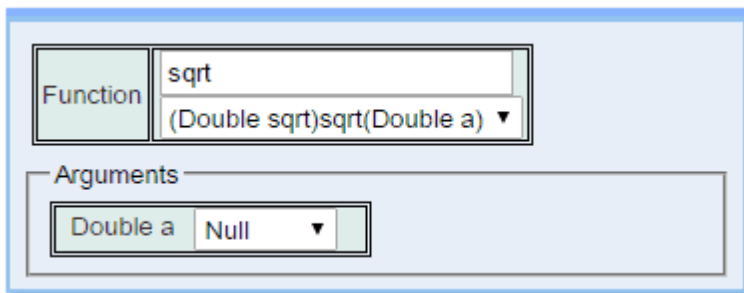
The image shows a configuration window for the 'sinh' function. It has a 'Function' section with a text box containing 'sinh' and a dropdown menu showing '(Double sinh)sinh(Double a)' with a downward arrow. Below this is an 'Arguments' section with a text box containing 'Double a' and a dropdown menu showing 'Null' with a downward arrow.

Returns the hyperbolic sine of a double value. The hyperbolic sine of x is defined to be $(e^x - e^{-x})/2$ where e is Euler's number.

Special cases:

- If the argument is **NaN**, then the result is .
- If the argument is infinite, then the result is an infinity with the same sign as the argument.
- If the argument is zero, then the result is a zero with the same sign as the argument.

sqrt



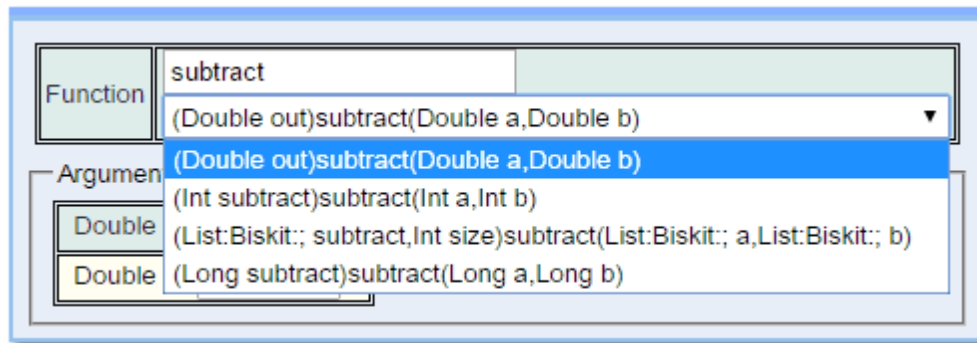
The image shows a configuration window for the 'sqrt' function. It has a 'Function' section with a text box containing 'sqrt' and a dropdown menu showing '(Double sqrt)sqrt(Double a)' with a downward arrow. Below this is an 'Arguments' section with a text box containing 'Double a' and a dropdown menu showing 'Null' with a downward arrow.

Returns the correctly rounded positive square root of a double value.

Special cases:

- If the argument is **NaN** or less than zero, then the result is **NaN**.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is positive zero or negative zero, then the result is the same as the argument.

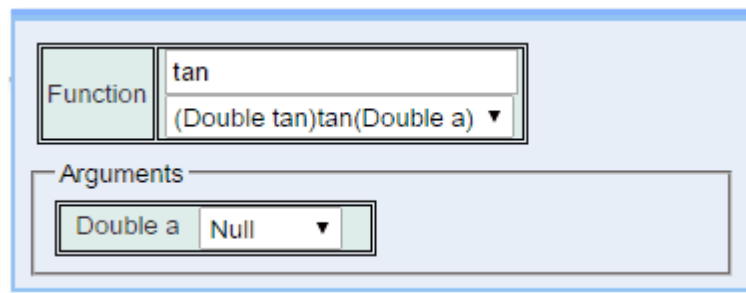
subtract



Subtracts the second argument from the first. Takes **Doubles**, **Integers** or **Longs** as inputs. Returns the result of the same type as the inputs.

(The **List:Biskit** option is described in the [List/Set Function Type](#)⁴³⁵ section)

tan



Returns the trigonometric tangent of an angle.

Special cases:

- If the argument is **NaN** or an infinity, then the result is **NaN**.
- If the argument is zero, then the result is a zero with the same sign as the argument.

tanh

Function	tanh (Double tanh)tanh(Double a) ▼
Arguments	
Double a	Null ▼

Returns the hyperbolic tangent of a double value. The hyperbolic tangent of x is defined to be $(e^x - e^{-x}) / (e^x + e^{-x})$, in other words, **sinh**(x)/**cosh**(x). Note that the absolute value of the exact **tanh** is always less than 1.

Special cases:

- If the argument is **NaN**, then the result is **NaN**.
- If the argument is zero, then the result is a zero with the same sign as the argument.
- If the argument is positive infinity, then the result is +1.0.
- If the argument is negative infinity, then the result is -1.0.

toDegrees

Function	toDegrees (Double toDegrees)toDegrees(Double a) ▼
Arguments	
Double a	Null ▼

Converts an angle measured in radians to an approximately equivalent angle measured in degrees. The conversion from radians to degrees is generally inexact; users should not expect **cos**((90.0)) to exactly equal 0.0.

toRadians

Function	toRadians (Double toRadians)toRadians(Double a) ▼
Arguments	
Double a	Null ▼

Converts an angle measured in degrees to an approximately equivalent angle measured in radians. The conversion from degrees to radians is generally inexact.

6.13.3.12.9 Miscellaneous Function Type

Function	Description
addUserLayout	Adds a layout that will be used by a user in preference to any standard layouts
assertActiveUser	Specifies who the current active user is.
authenticate	Authenticates a users password.
generateUserWorkflowEventURL	Allows links to be emailed that will cause a UserWorkflowEvent to be run.
getUserInfo	Returns whether a user can login, and whether they can receive email.
logSystemEvent	Stores a new system event.
randomDistinct	Produces a string of random characters of a given length, using only upper case letters or numbers that cannot be mistaken for other letters or numbers. That means we do not use numbers 0 or 1, or letters i, L, O, U or V
randomURL	Produces a string of random characters of a given length, using only characters that are suitable for use in a URL.
sleep	Pauses execution for a given number of milliseconds.
warn	Sends a warning to the workflow user.

addUserLayout

Function

addUserLayout

1 functions

()addUserLayout(Biskit:BiskitLayoutRoot; layout,Biskit:UserLayouts; layouts) ▼

Arguments

Biskit:BiskitLayoutRoot; layout

Null ▼

Biskit:UserLayouts; layouts

Null ▼

Takes as input the the layouts a user will be given, and all the layouts for a user as returned from a UserLoginWorkflowEvent. Returns nothing.

Can only be used as part of a UserLoginWorkflowEvent, and the function will add a layout to those that the user will see. Layouts assigned here will override any *Permissions* set up for layouts with reference to this user.

This can be used to specify particular layouts for a user to control the way data is presented.

For example, the existence of some properties may be hidden from some users, rather than only hiding the value of those properties.

assertActiveUser

Function	assertActiveUser ()assertActiveUser(Biskit:Calpendo.CalpendoUser; activeUser) ▼
Arguments	Biskit:Calpendo.CalpendoUser; activeUser Null ▼

Takes as input the currently active user. Returns nothing.

This is useful in situations where the original event is not triggered directly by a user, but the user can be worked out by other means. For example, if there is an anonymous HTTP event triggered from a card reader, and the user can be identified from their card.

If this function is called, then anything created or updated that records who the user is, will use the newly asserted active user.

If the initiating event can identify the active user, then calling this function will not change the active user. Moreover, trying to change the active user, and set a different user from the one the event knows it to be, will be treated as an error.

authenticate

Function	authenticate (Boolean authenticatedOkay)authenticate(Biskit:Calpendo.CalpendoUser; user,String password) ▼				
Arguments	<table border="1"> <tr> <td>Biskit:Calpendo.CalpendoUser; user</td> <td>Null ▼</td> </tr> <tr> <td>String password</td> <td>Null ▼</td> </tr> </table>	Biskit:Calpendo.CalpendoUser; user	Null ▼	String password	Null ▼
Biskit:Calpendo.CalpendoUser; user	Null ▼				
String password	Null ▼				

Takes as input the user [Biskit](#)⁶⁵² and the password. Returns a boolean.

generateUserWorkflowEventURL

Function	generateUserWorkflowEvent 1 functions ? (String url)generateUserWorkflowEventURL(Biskit:UserWorkflowEvent; userWorkflowEvent,Biskit:: biskit,String state) ▼						
Arguments	<table border="1"> <tr> <td>Biskit:UserWorkflowEvent; userWorkflowEvent</td> <td>Null ▼</td> </tr> <tr> <td>Biskit:: biskit</td> <td>Null ▼</td> </tr> <tr> <td>String state</td> <td>Null ▼</td> </tr> </table>	Biskit:UserWorkflowEvent; userWorkflowEvent	Null ▼	Biskit:: biskit	Null ▼	String state	Null ▼
Biskit:UserWorkflowEvent; userWorkflowEvent	Null ▼						
Biskit:: biskit	Null ▼						
String state	Null ▼						

Takes as input the user event to be triggered, the [Biskit](#)⁶⁵² that is encoded into the date and a string to be passed into the event that can be used to pass information to the event. Returns a string which is the URL that when accessed will trigger the user workflow event.

getUserInfo

Function	getUserInfo (Boolean canLogin, Boolean canReceiveEmail)getUserInfo(Biskit:Calpendo.CalpendoUser; user) ▼
Arguments	
Biskit:Calpendo.CalpendoUser; user	Null ▼

Takes as input the user *Biskit*. Returns a boolean defining whether a user can login and a boolean defining whether they can receive email.

logSystemEvent

Function	logSystemEvent (Biskit:ExprodoEvent; systemEvent)logSystemEvent(Enum:ExprodoEvent\$Type; type,String category,String message,Biskit; biskit) ▼
Arguments	
Enum:ExprodoEvent\$Type; type	Null ▼
String category	Null ▼
String message	Null ▼
Biskit; biskit	Null ▼

Takes as input the type of the Event, the category for this type of event, the event message and a *Biskit* to be associated with the event. Returns the system event that was created.

randomDistinct

Function	randomDistinct (String randomDistinct)randomDistinct(Int length) ▼
Arguments	
Int length	Null ▼

Takes as input the length of the **String** to be produced. Returns the created **String**.

randomURL

Function	randomURL (String randomURL)randomURL(Int length) ▼
Arguments	
Int length	Null ▼

Takes as input the length of the **String** to be produced. Returns the created **String**.

sleep

Function	sleep (Int sleep)sleep(Int millis) ▼
Arguments	
Int millis	Null ▼

Takes as input the number of milliseconds to sleep. Returns the initial argument.

warn

Function	warn (Boolean delivered)warn(String message) ▼
Arguments	
String message	Null ▼

Takes as input the message to be sent. Returns whether the warning was delivered.

The workflow must be directly connected to a user's session to be able to do this. So will not work with timed and other types of **Events** which are not being directly run by a user.

6.13.3.12.10 Network Function Type

Function	Description
IO24TCP_getStatus	Returns the status of an EloxeI IO24 TCP . See www.elexol.com for more.
IO24TCP_identity	Returns the status of an MAC address and firmware version of an EloxeI IO24 TCP . See www.elexol.com for more.
IO24TCP_setPin	Sets a single output pin on an EloxeI IO24 TCP . See www.elexol.com for more.
IO24TCP_setPort	Sets the 8-bit value of a whole port on an EloxeI IO24 TCP . See www.elexol.com for more.
LDAPSearch	Searches the LDAP Server.
getCookie	Returns the value of a particular Cookie .
ping	Send a ping to the specified address.

IO24TCP_getStatus

Function	IO24TCP_getStatus (Int valueA,Int valueB,Int valueC,Int directionA,Int directionB,Int directionC,Int pin00,Int pin01,Int pin02,Int pin03,Int pin04,Int pin05,Int pin06,Int pin07,Int pin08,Int pin09,Int pin10,Int pin11,Int pin12,Int pin13,Int pin14,Int pin15,Int pin16,Int pin17,Int pin18,Int pin19,Int pin20,Int pin21,Int pin22,Int pin23)IO24TCP_getStatus(String ipAddress,Int networkPortNumber)
Arguments	<div>String ipAddress Null</div> <div>Int networkPortNumber Null</div>

Takes as input the **IP Address** of the device sending the communication and the **Port Number** the data is being received on. This defaults to 2424.

Output	Description
valueA, valueB, valueC	Each is an 8 bit value showing the current state of 8 pins. A 0-7, B 8-15, C 9-23.
directionA, directionB, directionC	Each is an 8 bit value showing for 8 pins whether they are Input or Output . 0 is Output 1 is Input
pin00-pin23	The current state of an individual pin, 0 or 1.

IO24TCP_identity

The screenshot shows a configuration window for the **IO24TCP_identity** function. The **Function** dropdown is set to **IO24TCP_identity(String ipAddress,Int firmwareVersion)IO24TCP_identity(String ipAddress,Int networkPortNumber)**. Below, the **Arguments** section contains two rows: **String ipAddress** with a **Null** dropdown, and **Int networkPortNumber** with a **Null** dropdown.

Takes as input the **IPAddress** of the device sending the communication and the **Port Number** the data is being received on. This defaults to 2424.

Output	Description
macAddress	The Mac Address as a String .
firmwareVersion	The firmware version as an Integer

IO24TCP_setPin

The screenshot shows a configuration window for the **IO24TCP_setPin** function. The **Function** dropdown is set to **()IO24TCP_setPin(String ipAddress,Int networkPortNumber,Int pinNumber,Boolean raise)**. Below, the **Arguments** section contains four rows: **String ipAddress** with a **Null** dropdown, **Int networkPortNumber** with a **Null** dropdown, **Int pinNumber** with a **Null** dropdown, and **Boolean raise** with a **Null** dropdown.

Takes as input the **IPAddress** of the device sending the communication and the **Port Number** the data is being received on. This defaults to 2424. Also the pin number to be set and the raise, 0 or 1 to set the pin to. Returns nothing.

IO24TCP_setPort

Function	IO24TCP_setPort (()IO24TCP_setPort(String ipAddress,Int networkPortNumber,Int portNumber,Int value) ▾
Arguments	
String ipAddress	Null ▾
Int networkPortNumber	Null ▾
Int portNumber	Null ▾
Int value	Null ▾

Takes as input the **IPAddress** of the device sending the communication and the **Port Number** the data is being received on. This defaults to 2424. Also the port group to be set (0=A, 1=B, 2=C) and the value which is an 8 bit number, 0 or 1 for each pin that is being set. Returns nothing.

getCookie

Function	getCookie (String cookieValue,Boolean cookieFound)getCookie(String name) ▾ (String cookieValue,Boolean cookieFound)getCookie(String name) (String cookieValue,Boolean cookieFound)getCookie(String name,String path)
Argument	String name Null ▾

Takes as input the name of the **Cookie** and optionally the path to that **Cookie**. Returns the value of a particular **Cookie**. If multiple **Cookies** match the name (which can happen if they have different paths), then one of them is returned at random. This can only be run when triggered by an anonymous http request.

LDAPSearch

Function	LDAPSearch (List:Biskit:LDAPSearchResult; biskitList)LDAPSearch(Biskit;; LDAPServer,String baseDN,String filter,Enum:LDAPSearchScope; scope) ▾
Arguments	
Biskit;; LDAPServer	Null ▾
String baseDN	Null ▾
String filter	Null ▾
Enum:LDAPSearchScope; scope	Null ▾

Takes as input a Biskit pointing to the server, the point from which the search is started, a string defining the filter to be used, and the scope of the search. Returns a list of results.

An LDAP filter, as defined by RFC2254, used for restricting which entries in the LDAP server are visible during authentication. If brackets are not supplied (like this) around the filter, then they will be added automatically.

For example:

(| (foo=123)(bar=37))

Or

foo=123

Note that a filter must be provided.

ping

Function	ping (Boolean success)ping(String address,Int timeoutMillis) ▾
Arguments	
String address	Null ▾
Int timeoutMillis	Null ▾

Takes as input address, which is either the IP address or name of the device to be contacted, and the timeoutMills being how long the maximum wait for a response is. Returns a boolean value defining whether the pong was successful.

6.13.3.12.11 Text Function Type

Function	Description
add	Appends one string onto the end of another.
concatenate	Appends one string onto the end of one or more others.
csvDecode	Decodes a string of comma-separated values and returns a list of Double.
csvEncode	Encodes a list and returns a CSV string.
strlen	Returns the length of the string
substring	Returns a substring of a string.

add

Function: add
(String add)add(String a,String b)

Arguments:

- String a: Null
- String b: Null

Takes two **Strings** as input, appends the second **String** onto the end of the first one. Returns the new **String**.

concatenate

Function: concatenate
(String concatenate)concatenate(String a,String b)

Arguments:

- String a: Null
- String b: Null

Takes two **Strings** as input, appends the second **String** onto the end of the first one. There are versions of this function that allow up to five strings to be concatenated. Returns the new **String**.

csvDecode

Function: csvDecode
(List:Double values)csvDecode(String csv)

Arguments:

- String csv: Null

Takes a **String** of comma-separated values, decodes them and returns a list of **Doubles**.

csvEncode

Function	csvEncode
	14 functions
	(String csv)csvEncode(List: Boolean values, Boolean quoteItems) ▼
Arguments	(String csv)csvEncode(List: Boolean values)
List: Boolean v	(String csv)csvEncode(List: Boolean values, Boolean quoteItems)
Boolean quote	(String csv)csvEncode(List: Double values)
	(String csv)csvEncode(List: Double values, Boolean quoteItems)
	(String csv)csvEncode(List: Enum: ?; values)
	(String csv)csvEncode(List: Enum: ?; values, Boolean quoteItems)
	(String csv)csvEncode(List: Int values)
	(String csv)csvEncode(List: Int values, Boolean quoteItems)
	(String csv)csvEncode(List: Long values)
	(String csv)csvEncode(List: Long values, Boolean quoteItems)
	(String csv)csvEncode(List: String values)
	(String csv)csvEncode(List: String values, Boolean quoteItems)
	(String csv)csvEncode(List: StringEnum values)
	(String csv)csvEncode(List: StringEnum values, Boolean quoteItems)

Takes a List of values, optionally whether all items should be quoted regardless of content or when necessary, and returns a string of comma separated values.

strlen

Function	strlen
	1 functions
	(Int length)strlen(String text) ▼
Arguments	
String text	Null ▼

Takes a string, and returns its length.

substring

Function	substring
	3 functions
	(String substring)substring(String string, Int beginIndex, Int length, Boolean autoShorten) ▼
Arguments	
String string	Null ▼
Int beginIndex	Null ▼
Int length	Null ▼
Boolean autoShorten	Null ▼

Takes a string and the index of the start character, optionally the number of characters and whether to autoshorten, and returns a substring of the original.

6.13.3.13 List Extract Workflow Action

Returns a list of [property](#)⁶⁵⁵ values extracted from the input list.

Takes as input a list of [Biskits](#)⁶⁵², and the *property* whose value needs to be extracted from each *Biskit*.

Returns a **list**, which holds the extracted list for use with subsequent [Actions](#)⁶⁵⁶.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
list	The extracted list.

6.13.3.14 Network Message Workflow Action

Creates a **Network Message** and sends it.

Takes as input the information required to create the information for the server request.

Input	Description
Server	The name of the server to be contacted.
User	The user name used for authentication.
Password	The password used for authentication.
Port	Which port to connect to, usually 80.
Protocol	The protocol to be used: HTTP, HTTPS or RAW
HTTP Method	The HTTP method to be used:

Input	Description
	GET, PUT, DELETE, HEAD or POST
URL Path	The path to the URL to be used.
Timeout Millis	How long the connection will be kept open before timing out.
Request Character Encoding	The encoding for the characters in the request.
Response Character Encoding	The encoding for the characters in the response.

The HTTP header fields are transmitted after the request or response line, which is the first line of a message. Header fields are colon-separated name-value pairs in clear-text string format, terminated by a carriage return (CR) and line feed (LF) character sequence. The end of the header section is indicated by an empty field, resulting in the transmission of two consecutive CR-LF pairs.

Returns **request**, **response** which hold information about the message and **code** which holds the messages code as a Mapped Int, for use with subsequent [Actions](#)⁶⁵⁶ which holds information about the email.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
request response	Information about the request and the response.
code	The code of the message.

6.13.3.15 Search Workflow Action

Searches for [Biskits](#)⁶⁵² of the input [BiskitDef](#)⁶⁵² to be searched for, that match the [Conditions](#)⁶⁵³ defined or it can be asked to further refine a previously created list.

Takes as input either nothing (**New Search**) or the list to be refined (**Refine an existing collection**). The maximum number of results may also be defined. If using this for a non-enumerable BiskitDef then conditions are not required.

When creating the conditions for the **Search** the initial part of the search condition may now hold a reference to a value stored in a previous **Action/Event**.

Thus conditions like this can be used:

```
any of the following are true
  all of the following are true
    Source #1 sex equals "Male"
    Value of sex equals Variable Source #1 new.sex
  all of the following are true
    Source#2 sex equals "female"
    Value of sex equals Variable Source #2 new.sex
```

When making the comparisons the **Search** Value must come first and then what you are comparing it against.

Returns **listSize** the number of *Biskits* found and **list** which holds all the found *Biskits* for use with subsequent [Actions](#)⁶⁵⁶. Use [ForEach](#)³⁸⁷, [ListExtract](#)⁴⁷⁶, **Search** or [Sort](#)⁴⁷⁹ *Actions* on the returned **list**.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
first	The first <i>Biskit</i> found in the search
last	The last <i>Biskit</i> found in the search
listSize	How many <i>Biskits</i> found in the search.
list	The list of <i>Biskits</i> found

6.13.3.16 Simple Workflow Action

Has no input, useful as a way of splitting the flow of a [Workflow](#)⁶⁵⁶ based on the conditions placed on the [Action](#)⁶⁵⁶. Used to make the equivalent of **If** or **Case** statements.

Properties Available for Child Actions To Use

There are no additional properties available for use in any child or descendant action beyond the standard set.

6.13.3.17 Sort Workflow Action

Sorts a list. Specify the list to be sorted and the **Randomisation Type** required

Function Type	Description
Any	Leaves the order as is.
Random	Sorts in a random order.
Weighted Random	Sorts in a random order with weight given by a property.
Sorted	Standard sort.

For any sort other than **Any** then define whether the **Path** is going to be **Direct**, **Indirect(select highest weight)** or **Indirect(select lowest weight)**, set up the **Sort Order**, **Increasing** or **Decreasing** and specify the [property](#)⁶⁵⁵ which will define how the list is sorted. This can be of type **Integer**, **Date**, **String** or **JavaEnum**. If the **Path** is set to be **Indirect** the user will also be asked for the **Referee Path** to find the **Weight Path**. Finally once the **Weight Path** is set up the user will be asked to define a **Default Weight** for those items that do not have one. When dealing with dates the one closest to **Now** has the lowest weight whether its in the future or the past.

Collection to Sort	Variable ▾	Source #3 ▾	list
Randomisation Type	WeightedRandom ▾		
Direct Path	Indirect Path (select lowest weight) ▾		
Sort Order	Decreasing ▾		
Referee Path	Calpendo.Booking.project		
Weight Path	dateRange.start.month		
Default Weight	Specified value ▾		

Returns **size** the number of objects in the list and **list** which is the sorted list for use with subsequent [Actions](#)⁶⁵⁶. Use [ForEach](#)³⁸⁷ or [ListExtract](#)⁴⁷⁶ on the returned **list**.

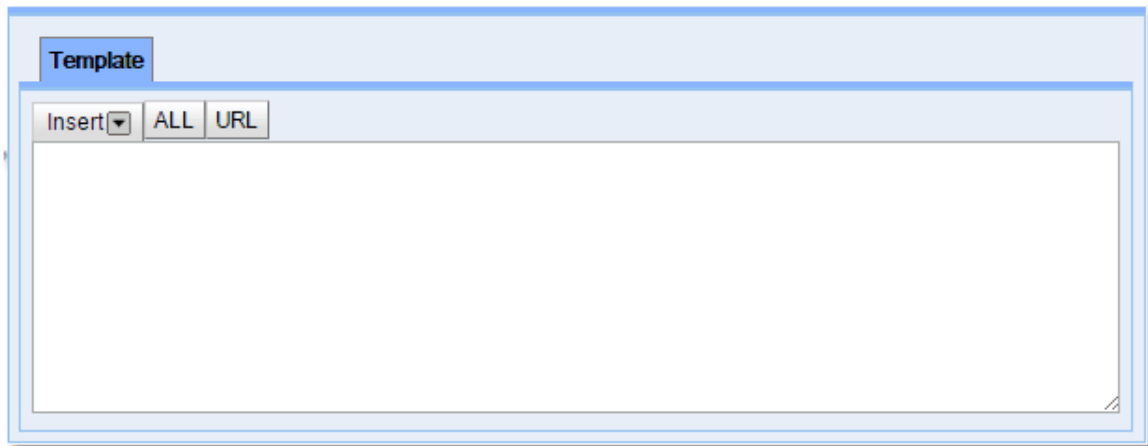
Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
size	How many <i>Biskits</i> in the sort.
list	The sorted list.

6.13.3.18 Templated Text Workflow Action

Allows the user to create a text string from multiple inputs to be used by another [Action](#)⁶⁵⁶. Use the Insert option to insert text either as **HTML**, **JavaScript** or data from previous *Actions*. In addition the user can type in any connecting text between inputs. **ALL** will insert all [properties](#)⁶⁵⁵, this will be very large. **URL** will insert the **URL** that points at this application. Data is inserted into a **Templated Text** based on the roles and/or user type specified by the action. Unlike **Email Workflow Action** and **Network Message Workflow Action** which apply the *nobody* check. This means that otherwise hidden data can easily appear in an email by creating a the body of the email using a **Templated Text Workflow Action** and then using the result as input into the **Email Workflow Action**.



The **Templated Text Workflow Action** can also parse **FreeMarker** directives. If there are any **FreeMarker** directives then the template is also run through **FreeMarker**.

There are values provided for the **FreeMarker** data model, which can all be accessed using standard **FreeMarker** syntax. The UI's "Insert" button has a new item that appears in the drop-down. One for selecting a value that **FreeMarker** will parse, and another for creating a **FreeMarker** loop through a list of *biskits*.

For example, this is a valid **FreeMarker** template, where event #1 is a database event for a *booking* being created:

```
○ Hello there ${METAPROPERTIES.user.userIdentity.loginName}, booking
  made for ${source1.biskit.resource.name}
```

Returns **value** for use with subsequent *Actions* which holds the created text **String**.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
value	The created text string.

6.13.3.19 Type Cast Workflow Action

Checks whether a [Biskit](#)⁶⁵² has a particular subtype, and it only runs the child actions if the *Biskit* does have that subtype. The child actions can then treat the *Biskit* as having that subtype, which means you have access to all the [properties](#)⁶⁵⁵ that exist on the subtype. Choose the *Biskit* to test and the [Biskit Type](#)⁶⁵² to cast it to.

Biskit to Test	Fixed ▼	Please select a Biskit Def
Type To Cast To	Fixed ▼	Please select a Biskit Def

Returns the *Biskit* cast to the new type allowing access to the new types *properties*.

Properties Available for Child Actions To Use

The following properties are available for use in any child or descendant action:

Properties	Notes
biskit	The newly typed <i>Biskit</i> .

6.13.3.20 Veto Workflow Action

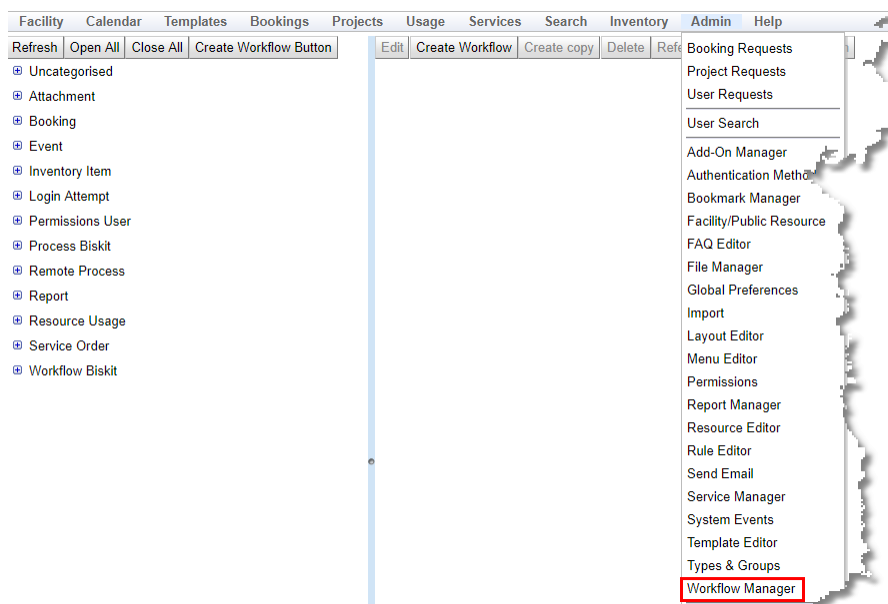
Takes the message to be displayed as input, can either be one created by the user or pulled from another source. If this [Action](#)⁶⁵⁶ runs it will stop the current [Workflow](#)⁶⁵⁶, vetoing any work already done in the *Workflow* and popping up the message. When vetoing a **Database Event** any changes to a [Biskit](#)⁶⁵² will be rolled back including stopping the *Biskit* from being created or updated. Therefore the *Workflow* can prevent a [Booking](#)⁶⁵² from being created or updated.

Properties Available for Child Actions To Use

There are no additional properties available for use in any child or descendant action beyond the standard set.

6.13.4 The Workflow Editor

The **Workflow Editor** shows all the [Workflows](#)⁶⁵⁶ and allows creating, updating and deleting them. By default, it appears on the menu here:



The **Create Workflow Button** creates a button that will run a *Workflow*. **Create Workflow** creates a new *Workflow*. See the [How Workflows Work](#)³³⁴ section for more details.

The Tree Of Workflows

Workflows are triggered when the [Workflow Event](#)³⁴⁰ is activated.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

- ⊕ Uncategorised
- ⊕ Attachment
- ⊕ Booking
- ⊕ Event
 - Workflow 10: Exception to email
- ⊕ Inventory Item
- ⊕ Login Attempt
- ⊕ Permissions User
- ⊕ Process Biskit
- ⊕ Remote Process
- ⊕ Report
- ⊕ Resource Usage
- ⊕ Service Order
 - ⊕ Histopathology Service Order
 - Workflow 2990: Create Label
 - Button 2992: Print Label
- ⊕ Workflow Biskit

Once all the items in the tree are viewable, the *Workflows* in their **Main Type** categorisation will be seen, with any that are disabled shown in red.

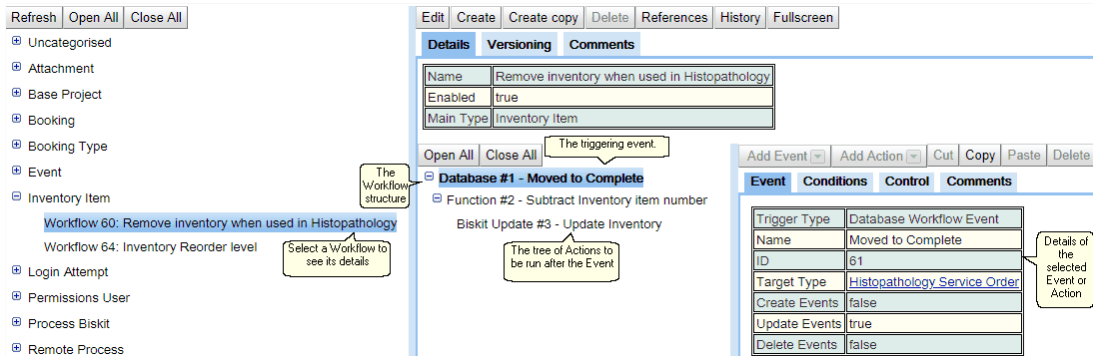
Anything without a **Main Type** defined will be stored under **Uncategorised**.

The definitions of *Workflows* and also those of the **Buttons** that run *Workflows* will both be found here.

This doesn't mean that a *Workflow* will be triggered, because the *Workflow* may have [conditions](#)⁶⁵³ that need to be checked.

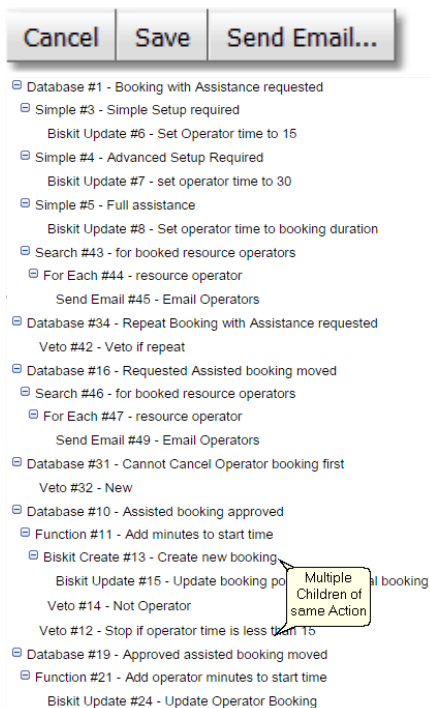
Workflow Details

Click on an *Workflow* in the tree, and see its details appear on the right:



For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Press the **Edit** button to make the page editable, and the button bar changes:



Once in **Edit** mode use the **Add Event** button to add a triggering [Event](#)⁶⁵⁶ to the *Workflow*. A *Workflow* can have many triggering *Events*.

For each *Event* add in the [Actions](#)⁶⁵⁶ that are going to be run. When adding an *Action* select the *Event* or *Action* that is to be its parent and press the **Add Action** button.

An *Event* or an *Action* may have many children, the **Sort Order** found in the *Actions* header information will decide which order the child *Action* will run in. The higher the **Sort Order** the later the child is run.

Sections of *Workflows* may be copied and pasted within a *Workflow* or to other *Workflows*. As *Actions* may refer to parent *Actions/Events* for information, some of this information may be lost when copied and a popup will show this information and so needs to be checked thoroughly.

Use the **Renumber** button to renumber the *Events* and *Actions* after moving/deleting has left the numbers no longer consecutive.

6.13.5 Workflow Use Cases

Here are some example reasons to want to use **Workflows**:

- Email the Lab Manager or PI when the end date of a grant is nearing so they have the opportunity to enter a new one before the monthly report is emailed to finance.
- Every week, automatically expire the accounts of any user that hasn't logged in for 6 months. It could also be configured to email such users a week before to let them know they have a week to log in before their account will be expired.
- Keep track of the number of [bookings](#)⁶⁵² and amount of time booked for each [project](#)⁶⁵⁴, storing those numbers on the project so they're always available for reporting and [booking rules](#)⁶⁵² to use.
- An alternative version of the previous item. The facility has a notion of core time and non-core time. For each *booking*, it is required to calculate the amount of time that is within core time, and the amount that is within non-core time, and update the *booking's project*, so that it keeps a running tally of its total use of core and non-core time.
- Calculate [properties](#)⁶⁵⁵ that you store on a booking. For example, a price or something that will be used as part of calculating a price.
- Somebody updates a *project*, and it indicates that it is funded by a grant, but some of the grant information is missing. If the [project status](#)⁶⁵⁴ is also set to **Approved**, which means it could be used for *bookings*, then either reject the change to the *project* (with a custom error message) or modify the *project's status* to reduce it to **Requested**.
- Provide grant code on a *project*, but allow a user to override that by selecting a different grant code when creating a *booking*.
- Consistency checks when entering *project* information with good, customised error messages when some requirement is broken
 - Example: if a *project* indicates it's using an MR scanner, then the user must provide information about ethics and safety information regarding what's going to taken into the scanner room.
- The facility has a significant number of users using it for a short period, they go away for a while, and come back again. It is required to know who is active, and to limit the number of active users to stay within the licence requirements. So a **Workflow** is configured to automatically expire users when they've been inactive for a while, but another **Workflow** can automatically un-expire users when they try to log in after a long period (perhaps provided they meet some other criteria)
- For a particular bookable [resource](#)⁶⁵⁵, it is required that at least one week before the *booking* date, a health and safety file is attached to the *booking*. A **Workflow** is created that will email the person who made the *booking* one week before it is due if the attachment is not there yet. The email gives them a reminder. Then two days before the *booking* is due, if the attachment is still not there, the *booking* is automatically cancelled and everybody who has used the *resource* over the past three months is sent an email telling them the slot has just opened up.
- Create a check-in and check-out system that enables an instrument for a duration, while recording the fact that this person is using it. This could be done by having a card reader that scans a university card, or by having a computer or tablet next to the instrument displaying a suitably configured page.

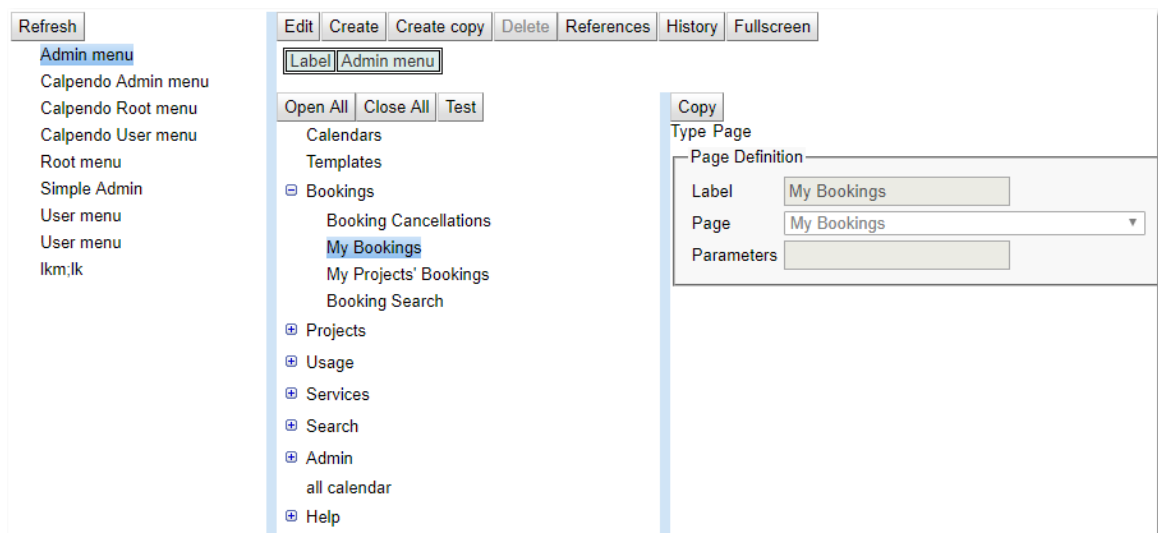
6.14 Menu Editor

Using the **Menu Editor** page the administrator can create, modify and delete the menus that exist in **Calpendo**. Once menus are created, a menu can be assigned to a user either by relying on the [global preferences](#)⁶⁵³ for new users (see the [Menus](#)⁵²⁷ tab of [Global Preferences](#))⁵²⁷ or by manually selecting the menu for particular users (see [Changing A User's Settings](#)¹⁸¹), allowing menus to be set up by the [users role](#)⁶⁵⁶ or on a user by user basis. There are many different types of pages that can be added to a users menu. They include but are not limited to:

1. Customised search or [data explorer](#)⁶⁵³ pages
2. Customised [calendar](#)⁶⁵² pages using [bookmarks](#)⁶⁵³
3. Pages that open up web pages external to **Calpendo**
4. Running predefined reports
5. Pages normally found only on the **Admin** menu
6. Customised [Biskit](#)⁶⁵² editor pages
7. Trigger a User Workflow Event

The **Menu Editor** page consists of three parts:

1. The left pane: This shows the available menus, and which menu is currently being edited. It has a single available Button on its menu toolbar to Refresh the contents of the pane.
2. The middle pane: this shows the content of the currently selected menu.
3. The right pane: this shows the detail of the currently selected menu item

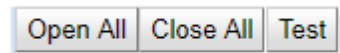


The Menu Toolbar (Middle Pane)

The tool bar at the top of the page of the middle pane contains buttons that operate on whole menus at a time.



There is also a toolbar for dealing with the individual menu.

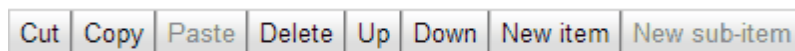


For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Tool Bar Item	Description
Test	Loads the currently selected menu into the menu bar to test.

The Menu Item Toolbar (Right pane in edit mode)

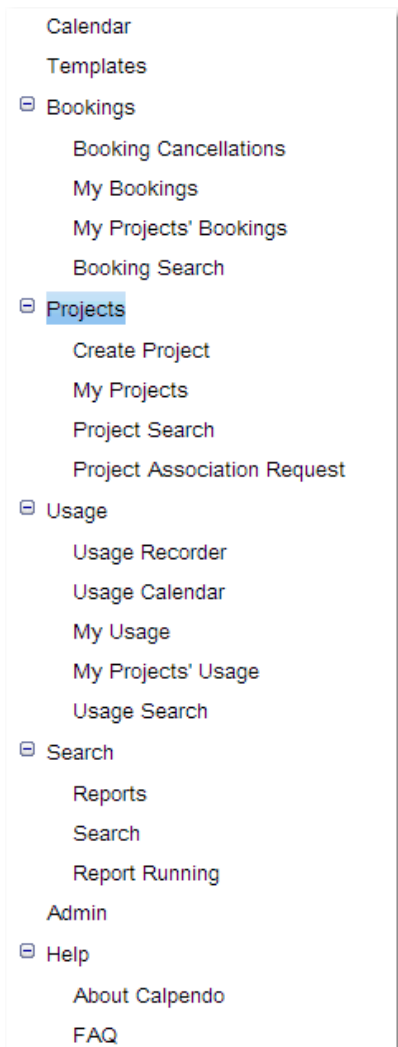
The tool bar in the right pane contains buttons that operate on single menu items.



For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Tool Bar Item	Description
Up	Moves the currently selected menu item up
Down	Moves the currently selected menu item down
New item	Creates a new menu item at the same level as the currently selected menu item.
New sub-item	Creates a new menu item as a child of the currently selected menu item.

The Middle Pane



This shows a tree with the contents of the currently selected menu. If an item is selected, then:

- that menu item is loaded into the right pane so you can modify the item
- use the Menu Item Toolbar to operate on the selected menu item

The Right Pane

This shows the details of the currently selected menu item. The first [property](#)⁶⁵⁵ shown is the menu item's type. There are five types, as described in this table:

The screenshot shows a user interface for defining a menu item. At the top, there is a 'Type' dropdown menu currently set to 'Page'. Below this is a section titled 'Page Definition'. Inside this section, there are three input fields: a 'Label' field containing the text 'Create Project', a 'Page' dropdown menu also showing 'Create Project', and a 'Parameters' text box which is currently empty.

Type	Description	
Page	This type of menu item is one that shows a page. When menu item is set to Page , the following properties need to be set:	
	Property	Description
	Label	This is the text that should be displayed on the menu.
	Page	This is a drop-down that allowing selection from all the different types of pages that Calpendo supports.
	Parameters	Some pages support additional parameters. If a page is selected that supports additional parameters, then the parameters value will be filled with a template that shows the sort of content expected.
Menu	This type of menu item is one that contains only sub-menu items. When this type is selected, also select the label that should be displayed for the sub-menu.	
Separator	Creates a horizontal line in the menu that separates the items before and after it. Once selected, also fill in the label. The label is not shown on the menu itself, but is used to display the item in the menu editor's left pane.	
Window	This represents a menu item that, when selected, will open another browser window. Set the following properties:	
	Property	Description
	Label	This is the text that should be displayed on the menu.
	URL	This is the address of the web page that should be loaded into the newly created browser window.

Type	Description	
Custom Page	This type of menu item is one that shows a custom built page. After choosing Custom Page , then set the following properties:	
	Property	Description
	Label	This is the text that should be displayed on the menu.
	Token	What the configurer wishes to appear in the browsers top address bar when this menu item is selected for example in the case of the Menu Editor: #menuEditor
	Custom Page Type	The type of custom page to create, 1. Biskit ⁶⁵² Tree Editor 2. <i>Biskit</i> Tree Viewer 3. <i>Bookmark</i> Page 4. Calendar By Location 5. Create <i>Biskit</i> Page 6. Customised Search Page 7. <i>Data Explorer</i> Page 8. Frame Page 9. Run Predefined Report 10. Run User Workflow Event
	Additional Information	This depends on the custom page type chosen

More information on the different types of **Custom Pages**.

Page	Description	Additional Information
<i>Biskit</i> Tree Editor	A page which which will display and edit a user-defined selection of Biskit Types ⁶⁵² . For example, if new <i>Biskit Types</i> have been created such as Doctor and Nurse and both a Doctor and a Nurse are associated with bookings ⁶⁵² , then create a menu item that takes the user to a page where they can create the Doctors and Nurses .	A list of the <i>Biskit Types</i> to be displayed on the page. Once created the order can be adjusted.
<i>Biskit</i> Tree Viewer Page	Provides a way to display a number of HTML pages with a navigation panel for users to choose which page to view.	1) The <i>Biskit Type</i> to be displayed 2) The <i>property</i> with the information to be viewed.

Page	Description	Additional Information
<i>Bookmark Page</i>	A <i>Calendar</i> page with the specified <i>bookmark</i> set up	The <i>bookmark</i> to be used.
<i>Calendar By Location</i>	Creates a cascading menu based on the defined locations, showing all resources at each location.	The Base Location, from which to cascade. The Label of the option that will take the user back to the calendar without changing the <i>resources</i> . Null means its not required.
Create <i>Biskit</i> Page	A page that creates a particular type of <i>Biskit</i> .	The <i>Biskit Type</i> to create <i>Biskits</i> of.
Customised Search Page	A search page customised to your specific requirements	1) The <i>Biskit Type</i> to be searched for 2) The report type to be used 3) A descriptor to define additional search buttons available on the search page. See below for a definition of the descriptors allowed. 4) Any Conditions to be used in the search
Data Explorer Page	A customised <i>Data Explorer</i> page	1) The <i>Biskit Type</i> to be displayed 2) The ID of the particular record to be displayed. 0 or empty means display all 3) Whether a user viewing the custom page may change the <i>Biskit Type</i> being displayed.
Frame Page	Opens any URL as a frame inside Calpendo . This will display with the menu at the top of the page and then the content of the URL specified below.	The URL to be displayed. For example: url=http://www.exprodo.com
Run Predefined Report	Runs a report either system or personal.	The report to be run
Run User Workflow Event	Allows then user to run a Workflow from the menu.	1) <i>waitForEvent</i> : whether the system will wait for the workflow event to finish or continue straight away, defaults to True. 2) <i>holdingMessage</i> : message to show whilst waiting for the event.

The following page type may be useful when customising menus:

Page	Description	Parameters Example
Layout Editor ⁶⁵³	Opens up an editor which allows the user to define how to layout <i>properties</i> in a <i>Biskit</i> so that they can be read and managed easily. This is useful if your type has a large number of properties to be displayed.	
Find by URL	Opens up a page based on a URL	searchtype=

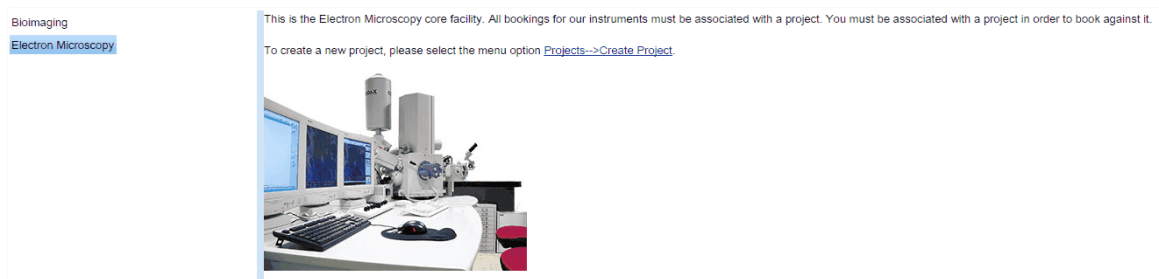
Biskit Tree Viewer

Create a custom *biskit* and give it a **Name** *property* and a **Content** *property*. The **Content** *property* should be of type **String**, and of subtype **HTML**.

Create a number of instances of the new Biskit Type, giving each a name and adding content in rich text associated with the name. This can include links and images.

Use the **Menu Editor** to add a custom page of type **Biskit Tree Viewer**, and specify the type of *biskit* and which *property* contains the **HTML** content to display.

Users then see the names of the *biskits* appear in a navigation pane on the left with the associated content on the right.



Descriptors for Custom Search Pages

The definition of the format of a descriptor:

- line := {group}{;{group}}*
- {group} := [{initialFlags}[:{css-class}[:label]]]{item}{, {item}}*
- {initialFlags} := (H|V|G|F|E|C)[S|L][A][T|O]
- {item} := {propPath}[:{itemFlags}][*][!]
- {itemFlags} := (S|L)
- {propPath} = a path, this may be the name of a property e.g. "status", or a path to the property e.g. "project.status", or [Free] in which case all String properties are searched for the value.

Where:

- css-class is the CSS class to be applied to the top level
- the initial flags are:
 - flags to indicate the overall behaviour of the search:
 - H = horizontal button bar
 - V = vertical panel
 - F = Form-like container that displays each item with a heading in the line above the search item.
 - G = Growing form - displays like a form but can be hidden or shown by ticking a checkbox.
 - E = Expanding, a container showing a checkbox for each item, where clicking the checkbox will open up the search item
 - C = Captions around each of the items in the container
 - F = Form, labels above each widget
 - flags to indicate the default size of items. Note that these can be overridden by size flags placed on each item:
 - S = Small - widgets in this container should be small size by default
 - L = Large - widgets in this container should be full size by default
 - Sorting flags:
 - No flag means items are presented in the order specified and (when using the A flag, their natural order)
 - T = sort all items in this container by their text label
 - O = sort all items in this container by their bakery-assigned order
 - Miscellaneous flags:
 - A = automatically add items for all known properties not added elsewhere

- item flags define the size:
 - S = small - minimal widgets in a row
 - L = large - widgets taking full size the whole time

Examples:

- `status:*;service.provider:*;service.location:*;{EAT}`
- `{CL:myCSSclass:My Label}status:*,service.provider:*;{F}service.location:*`

Creating A New Menu

Press **Create** on the menu toolbar, and type the name of the new menu into **Label** box.

 A screenshot showing a 'Label' button next to a text input field containing the text 'new Menu'.

Using the **Menu Item Toolbar**, press **New Item**, then in the right pane set up the **Type** to be **Page**, and for the **Page Definition** set **Label** to **Calendar** and **Page** to **Calendar**. This will create the first menu item.

 A screenshot of the menu configuration interface. On the left, there are buttons for 'Cancel', 'Save', 'Open All', 'Close All', and 'Test'. Below these is a 'Label' input field with 'new Menu' and a list containing 'Calendar'. On the right, the 'Page Definition' pane is active, showing 'Type' as 'Page', 'Label' as 'Calendar', 'Page' as 'Calendar', and an empty 'Parameters' field.

Using the **Menu Item Toolbar**, press on **New Item**, then in the right pane set up the **Type** to be **Menu**, and for the **Submenu Definition** set **Label** to **Simple User Menu**. This creates a simple sub menu.

 A screenshot of the menu configuration interface. On the left, the 'Label' input field still contains 'new Menu', but the list now contains both 'Calendar' and 'Simple User Menu'. On the right, the 'Submenu Definition' pane is active, showing 'Type' as 'Menu' and 'Label' as 'Simple User Menu'.

Using the **Menu Item Toolbar**, press **New Sub Item**, then in the right pane set up the **Type** to be **Page**, and for the **Page Definition** set **Label** to **My Bookings** and **Page** to **My Bookings**. This will put an item on the **Simple User Menu** sub menu.

The screenshot shows the Calpendo menu configuration interface. On the left, a tree view shows the menu structure: 'Calendar' and 'Simple User Menu' (expanded) with 'My Bookings' selected. The top bar has 'Cancel' and 'Save' buttons. Below the tree are 'Open All', 'Close All', and 'Test' buttons. The right pane shows the 'Page Definition' for the selected item. The 'Type' is set to 'Page'. The 'Page Definition' section has 'Label' set to 'My Bookings' and 'Page' set to 'My Bookings'. The 'Parameters' field is empty. The 'Menu Item Toolbar' at the top right includes buttons for Cut, Copy, Paste, Delete, Up, Down, New item, and New sub-item.

Click on **Simple User Menu** then using the **Menu Item Toolbar**, press **New Item**, then in the right pane set up the **Type** to be **Separator**. This creates a simple separator to which is a line across the menu to split the menu into sections.

The screenshot shows the Calpendo menu configuration interface. On the left, the tree view shows 'Calendar', 'Simple User Menu' (expanded), and 'My Bookings' with 'Separator' selected. The top bar has 'Cancel' and 'Save' buttons. Below the tree are 'Open All', 'Close All', and 'Test' buttons. The right pane shows the 'Page Definition' for the selected item. The 'Type' is set to 'Separator'. The 'Page Definition' section is empty. The 'Menu Item Toolbar' at the top right includes buttons for Cut, Copy, Paste, Delete, Up, Down, New item, and New sub-item.

Continue adding in pages, custom pages, sub menus and separators until the menu is created. Once finished use the **Save** button and then use the **Test** button to install the menu temporarily into the browser. Check the menu to make sure it is what is required. Remember to use the **Up**, **Down** buttons to change the order of items in the menu. Look at the chapter on [Changing a Users Settings](#)¹⁸¹ to see how to apply this menu to a particular user. Remember putting this menu as an option in Global Preferences will only change that menu for new users not existing users.

6.15 User Authentication Methods

This section outlines how an Administrator can set up a number of different methods for user authentication, allowing **Calpendo** to be used in a single sign on environment.

6.15.1 Authentication Methods

There are three main types of authentication:

- 1) **Local**: The current **Calpendo** does the authentication. User names and encrypted passwords are stored in the **Calpendo** database.
- 2) **Internal**: These are built-in methods that allow **Calpendo** to send your user name and password to another system for authentication. **Calpendo** does not store your password in this case, but does handle it during login. The currently supported internal authentication methods include using an email server (**SMTP** or **IMAP**), any HTTP basic authentication system, or another **Calpendo**.
- 3) **External**: The web server that users connect to is configured to provide authentication, for example using a single-sign-on system or perhaps LDAP. In this case, **Calpendo** never sees users passwords, and authentication is solely the responsibility of the web server.

All systems automatically have **Local** authentication initialised but the administrator may also set up a number of non-local authentications and may switch off **Local** authentication. The administrator cannot switch off the authentication system they are currently logged in with, to ensure that they cannot be accidentally locked out.

Common Options For All Authentication Methods

Name	Data Type	Description
Type	String	The type of authentication method.
Name	String	The name assigned to this method. Must be unique.
Login Allowed	Boolean	Whether users can login using this method.
Hide Login Button	Boolean	Whether the login button is visible or not.
New User Registration Allowed	Boolean	Whether new users can register using this method. If True then new users may register for this authentication method even if logins are not allowed.
Allow Custom Nick Names	Boolean	Whether login nick names are allowed or are nick names identical to the login identifier.

Local Authentication

Local authentication has no additional options.

IMAP Authentication and SMTP Authentication

Name	Data Type	Description
Host	String	The mail server's IP address or name.
Security	JavaEnum	Choose the type of security to be used: None, STARTTLS, SSL/TLS

Basic Authentication

Name	Data Type	Description
URL	String	The address of the website doing the authentication.
Realm	String	The name of the system, that is doing the authentication.

Exprodo Authentication

Name	Data Type	Description
URL	String	The address of the website doing the authentication.

LDAP Authentication

LDAP Options	Data Type	Description
Host URLs	String	The address of the website doing the authentication.
Port	String	The port to connect on.
Security	JavaEnum	Choose the type of security to be used: None, STARTTLS, SSL/TLS
Ldap Trust Server	JavaEnum	Only visible if Security is not None . Choose whether to Blindly Trust Server or Examine Security certificates as normal .
Method	JavaEnum	Bind As User or Bind As Admin
User DN	String	User name for access within LDAP

LDAP Advanced	Data Type	Description
Base DN	String	The starting point of the LDAP branch.
Filter	String	

Bind as User: The user provides their login name and password, and there is an attempt to connect (bind) to the **LDAP** server as that user and password. If a connection occurs, then the user is authenticated.

Bind As Admin: This is where a known **DN** and password is configured for a particular user (the admin), and then once connected to the **LDAP** server as that user, a search is performed for the user being authenticated. Once the user is found, the password the user provides is used along with the user **DN** to authenticate the user.

When choosing between these two methods, they each have advantages and disadvantages:

- **bind-As-user** cannot handle multiple formats of user **DN**, but it does not require an admin's user **DN** and password.
- **bind-As-admin** has the disadvantage of having to store an admin's user **DN** and password, but can handle multiple formats of user **DN**.

External Authentication

Name	Data Type	Description
Login Name Header	String	The name of the HTTP header that will contain the user's login name (this defaults to X-Forwarded-User)
Display Type	JavaEnum	How to display the frame that will contain the authentication handshake. If not hidden then choose its size. Choice of: Hidden, Inline, Popup, Redirect Hidden, Inline and Popup all work by doing the authentication handshake in an iframe. Redirect works without an iframe, and so takes over the user's whole browser tab while the authentication is done.
Cookies to remove on logout	JavaEnum	Defines which cookies will be removed at logout. Choice of : Remove no cookies, Remove all cookies, Remove some cookies. Remove some cookies allows input to define which cookies are to be removed.
Display Width	String	The width of the frame within the pop up or on the login page.
Display Height	String	The height of the frame within the pop up or on the login page.
Attribute Map	Set	Allows mapping from headers to user property. if multiple are mapped to the same property the first value to populate is used.

External Proxy Authentication

Name	Data Type	Description
Login Name Header	String	The name of the HTTP header that will contain the user's login name (this defaults to X-Forwarded-User)
Display Type	JavaEnum	How to display the frame that will contain the authentication handshake. If not hidden then choose its size. Choice of: Hidden, Inline, Popup, Redirect Hidden, Inline and Popup all work by doing the authentication handshake in an iframe. Redirect works without an iframe, and so takes over the user's whole browser tab while the authentication is done.
Cookies to remove on logout	JavaEnum	Defines which cookies will be removed at logout. Choice of : Remove no cookies, Remove all cookies, Remove some cookies.

		Remove some cookies allows input to define which cookies are to be removed.
Display Width	String	The width of the frame within the pop up or on the login page.
Display Height	String	The height of the frame within the pop up or on the login page.
Attribute Map	Set	Allows mapping from headers to user property. if multiple are mapped to the same property the first value to populate is used.

Name	Data Type	Description
Proxy URL	String	The URL which will hold the proxy server information.
Proxy Authentication Name	String	The external authentication name method to access the proxy server.
Identity Provider	Mapped Int	The Entity ID of the identity provider to use. If not set users will be asked to select their institution.

While you can use **HTTP** basic authentication using an internal authentication method, it's also possible to set up **HTTP** basic authentication using external authentication. The following shows an example of an excerpt from an **Apache** virtual host configuration that sets up **HTTP** basic authentication and also passes the **REMOTE_USER** setting that it generates to **Calpendo** by setting the **X-Forwarded-User** **HTTP** header. You can use any header for this, but you need to tell **Calpendo** which header to examine by setting the **Login Name Header** property. When Apache rewrites **URLs**, you protect **/private** under the rewritten **URL**

```
<Location "/Calpendo/com.springsolutions.calpendo.Calpendo/private">
  AuthType basic
  AuthName "Your Calpendo Realm Name"
  AuthBasicProvider file
  AuthUserFile /path/to/htpasswd/file
  Require valid-user
</Location>

RewriteEngine On

# prevent the client from setting this header
RequestHeader unset X-Forwarded-User

# see the Apache documentation on why this has to be lookahead
RewriteCond %{LA-U:REMOTE_USER} (.+)

# this actually doesn't rewrite anything. what we do here is to set RU to the match above
# "NS" prevents flooding the error log
RewriteRule .* - [E=RU:%1,NS]
RequestHeader set X-Forwarded-user %{RU}e

# strip the REALM of Kerberos Login
# RequestHeader edit X-Forwarded-User "@REALM$" ""

RewriteRule ^/Calpendo/Service(.*) /Calpendo/com.springsolutions.calpendo.Calpendo/Service$1 [PT,L]
RewriteRule ^/Calpendo/Calpendo.html(.*) /Calpendo/com.springsolutions.calpendo.Calpendo/Calpendo.html$1 [PT,L]
RewriteRule ^/Calpendo/$ /Calpendo/com.springsolutions.calpendo.Calpendo/Calpendo.html [PT,L]
RewriteRule ^/Calpendo/(.*) /Calpendo/com.springsolutions.calpendo.Calpendo/$1 [PT,L]
ProxyPass /Calpendo http://localhost:8080/Calpendo
ProxyPassReverse /Calpendo http://localhost:8080/Calpendo
```

When Apache rewrites URLs, you protect /private under the rewritten URL

Attribute Map

When installing **Calpendo** onto a local server, the **Apache** configuration is required to capture **Apache** environment variables and make them available as **HTTP** headers.

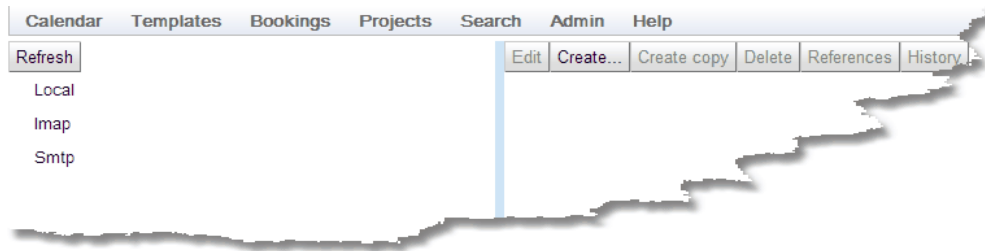
The shibboleth **External Proxy** authentication run by the **Exprodo Software** service at <https://sp.exprodo.com/> exposes many standard attributes as **HTTP** headers, although each identity provider may choose to expose each of those attributes to **Calpendo** or not.

The external authentication in **Calpendo** can now be set up to provide a mapping from an **HTTP** header to any string property of a user. Multiple headers can be mapped to the same user property, in which case the first one which is actually populated with a value is the one that will be used.

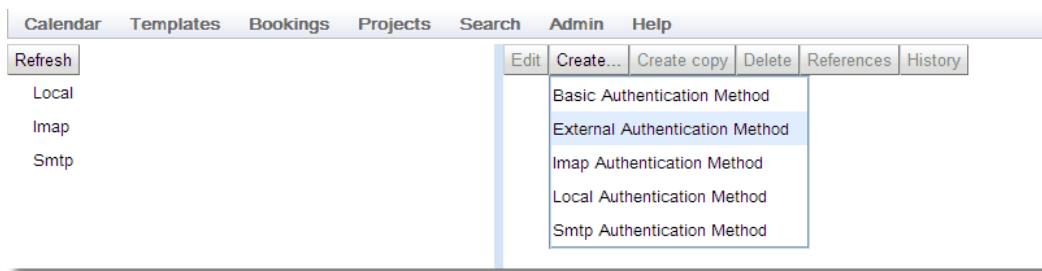
6.15.2 Authentication Methods Editor

Setting up different Authentication Methods (alternatively known as Single Sign On) is done using the **Authentication Methods Editor**, which by default is available on the menu as **Admin->Authentication Methods**.

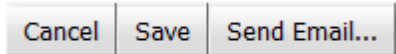
The **Authentication Methods Editor** shows all the currently defined authentication methods and allows the administrator to create, update and delete them. This is what the looks like when first opened.



For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter. The only difference from the standard is that when the **Create** button is clicked a drop down menu appears to allow the administrator to choose which type of Authentication Method to be created.



After pressing **Create** or the **Edit** button to make the page editable the button bar changes:



The detail view for the authentication method will also appear, fill in the details and **Save** the method.

Common Options	
Type	Imap Authentication Method
Name	<input type="text"/>
Login Allowed	Allowed ▼
New User Registration Allowed	Allowed ▼

IMAP Options	
Host	<input type="text"/>
Security	None ▼

6.16 FAQ Editor

Calpendo provides two ways to provide help for the users specific to your facility. Firstly, it is possible to create your own documentation and add links to them on the menu. See [Menu Editor](#)⁴⁸⁹ for guidance on how to configure the menus. The alternative is to configure the [Frequently Asked Questions](#)¹⁴⁵ page.

Do this with the **FAQ Editor**, which is by default on the menu as **Admin-->FAQ Editor**. This allows the creation of questions with answers, and the ability to assign them to categories. The categories provide a way to group the questions and answers. This is a way to make sure the users know who to contact in case of particular problems.

FAQ have the following properties:

Property	Description
Category	The category to place the FAQ ⁶⁵³ under
Question	The question part to the <i>FAQ</i>
Sort Order	The order this question appears in its category
Answer	The answer to the <i>FAQ</i> . See below for more information on how answers may be created.

Category has the following properties:

Property	Description
Name	The name of the category.
Sort Order	The order this category will appear in the <i>FAQ</i> page.

For more information on how the **FAQ Editor** works read the chapter on [Configuring Types and Groups](#)²⁸⁰ as the **FAQ Editor** works in a similar way.

The answers provided can contain HTML. As an example of the sort of HTML used to control the way your FAQs look, here's a sample HTML answer:

Empty lines make no difference - it all appears in the same paragraph, and line breaks can be anywhere without changing how it actually looks. But if you insert an HTML line break using the right
tag, then you get a new line. You can also use a new paragraph tag like this: <p>This will start a new line and leave space as well.<p>

```
<ul>
  <li> first item
  <li> second item
  <li> third item.
</ul>
```

```
<ol>
  <li> first item
  <li> second item
  <li> third item.
</ol>
```

This is some indented text. If you make it very long, you should see that the line will wrap around the way you'd expect it to, with each line continuing to be indented. You will need to make sure your web browser is narrow enough to show how this wraps onto other lines.

You can provide a ``link to other web sites if you want to. You could even use some other HTML editor and copy the HTML in here if you want.

If you want to include less than (<) or greater than (>) signs or the ampersand (&), you need to use special codes.

and this is the way that it looks:

Calendar
Templates
Bookings
Projects
Search
Admin
Help
FAQ | admin | [Reset password](#) | [Settings](#) | [Sign out](#)

General

What needs an HTML formatted answer?

This is the first paragraph. Empty lines make no difference – it all appears in the same paragraph, and line breaks can be anywhere without changing how it actually looks. But if you insert an HTML line break using the right tag, then you get a new line. You can also use a new paragraph tag like this:

This will start a new line and leave space as well.

For bullet points, you need an "unordered list" (ul), with "list items" (li), like this:

- first item
- second item
- third item.

For numbered sections, use an "ordered list" (ol) with list items:

1. first item
2. second item
3. third item.

If you want a section that's indented a bit, then there are various ways to do that in HTML. One option is to create a div with styling that gives it a margin:

This is some indented text. If you make it very long, you should see that the line will wrap around the way you'd expect it to, with each line continuing to be indented. You will need to make sure your web browser is narrow enough to show how this wraps onto other lines.

You can put some **text in bold** and some other *text can be emphasized*, which usually means italics. You can choose colours as well. You can provide a [link to other web sites](#) if you want to. You could even use some other HTML editor and copy the HTML in here if you want.

If you want a horizontal line, then think "horizontal rule" (hr) like this:

If you want to include less than (<) or greater than (>) signs or the ampersand (&), you need to use special codes.

Where does the name 'Calpendo' come from?

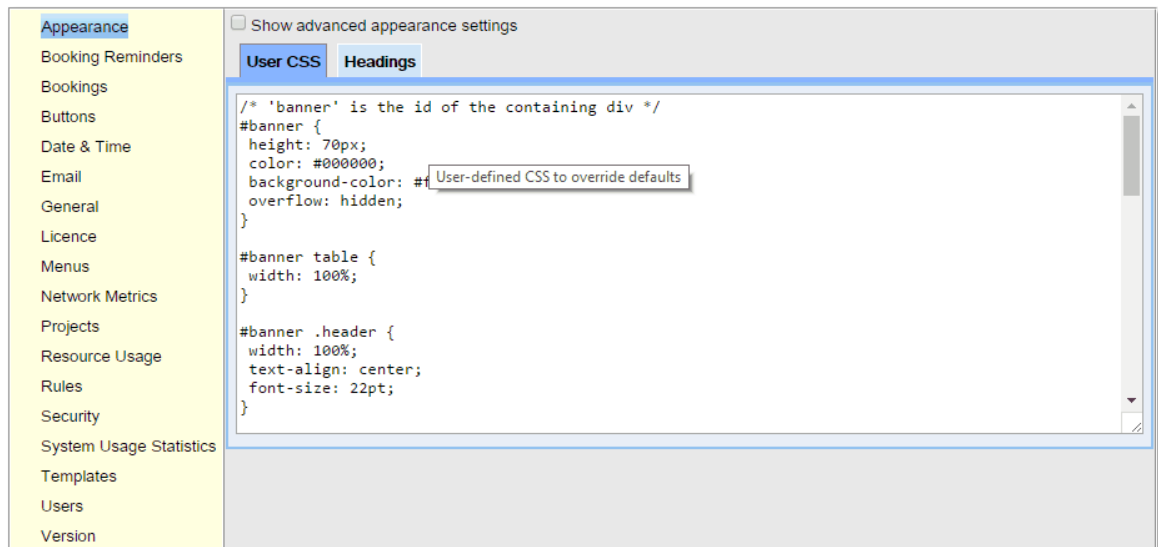
It is a mixture of 'calendar' and the latin 'pendo' which means to consider, to weigh, to judge.

Why can't I see my/any bookings?

Check that the resource you want to see is highlighted under 'resources' to the left of the calendar

6.17 Global Preferences

The **Global Preferences** page appears by default on the menu as **Admin-->Global Preferences** and specifies settings that change the way **Calpendo** operates.



Once all necessary changes have been made throughout the tabs use the **Save** button to implement them.

6.17.1 Appearance

Appearance controls the look and feel of **Calpendo**. It does this by allowing the creation of CSS to overwrite the default values. The user can also set the headings for the **Host Page** and for the **Login page**, as well as a banner to be displayed at the top of each page. The **Headings (debug)** setting will only be shown when the **Show advanced appearance** box is ticked.

Setting	Description
Skins	Allows the administrator to change the look and feel of the system for all users.
User CSS	To specify Cascading Style Sheets (CSS) that you would like to be used by Calpendo . Anything specified here will override the default values, so that you can customise the look and feel.
Headings	This sets several properties that control the look of Calpendo , see the table below for the headers that can be changed.
Headings (debug)	To specify the same sort of content as used for Headings above, but allows alternative values when Calpendo is in debug mode.

Skins

Administrators can choose the skins they want to use and can also define their own skin/theme. To create a new theme go to **Search->Search** and **Search for: Theme**.

Cancel Save Apply Export Theme Export CSS Test Theme

Name theme
Enabled YES

Values

calendar.allDayEvents.backgroundColor
calendar.allDayEventsAdd.backgroundColor
calendar.backgroundColor
calendar.borders.color
calendar.filters.backgroundColor
calendar.halfHourBorders.color
calendar.headers.color
calendar.hourLabel.color
calendar.timeOfDayMarker.color
calendar.unselectedTab.backgroundColor
calendar.unselectedTab.color
menu.bar.backgroundColor
menu.bar.border.color
menu.bar.color
menu.bar.selected.backgroundColor
menu.bar.selected.color
menu.item.backgroundColor

Palette

20% black
50% grey
7% black
73% grey
87% grey
91% grey
Calendar blue
Dark blue
Dark violetish
Greeny grey
Light blue
Light violetish
Medium blue
Medium pale blue
Pale blue
Pale green
Pale yellow

Give the Theme a name, set the colours required for each of the elements and adjust the palette if required. Click **Save** and the theme will be available for selection in both the Global and User settings areas. Adjusting the palette is a quick way to change the colour for all options that use it.

Export Themes: Will provide a download of the JSON that defines the theme.

Export CSS: Will provide a download of the CSS that defines the theme.

Test Theme: Works without saving and allows users to see changes straight away.

Value	Effects
calendar.allDayEvents.backgroundColor	Area which is behind an individual All Day event booking, which is in the colour of its resource.
calendar.allDayEventsAdd.backgroundColor	Area where the All Day bookings are placed between the date and the resource name.
calendar.backgroundColor	All free space behind the whole Calender
calendar.borders.color	Area between resources, under dates and times, and other border areas, including background of selected calendar type (week, day, etc)
calendar.filters.backgroundColor	Background area for the filters on the left of the calendar.
calendar.halfHourBorders.color	The colour of the lines showing the half hour marks
calendar.headers.color	Colour of the dates text and selected calender type (week,day etc) text.
calendar.hourLabel.color	Colour of the Hour text
calendar.timeOfDayMarker.color	The marker for the current time of day.
calendar.unselectedTab.backgroundColor	Background colour for all unselected calender types (week,day etc).
calendar.unselectedTab.color	Text colour for all unselected calender types (week,day etc).
menu.bar.backgroundColor	Background colour for the complete menu bar
menu.bar.border.color	Border color of the menu bar
menu.bar.color	Colour of text on the menu bar
menu.bar.selected.backgroundColor	Colour of the currently selected menu option
menu.item.selected.color	Colour of the text of the currently selected menu option
menu.item.backgroundColor	Colour of items in the pulldown menu.
menu.item.color	Colour of the text of items in the pulldown menu
menu.item.selected.backgroundColor	Colour of selected items in the pulldown menu
menu.item.selected.color	Colour of the text of selected items in the pulldown menu
splitter.color	Colour of border between sections of editors (resource, layout etc.)

Value	Effects
tab.bar.backgroundColor	Background colour of all unselected tabs used in Biskit Layout.
tab.bar.color	Text colour of all unselected tabs used in Biskit Layout.
tab.bar.hover.backgroundColor	Colour of tab background when hovering over it.
tab.bar.hover.color	Colour of tab text when hovering over it.
tab.bar.selected.backgroundColor	Background colour of selected tab.
tab.bar.selected.color	Colour of text of the selected tab.
tab.border.color	Colour of border around the tab.
tab.panel.backgroundColor	Colour of the background of the tabs panel.
tab.panel.color	Colour of text in the tabs panel.
table.headers.border.color	The colour of the borders around the table header.
table.rowEven.backgroundColor	Background colour of even rows.
table.rowEven.color	Text colour on even rows.
table.rowHeader.backgroundColor	Background colour for table header.
table.rowHeader.color	Text colour for table header.
table.rowMouseOver.backgroundColor	Background colour of row when mouse hovers over it.
table.rowMouseOver.color	Text colour of row when mouse hovers over it.
table.rowOdd.backgroundColor	Background colour of odd rows.
table.rowOdd.color	Text colour on odd rows.
table.rowSelected.backgroundColor	Background colour of selected row of table.
table.rowSelected.color	Text colour of selected row of table.
tree.backgroundColor	Background colour for all Tree areas such as Location, Resource.
tree.hover.backgroundColor	Background colour when hovering over a tree item.
tree.hover.color	Text colour when hovering over a tree item
tree.selected.backgroundColor	Background colour of a tree selected item.
tree.selected.color	Text colour of a tree selected item.
tree.selectedHover.backgroundColor	Background colour when hovering over a selected tree item.
tree.selectedHover.color	Text colour when hovering over a selected tree item.
trigger.highlight.child.backgroundColor	Background colour of the highlighted actions, child actions in a workflow.
trigger.highlight.child.color	Text colour of the highlighted actions, child actions in a workflow
trigger.highlight.grandchild.backgroundColor	Background colour of the highlighted actions, grandchild actions in a workflow.

Value	Effects
trigger.highlight.grandchild.color	Text colour of the highlighted actions, grandchild actions in a workflow.
trigger.highlight.parent.backgroundColor	Background colour of the highlighted actions, parent action in a workflow.
trigger.highlight.parent.color	Text colour of the highlighted actions, parent action in a workflow.
trigger.highlight.sibling.backgroundColor	Background colour of the highlighted actions, sibling actions in a workflow.
trigger.highlight.sibling.color	Text colour of the highlighted actions, sibling actions in a workflow.
trigger.highlight.siblingChild.backgroundColor	Background colour of the highlighted actions, sibling child actions in a workflow.
trigger.highlight.siblingChild.color	Text colour of the highlighted actions, sibling child actions in a workflow.
verticalTab.bar.backgroundColor	Colour of the vertical tab area.
verticalTab.bar.color	Colour of the text in a vertical tab.
verticalTab.bar.hover.backgroundColor	Background colour when hovering over a vertical tab item.
verticalTab.bar.hover.color	Text colour when hovering over a vertical tab item.
verticalTab.bar.selected.backgroundColor	Background colour of a vertical tab selected item.
verticalTab.bar.selected.color	Text colour of a vertical tab selected item.
verticalTab.bar.selectedHover.backgroundColor	Background colour when hovering over a selected vertical tab item.
verticalTab.bar.selectedHover.color	Text colour when hovering over a selected vertical tab item.
verticalTab.panel.backgroundColor	Colour of the vertical tab panel.
verticalTab.panel.color	Colour of text in the vertical tab panel.

User CSS

An example of User CSS, would be the following

```
#registerButton
{
display: none;
}
```

This would remove the *Register New User* button from the login page.

An example of User CSS for Calpendo would be to change the way **Cancelled** bookings looked:

This would draw a line through the text in the header (strike through).

```
.Booking-CANCELLED .Booking-heading {
  text-decoration: line-through
}
```

This would draw diagonal lines through the body of the booking, these lines can have multiple colours, here they are set to be the same colour (salmon).

```
.Booking-CANCELLED .Booking-body {
  background-image: repeating-linear-gradient(-45deg, #F9966B, #F9966B 1px, transparent
  1px, transparent 14px) !important;
  background-size: 20px 20px !important;
}
```

Headings

Heading options	Description
Host Page Title	This changes the HTML title of the page used for Calpendo so that it will show in the web browser.
Login Header	This lets the user provide alternative HTML for the header that appears on the login page.
Page Banner	If a page banner is required, this is the HTML that will be shown at the top of every page.

☒ Show advanced appearance settings

User CSS

Headings

Headings (debug)

Host Page Title	Calpendo
Login Header	<pre><table> <tr> <td></td> <td><div class='header'>\${title}</div><center><div class='version'>Version \${version}</div></center></td> </tr> </table></pre>
Page Banner	<input checked="" type="checkbox"/> Show Banner Calpendo

6.17.2 Booking Reminders

As described by [Creating Bookings](#)⁵⁶, when a [booking](#)⁶⁵² is made, the user can choose to enable *booking* reminders. For each user, their [User Settings](#)³⁴ specifies the default values for *booking* reminders. The [global preference](#)⁶⁵³ for *booking* reminders is similar, but allows an administrator to specify the *booking* reminder user settings for newly registered users. The global setting is used only for setting up new users and nothing else.

Choose who gets a reminder email, with the options being:

- The person that made the *booking*
- The person that owns the *booking*
- The person that owns the *booking's project*⁶⁵⁴. If there is no *project* on the *booking*, then this setting is ignored.
- The people associated with the *booking's project*. If there is no *project* on the *booking*, then this setting is ignored.

If one person falls into more than one category, for example the *project* owner is also the person that made the *booking*, they will only receive one reminder email.

6.17.3 Bookings

Default Booking Status

When a user creates a [booking](#)⁶⁵² with the status **Best possible**, and there are no [Time Templates](#)⁶⁵⁶, [Booking Rules](#)⁶⁵² or [Permissions](#)⁶⁵⁴, to alter that choice then the [Default Booking Status](#)⁶⁵³ is used. This will also define the list of possibilities a user will see in the [Booking Status](#)⁶⁵³ drop down box. If set to **Requested** only **Best Possible** and **Requested** will appear, if set to **Approved**, **Best Possible**, **Requested** and **Approved** will appear.

Setting	Description
Bookings Created By An Admin	Specify the default <i>status</i> for <i>bookings</i> created by a user with the Admin role ⁶⁵⁶ . This is Approved by default.
Bookings Create By Regular Users	Specify the default <i>status</i> for <i>bookings</i> created by users that do not have the Admin role. This is Requested by default.

Bookings Calendar Background Time Template Colours

When showing *Time Templates* on the background of the [bookings calendar](#)⁶⁵², different colours are used to indicate the implications of the *Time Templates* present. This is where the colours used can be setup. For each user, their [User Settings](#)³⁴ specifies the default colours for *Time Template* colours. The [global preference](#)⁶⁵³ for *Time Template* colours is similar, but allows an administrator to specify the *Time Template* colour user settings for newly registered users. The global setting is used only for setting up new users.

Bookings Calendar Background Template Colours	
Colour when bookings would be automatically denied	<input type="color" value="#f08080"/>
Colour when booking warnings would be given	<input type="color" value="#ffff00"/>
Colour when booking requests can be made	<input type="color" value="#ffff00"/>
Colour when bookings would be automatically approved	<input type="color" value="#90ee90"/>
Colour when templates are indeterminate	<input type="color" value="#d3d3d3"/>
Colour when no templates apply	<input type="color" value="#ffffff"/>
Acceptability to display when user has no suitable projects	Automatic denial ▼
Display of indeterminate templates	Show as indeterminate ▼

Reset colours

Setting	Description
Colour when <i>bookings</i> would be automatically denied	This is a light red by default
Colour when <i>booking</i> warnings would be given	This is a light yellow by default
Colour when <i>booking</i> requests can be made	This is a light yellow by default
Colour when <i>bookings</i> would be automatically approved	This is a light green by default
Colour when <i>templates</i> are indeterminate	This is grey by default
Colour when no <i>templates</i> apply	This is white by default
Acceptability to display when user has no suitable projects.	The levels to choose from are: Automatic denial Warning Acceptable Automatic approval Do not display
Display of indeterminate templates.	The choices are Show as indeterminate Show least restrictive template

There is also a **Reset Colours** button which resets the colours back to the original defaults.

Bookings Calendar Background Tooltip When Time Template Has No Message

When displaying *Time Templates* on the background of the *Bookings Calendar*, if the user lets their mouse hover over the background then a tool tip will be shown that gives a message. If the *Time Templates* that apply at that time contain a message, then that's what will be shown in the tool tip. However, when there is no message provided, then **Calpendo** needs to know what message to show. This depends on whether the *Time Templates* indicate that a *booking* would be automatically approved, or denied etc.

Bookings Calendar Background Tooltip When Template Has No Message

Tooltip text when bookings would be automatically denied	Show tooltip ▼	You cannot book here
Tooltip text when booking warnings would be given	Show tooltip ▼	You will be warned if you book here
Tooltip text when booking requests can be made	Show tooltip ▼	You may make booking requests here
Tooltip text when bookings would be automatically approved	Show tooltip ▼	Bookings here will be automatically approved
Tooltip text when templates are indeterminate	Show tooltip ▼	There are multiple conflicting possibilities for template inf
Tooltip text when no templates apply	Do not show tooltip ▼	There is no template information here

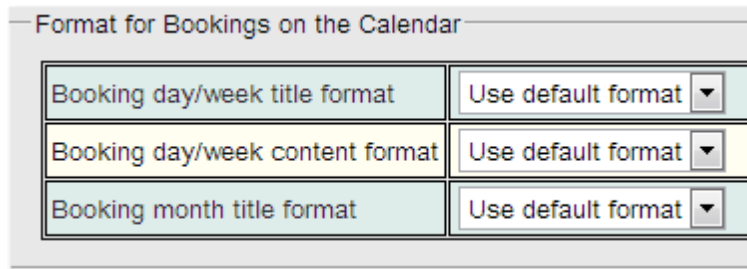
Reset default text

Setting	Description
Tooltip text when <i>bookings</i> would be automatically denied	Default: "You cannot book here"
Tooltip text when <i>booking</i> warnings would be given	Default: "You will be warned if you book here"
Tooltip text when <i>booking</i> requests can be made	Default: "You may make booking requests here"
Tooltip text when <i>bookings</i> would be automatically approved	Default: "Bookings here will be automatically approved"
Tooltip text when Templates are indeterminate	Default: "There are multiple conflicting possibilities for template information depending on your chosen project"
Tooltip text when no <i>templates</i> apply	Default: "There is no template information here"

There is a button to restore the text to the original defaults.

Format For Bookings On The Calendar

When **Calpendo** displays *bookings* on the *calendar*, it extracts information from each *booking*. Specify what information it uses, and how it is laid out by providing format specifiers as described below.



Format for Bookings on the Calendar	
Booking day/week title format	Use default format ▼
Booking day/week content format	Use default format ▼
Booking month title format	Use default format ▼

Setting	Description
<i>Booking</i> day/week title format	This specifies what information should be displayed in the title bar of each <i>booking</i> in the day and week view.
<i>Booking</i> day/week content format	This specifies what information should be displayed in the main body of each <i>booking</i> in the day and week view.
<i>Booking</i> month title format	This specifies the content of each <i>booking</i> in the month view.

The format for these settings is specified by providing a single line of text using a particularly complicated syntax. A future version of **Calpendo** will provide a graphical editor so that it's much easier to specify how *bookings* should be laid out on screen. For now, if you want to do this, please contact support@calpendo.com.

Bookings Calendar Start And Finish Time

By default, the *calendar* will display from midnight to midnight with a week view that contains seven days, starting on a Monday. However, these settings can be changed so that, for example, the *calendar* will only show from 8am to 6pm and from Monday to Friday if your building was always closed outside those hours.

Bookings Calendar Start and Finish

Day starting hour	0	▼	
Day finish hour	24	▼	
First day of week	Monday	▼	
Number of days per week	7	▼	
Number of day/week vertical pixels per 30 minutes	30		
Vertical Timeline Day Height	14	▼	Hours ▼

Setting	Description
Day starting hour	Which hour the calendar display will start at
Day finish hour	Which hour the calendar display will finish at
First day of week	Which day of the week the calendar display will start at.
Number of days per week	How many days of the week will be displayed
Number of day/week vertical pixels per 30 minutes	Choose the amount of space allocated to each hour of the day in the calendar
Vertical Timeline Day Height	How much time will be displayed in the Vertical and Horizontal views.

Miscellaneous Settings

There are various miscellaneous settings.

The first two refer to how the [resource](#)⁶⁵⁵ columns will be displayed within the calendar. For each user, their [User Settings](#)³⁴ specifies the default values for *resource* columns. The *global preference* for *resource* columns is similar, but allows an administrator to specify the *resource* columns user settings for newly registered users. The global setting is used only for setting up new users and nothing else.

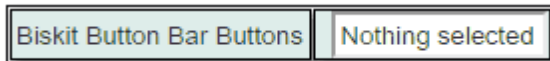
The default *calendar* view type can be overridden by a user in their [User Settings](#)³⁴.

Miscellaneous	
Resource Column Display	Always present ▼
Show header for resource columns	<input checked="" type="checkbox"/>
Default Calendar View Type	Week ▼
Number of days of bookings to check for repeat bookings	56
Request Filtering By User Type	false ▼
Owner Defaults to Project Owner	true ▼
Booking iCal Feeds	iCal Booking Feeds Enabled ▼
Booking Download User	Use default user (ical_viewer) ▼
Time of Day Markers	Show time-of-day markers ▼

Setting	Description	
Resource Column Display	This controls the way <i>bookings</i> are separated into columns within each day in a day or week view. The options available are:	
	Setting	Description
	Share When Possible	<i>Bookings</i> for different <i>resources</i> will be put into the same column as long as they do not occur at the same time
	Separate	<i>Bookings</i> for different <i>resources</i> always appear in separate columns. If there are no <i>bookings</i> for a particular <i>resource</i> , then no column will be shown for that <i>resource</i> .
	Always Present	<i>Bookings</i> for different <i>resources</i> always appear in separate columns, and the column will display even if there happen to be no <i>bookings</i> .
Show header for resource columns	When checked, the <i>bookings calendar</i> will show column headings for each day in the <i>calendar</i> , showing the name or names of the <i>resource(s)</i> displayed in that column.	
Default Calendar View Type	This selects whether to display a Day, Week or Month view by default when Calpendo first starts.	
Number of days of bookings to	When <i>Booking Rules</i> and <i>Time Templates</i> check a <i>booking</i> to see if it's valid, it needs to handle repeat <i>bookings</i> specially. The repeat ⁶⁵⁵ instances are expanded and each instance examined separately.	

Setting	Description
check for repeat bookings	However, since repeats can continue forever, Calpendo cannot check them all. So this setting lets you specify how far into the future Calpendo should expand <i>repeat bookings</i> when checking its <i>Booking Rules</i> and <i>Time Templates</i> . The default is 56 days.
Request Filtering By User Type	Indicates whether the <i>booking</i> requests page should only show requests made by or owned by users of the same type as the current user. This is false by default, so that <i>booking</i> requests are unfiltered.
Owner Defaults to Project Owner	When entering a <i>booking</i> in the calendar, the <i>booking's owner</i> ⁶⁵⁴ will be given a default value. Setting Owner Defaults to Project Owner to true makes the <i>booking's owner</i> default to the <i>project's</i> ⁶⁵⁴ owner (if there is a project), and setting it to false makes the <i>booking's owner</i> default to the user that makes the <i>booking</i> (the <i>booker</i> ⁶⁵²).
Booking iCal Feeds	The administrator can determine whether users can get iCal feeds for their calendars or not. If enabled an iCal button appears in the button bar above the calendar. This is off by default. Users will need a create permission for ICSFeed in order to access this.
Booking Download User	Defines which users permissions will be used when downloading the data. Either the default user (ical_viewer) or the currently logged in user.
Time of Day markers	Defines whether the Time of Day Markers on the calendars will be displayed or not.

6.17.4 Buttons



This controls what buttons appear when viewing a single *biskit*⁶⁵² or a list of *biskits*.

This initially only controls whether the relatively new **"Send Email"** button is displayed, but other buttons are likely to appear here in the future.

6.17.5 Date And Time

The **Date And Time** tab of the **Global Preferences** page specifies formats to use for dates and times within **Calpendo**. These are used in a number of places:

1. In Automatic Emails sent out by the server that contain and dates or times
2. When a new user is registered, the date and time formats of their [user settings](#)³⁴ is copied from the [global preference](#)⁶⁵³
3. The top of a column of the calendar views

Date Format	8 Jan 2010 ▼
Date & Time Format	8 Jan 2010 13:45 ▼
Time Format	1:45 PM ▼
Prefer US Date Format	false ▼
Calendar Day Column Date Format	Fri 08/01 ▼

Setting	Description
Enable SMTP Debugging	<p>This provides an option for logging the details of the transaction with an SMTP server, and can be useful in identifying problems with sending emails.</p> <p>If this is enabled, then every time Calpendo tries to send an email, the system event generated will record the log of the communication with the SMTP server.</p> <p>This may be helpful for debugging, but is a security risk because the entire content of all emails sent will be recorded in the system events.</p>
Authentication	Select whether to use authentication for the connection between your SMTP email server and the Calpendo server, and if so, select the user name and password that should be used by Calpendo to connect.
Emailed Base URL	This is the URL that should be used to get to Calpendo and is used by Email WorkFlow Actions ³⁷³ that reference a URL. It is useful when sending an email that includes a direct link to a page within Calpendo , and the main part of the URL needs to be injected automatically.
Limited Recipients	During testing anything related to email, restrict the email addresses that are allowed to receive email from Calpendo . Provide a comma-separated list of email addresses to restrict outgoing email to go only to those addresses. If this field is left empty, then email will be allowed to go to anybody.
Email BCC	This is a comma-separated list of email addresses that should receive a copy of all outgoing emails from Calpendo . This is useful for checking that Calpendo is sending the expected emails, particular while Calpendo is being configured.
Email Signature	This is a signature that is included at the end of every outgoing email. If left empty, then a default signature will be added.
Email HTML Signature	This is a signature that is included at the end of every outgoing email using HTML. If left empty, then a default signature will be added.

6.17.7 General

The **General** tab of the **Global Preferences** page provides the following:

Icon	
Past Repeatable Minutes Between Flushes	60
Minimum Time Between Database Dumps In Minutes	30
Temporary Directory	/tmp
Run Data-Definition Validation At Boot	false ▼
Read-Only Mode	false ▼
Check Reserved Words	true ▼
Submit crash reports	true ▼
Default Initial Page Token	
Maximum File Upload Megabytes	10

Setting	Description
Icon	This provides the name of an image file that should be used to display on the Bookings Calendar ⁶⁵² and (using the default settings for the appearance headings ⁵¹⁰) on the login page.
Past Repeatable Minutes Between Flushes	Calpendo has some Biskit Types ⁶⁵² that support repeats such as Events ⁶⁵³ , Bookings ⁶⁵² and Time Templates ⁶⁵⁶ . As the time for each instance of the repeat ⁶⁵⁵ passes, a new <i>Booking</i> or <i>Event</i> etc. is created to represent the instance of the repeat that happened. This is done so that, when a repeat <i>Booking</i> (or <i>Event</i> etc) is changed, you do not change the record of what happened in the past. This setting sets up how often Calpendo looks for <i>repeats</i> that have just happened and need a new historical entry creating.
Minimum Time Between Database Dumps In Minutes	This limits the frequency with which database dumps can be created. This is here because database dumps take some time and can degrade system performance, so they don't want to be allowed to be run too often.
Temporary Directory	This is the name of a directory that is used to store temporary files that are created for downloading.
Run Data-Definition Validation At Boot	If Biskit definitions ⁶⁵² are changed using the Bakery ⁵³⁷ , then it is possible that the system may be left in an inconsistent state. This setting disables the boot-time checks that are done to make sure the <i>Biskit definitions</i> are consistent. If there is a problem and boot-time checks are enabled, then Calpendo will not boot properly. If there is a problem and boot-time checks are disabled, then the user may be able to correct the problem in the Bakery ⁵³⁷ , but the issue may cause something to work incorrectly in the meantime.
Read-Only Mode	This indicates whether Calpendo should operate in a read-only mode.
Check Reserved Words	Choose whether to enable protection that stops users from using reserved database words for table, column and property names when using the Bakery ⁵³⁷ . This is particularly important if exporting the Calpendo database to a different type of database.

Setting	Description
Submit Crash Reports	Choose whether to allow crash reports to be emailed to Exprodo Software . Automatic delivery of crash reports are a very useful tool for helping to find and fix problems and are only used for this purpose.
Default Initial Page Token	Choose which page is displayed by default when this program is first started. Enter the text after the '#' in the browser's address bar, and before any non-alphanumeric characters
Maximum File Upload Megabytes	choose the maximum size of a file to be uploaded as an attachment. The default is 10 Megabytes.

6.17.8 Licence

This tab lets the administrator enter a licence key as well as see what the current licence key limits are. When provided with a new key, click the **Enter New Licence Key** button and copy and paste the new licence into the **Enter New Licence Key** text box, and press **OK**. Once you have been returned back to this view make sure you save the changes using the **Global Preferences Save** button.

Program Name Calpendo

Licensed to

XPS Calpendo - for internal use within Conaptic only
UK

Expires 30 Jun 2014

Multiple Logins Yes

Limits

Property	Number Used	Limit
Max Number Of Users	7	Unlimited
Max Number Of Bookings	257	Unlimited
Max Number Of Projects	3	Unlimited
Max Number Of Resources	7	Unlimited
Max Number Of Rules	3	Unlimited
Max Number Of Templates	348	Unlimited

Enter New Licence Key

Indicates whether the same user can be logged in concu

The table at the bottom of the **Licence** area will specify how many of each of the [properties](#) ⁶⁵⁵ are being used and the limit defined by the current key.

6.17.9 Menus

When a new user logs in to **Calpendo** for the first time, they are assigned a menu. This can be overridden by specifying a menu, as described in [Changing A User's Settings](#)¹⁸¹. However, if [settings](#)⁶⁵⁵ for a new user have not been set up, then the menu they are assigned depends on their [user roles](#)⁶⁵⁶, and this is controlled by the **Menus** tab of the **Global Preferences** page:

Default User Menu	User menu ▼
Default Admin Menu	Admin menu ▼
Default Root Menu	Root menu ▼

Setting	Description
Default User Menu	This is the menu to be assigned to users that have neither the Admin nor the Root role .
Default Admin Menu	This is the menu to be assigned to users that have the Admin role but not the Root role .
Default Root Menu	This is the menu to be assigned to users that have the Root role .

6.17.10 Network Metrics

Calpendo can keep information that tracks how long each network call took. This information can be turned on if you want to investigate network performance, but it should normally be left turned off.

The **Network Metrics** tab of the **Global Preferences** page provides the following:

Network Metrics Enabled	false ▼
Number Of Network Calls Between Sending Metrics	20
Minutes Between Sending Metrics	60

Setting	Description
Network Metrics Enabled	If this is set to <i>true</i> , then network metrics ⁶⁵⁴ will be kept, otherwise they will not. Note that by setting this to false, statistics that are already in the database will not be affected. Only new information will be affected by this setting. Enabling Network Metrics will slow down Calpendo a little.
Number Of Network Calls Between Sending Metrics	Indicates how many network calls each web browser must make before sending its <i>network metric</i> information to the server. The larger this number is, the fewer network calls will be created as an overhead. However, larger numbers increase the greater the memory requirements and also increase the amount of data that will be lost when the user closes their browser. This is because outstanding information collected by the browser but not sent to the server is not sent when the browser is closed.
Minutes Between Sending Metrics	This is the period of time between sending <i>metrics</i> to the server. This is only used during periods of low activity when you have some information to send to the server, but you are not active enough to reach the Number Of Network Calls Between Sending Metrics .

If *Network Metrics* are enabled, the data will be stored in the database. In order to access the data use [Search](#)¹¹⁸. Once in the **Search** page set the **Search Biskit Type**⁶⁵² to **Network Metric**, as there could be a lot of information to be returned make sure [Conditions...](#)¹⁰⁷ are used to set at least time limits for the information returned. For more information read the chapter on using [Search](#)¹¹⁸.

6.17.11 Projects

The **Projects** tab of the **Global Preferences** page allows the following to be set up:

Template Project	Temp (Template Project) ▼
Project Rendering Type	Vertical Tabs ▼
Default Project name	Choose project name
Request Filtering By User Type	false ▼
Roles That Can Book Any Project	Admin

Setting	Description										
Template Project	This specifies the project ⁶⁵⁴ that should be used as a template ⁶⁵⁶ for newly constructed <i>projects</i> . See Project Template ²¹⁸ for more details.										
Project Rendering Type	<p>When configuring project properties²¹⁸, a group name can be assigned to each property⁶⁵⁵. Then, when a <i>project</i> is displayed, its <i>properties</i> are shown in these groups according to the group name set on each <i>property</i>. Normally, a <i>project</i> displays its <i>properties</i> with all the <i>property</i> groups visible all at once. However, if a large number of <i>properties</i> and groups are to be created, then it may be better to display those <i>properties</i> in separate tabs, with one tab per group.</p> <p>Then choose the Project Rendering Type to select the way that the different <i>property</i> groups are displayed. The options to use are:</p> <table> <tr> <th>Setting</th><th>Description</th></tr> <tr> <td>Vertical Tabs</td><td>Tabbed vertically, useful for a large number of tabs.</td></tr> <tr> <td>Vertical Tabs (sorted by name)</td><td>As above but sorted by tab name.</td></tr> <tr> <td>Horizontal Tabs</td><td>Tabbed horizontally.</td></tr> <tr> <td>Captions</td><td>All properties visible at the same time, arranged by groups. Below is an example of project information displayed in caption format.</td></tr> </table>	Setting	Description	Vertical Tabs	Tabbed vertically, useful for a large number of tabs.	Vertical Tabs (sorted by name)	As above but sorted by tab name.	Horizontal Tabs	Tabbed horizontally.	Captions	All properties visible at the same time, arranged by groups. Below is an example of project information displayed in caption format.
Setting	Description										
Vertical Tabs	Tabbed vertically, useful for a large number of tabs.										
Vertical Tabs (sorted by name)	As above but sorted by tab name.										
Horizontal Tabs	Tabbed horizontally.										
Captions	All properties visible at the same time, arranged by groups. Below is an example of project information displayed in caption format.										
Default Project Name	When somebody creates a new <i>project</i> , the new <i>project</i> will initially be copied from the Template Project , but then its status will be changed to Requested , and its name will be changed to whatever is specified here for Default Project Name .										
Request Filtering By User Type	If this is set to true, then the <i>project</i> requests page will only show <i>project</i> requests from users with the same user type ⁶⁵⁶ as the currently logged in user. Otherwise, all <i>project</i> requests will be shown.										
Roles That Can Book Any Project	Specify which roles can book using any project.										

6.17.12 Resource Usage

The **Resource Usage** tab of the **Global Preferences** page allows set up of the items below:

Miscellaneous

These are some miscellaneous set up for the [Resource Usage Calendar](#)⁶⁵⁵.

Miscellaneous

Usage Outcome When All Went Well: OK

Resource usage calendar future background colour: [Color Picker]

Reset default colour

Setting	Description
Usage Outcome When All Went Well	The potential resource usage ⁶⁵⁵ outcomes can be customised, but one of them will be used when the user says that everything went well with their session. This setting chooses which outcome is the one recorded when they say it went well.
Resource usage calendar future background colour	Users cannot enter resource <i>usage</i> in the future because it doesn't make sense. This setting allows chooses the colour that will be displayed in the background of the <i>Resource Usage Calendar</i> for future dates.

Format For Actual Usage On The Calendar

This allows selection of the format for items displayed on the *Resource Usage Calendar*.

Format for Actual Usage on the Calendar

Usage day/week title format	Use default format ▼
Usage day/week content format	Use default format ▼
Usage month title format	Use default format ▼

Setting	Description
Usage day/week title format	This specifies what information should be displayed in the title bar of each resource <i>usage</i> in the day and week view.
Usage day/week content format	This specifies what information should be displayed in the main body of each resource <i>usage</i> in the day and week view.
Usage month title format	This specifies the content of each resource <i>usage</i> in the month view.



Note

The format is specified by providing a single line of text using a particularly complicated syntax. A future version of **Calpendo** will provide a graphical editor so that it's much easier to specify how *usage* should be laid out on screen. For now, if this is required, please contact support@calpendo.com.

6.17.13 Rules

The **Rules** tab of the **Global Preferences** page sets up the following:

Rules Apply To Admin	true ▼
----------------------	--------

Setting	Description
Rules Apply To Admin	This chooses whether or not Booking Rules apply to users with the Admin role ⁶⁵⁶ .

6.17.14 Security

The **Security** tab of the **Global Preferences** page sets up the items below:

Miscellaneous

Browser Allowed To Remember Passwords	Browser can remember passwords ▼
Brute Force Password Hacking	Enable brute force password hacking protection ▼
Forgotten Login Names/Passwords	Provide support for automatic password reset ▼
On IP Address Change	Allow session to span IP addresses ▼
On Password Change	Force logout on password change ▼
User Session Timeout In Minutes	100000
Minimum Login Name Length	4
Auto Refresh Delay	2

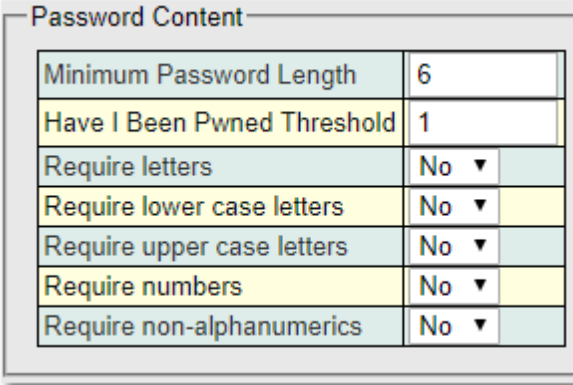
Setting	Description
Browser Allowed To Remember Passwords	Calpendo is configured so that it allows the user's browser to remember their login name and password (if this facility is enabled by each user's browser). Disable this facility at a global level by setting this to false .
Brute Force Password Hacking	This enables or disables protection against brute force password hacking attempts. This should probably always be enabled. The only potential problem with this is that the protection involves disabling logging in from the IP address being used by the hacker. So if there is a hacker on your own network, it's possible that this protection could stop some legitimate users logging in temporarily. This is only ever temporary
Forgotten Login Names/Passwords	Calpendo is configured to allow users to reset their password automatically if forgotten.
On IP Address Change	Calpendo is configured to force a logout if the users IP address changes. Set to Allow session to span IP Address to switch this off.
On Password Change	Calpendo is configured to force users to re-login if they change their password.
User Session Timeout In Minutes	The number of minutes before the users session will timeout.
Minimum Login Name	The minimum number of characters a login name is allowed to be.
Auto Refresh Delay	The delay in minutes between refreshes for those pages that support them.

IP Address Stability

A user can specify whether to force a logout when an IP address changes. Some networks are configured to change the apparent IP address used by the browser at a regular interval. This change in IP address is treated in **Calpendo** as a likely hack attempt and so automatically logs users out when the same session is used from multiple IP addresses. This can now optionally be disabled for networks that normally have unstable IP addresses.

Password Content

This provides control of the content of passwords that people are allowed to use.



Password Content	
Minimum Password Length	6
Have I Been Pwned Threshold	1
Require letters	No ▼
Require lower case letters	No ▼
Require upper case letters	No ▼
Require numbers	No ▼
Require non-alphanumerics	No ▼

Setting	Description
Minimum Password Length	This specifies how many characters must be in each password. A user will not be able to set a password that is shorter than this setting.
Have a Been Pwned Threshold.	This specifies the threshold of how many times the password has been exposed before rejection.
Require lower case characters	Specifies whether passwords must include lower case characters.
Require upper case characters	Specifies whether passwords must include upper case characters.
Require numbers	Specifies whether passwords must include numbers.
Require non-alphanumerics	Specifies whether passwords must include characters that are not numbers or letters.

Passwords are prevented from containing identifying information. Specifically, we prevent passwords from containing:

- given name, family name, middle name
- login name, login identifier
- "exprodo", "calpendo"

6.17.15 System Usage Statistics

Calpendo can keep [statistics](#)⁶⁵⁶ that track how much each user has been using it. *Statistics* are kept by collating user activity into periods of time, so that **Calpendo** can record how much activity there was in each block of time. For each user that has been active in a given block of time, a *statistic* is stored in the database.

The **System Usage Statistics** tab of the **Global Preferences** page lets you provide the following:

System Usage Statistics Enabled	true ▼
System Usage Statistics Minutes Between Flushes	5
System Usage Statistics Minutes Per Statistic	15

Setting	Description
Usage Statistics Enabled	If this is set to true , then <i>usage statistics</i> will be kept, otherwise they will not. Note that by setting this to false , <i>statistics</i> that are already in the database will not be affected. Only new <i>statistics</i> will be affected by this setting. Enabling Usage Statistics will slow down Calpendo a little.
Usage Statistics Minutes Between Flushes	This specifies how often Calpendo should flush <i>statistics</i> out to the database. For performance reasons, this should not be made too small.
Usage Statistics Minutes Per Statistic	This indicates how long each block of time is, during which <i>statistics</i> are put together. The larger this value is, then the less space will be taken in the database, and the lower the time resolution of the <i>statistics</i> .

If **System Usage Statistics** is enabled, the data will be stored in the database. In order to access the data use [Search](#)¹¹⁸. Once in the **Search** page set the search [Biskit Type](#)⁶⁵² to **System Usage**. As there could be a lot of information to be returned make sure you use [Conditions...](#)¹⁰⁷ to set at least time limits for the information returned. For more information read the chapter on using [Search](#)¹¹⁸.

6.17.16 Time Templates

The **Templates** tab of the **Global Preferences** page sets up the following:

Templates Apply To Admin	true ▼
--------------------------	--------

Setting	Description
Templates Apply To Admin	This chooses whether or not Time Templates ⁶⁵⁶ apply to users with the Admin role ⁶⁵⁶ .

6.17.17 Users

The **Users** tab of the **Global Preferences** page sets up the following:

Miscellaneous	
New User Request Requires User Type	Yes - new user requests require a user type ▼
Request Filtering By User Type	false ▼
New User Default Roles	No roles
Message On Blocked User Login	
You are not allowed to log in	
Message On Denied User Login	
You are not allowed to log in	
Message On Expired User Login	
Your account has expired	
Message On Lurker User Login	
You are not allowed to log in	
Message On Requested User Login	
You cannot log in until after your registration request has been approved	

Setting	Description
New User Request Requires User Type	If this is set to true, then whenever a new user registers with Calpendo , they must specify their user type ⁶⁵⁶ . See Configuring Types And Groups ²⁸⁰ for more information about types.
Request Filtering By User Type	If this is set to true, then the user requests page will only show user requests from users with the same <i>user type</i> as the currently logged in user. Otherwise, all user requests will be shown.
New User Default Roles	When a new user is created specify the roles ⁶⁵⁶ they can have by default. See User Roles ¹⁶³ for more information.
User Login Messages	Set up customised messages if the user tries to log in but cannot.

6.18 Bakery

Calpendo is built on [Exprodo](#), a web-based system that is designed to be able to display and edit the contents of a database. To be able to do this, **Exprodo** allows the building of definitions of the data in the database. In other words, while **Calpendo** has a built-in set of data types that it handles, the user can add extra data types and modify the predefined data types by adding new properties and change the way existing properties are displayed.

As well as being able to create definitions of the data in a database so that the user can see and edit the database contents, **Exprodo** provides additional facilities to help with interaction with that data:

- [Email Workflow Actions](#)³⁷³ to be sent in response to data being changed
- Customisable [Menus](#)⁴⁸⁹
- A [Permissions](#)³¹⁴ system based on very adaptable [Conditions](#)¹⁰⁷
- Dynamic reporting tools.

Each of these facilities makes heavy use of the fact that the data types themselves are defined in the database. It lends itself to providing a very flexible system. Therefore, the very heart of the system is the means by which one can view and modify the definitions of the data types, and that is precisely what the [Bakery](#)⁶⁵² does.

The data types themselves are called [Biskit Types](#)⁶⁵², and the definition of the content of a *Biskit Type* is known as a [BiskitDef](#)⁶⁵². Each *BiskitDef* defines certain information, such as its name, the labels used to display that name, the database table that contains the *Biskit Type*, and how to display the name of its [Biskits](#)⁶⁵² (individual occurrences of the *Biskit Type*. i.e. User is a *Bisket Type*, Joe Bloggs would have a *Biskit* in the database defining his information as a specific occurrence of the User *Bisket Type*, with the [properties](#)⁶⁵⁵ of that *Biskit* filled in with his specific information.). It also stores a list of *property* definitions, with each *property* definition defining its *Biskit Type*, how it's stored in the database, how to display the label for the property, the tool tip text to be displayed when the mouse hovers over each *property* value and a number of other aspects of the *property*.

When saving *Biskit Types* and *properties*, **Calpendo** does check that reserved words have not been used for column names, table names, *Biskit Types* and *property* names. There are currently over eight hundred words such as Select, Into, From, Start, Full which are reserved. The system administrator can switch this checking off in [Global Preferences, General](#)⁶²⁵ section, but this is not advisable.

The *Bakery* allows these things to be customised, as well as adding the users own *BiskitDefs* so that **Calpendo** can be used to interact with the users own data.

6.18.1 Property Storage Mechanisms

A database has many tables, with each table having many columns. When a new [Biskit](#)⁶⁵² is created, a new row is added to the *Biskit's* table, containing its [properties](#)⁶⁵⁵. When a new *property* is added to a *Biskit* in **Calpendo**, one would normally also add a new column to the appropriate database table. However, modifying tables like this is not something that all users would be comfortable doing.

Calpendo helps with this by creating table columns automatically, but it will not remove unused columns, and it will not change the [Biskit Type](#)⁶⁵² contained in a column. This provides some protection against accidental destruction of data while changing the *Biskit* definitions, but also means that getting the configuration right does have some difficulties. This is particularly true if there needs to be a change of the type of data stored by a *property*, which has to be handled by creating a new column for it.

However, there is an alternative that does not involve creating a new table column when adding a new *property*. **Calpendo** supports what are called **Indirect** properties which are stored in a separate table in such a way that new *properties* do not require changing the table structure.

In all, there are three *property* storage mechanisms: **Static**, **Dynamic** and **Indirect**. This table compares them:

Comparison	Static Properties	Dynamic Properties	Indirect Properties
Description	Built-in to Calpendo	Created using the Bakery ⁶⁵²	Created using the <i>Bakery</i>
Structure	Each <i>property</i> in a column in the <i>Biskit's</i> main table	Each <i>property</i> in a column in the <i>Biskit's</i> main table	One row added to the <i>Biskit's</i> indirect <i>properties</i> table for each indirect <i>property</i> .
Adding new property	Not possible without a new version of Calpendo	<ul style="list-style-type: none"> • Add in the <i>Bakery</i> • Ask the <i>Bakery</i> to update the database schema • Tell Calpendo server to reload the data definitions when you have finished making all changes 	Add in the <i>Bakery</i>
Advantages	<ul style="list-style-type: none"> • Efficient storage and retrieval • The fastest access mechanism 	<ul style="list-style-type: none"> • Efficient storage and retrieval • Not quite as fast as static properties 	<ul style="list-style-type: none"> • Does not require table structure to be changed to add a new indirect <i>property</i> • Harder to make mistakes and get confused
Disadvantages	Cannot be changed without new version of Calpendo	Need to be aware of database columns and their <i>Biskit Types</i>	Much less efficient storage and retrieval than the other property types

Users of **Calpendo** cannot create or delete **Static Biskits** or *properties* but can change some of the [meta properties](#)⁶⁵⁴ of both the *Biskit* and the *properties*, such as the labels, tool tips, whether required and default values.

Users should be using dynamic *properties* rather than indirect *properties*, unless they really do not wish to change the **DB Schema**, and are happy with the speed of the indirect *properties*.

6.18.2 Biskit Definitions

It can be a bit confusing to talk about definitions of [properties](#)⁶⁵⁵. So let's start by using an example, and looking at the definition of the **Session Biskit**⁶⁵² that is used to store information about who is using **Calpendo**. A **Session** has the following *properties*:

Name	Data Type	Description
user	<i>Biskit</i>	The user whose session this is
started	Date Time	The date/time when this session started
lastUsed	Date Time	The date/time when this session was last used
IPAddress	String	The IP address of the user's computer
sessionID	String	An ID used to represent the session when the user's web browser sends requests to the server
userAgent	String	The web browser's reported user agent information. This tells you which web browser they are using.

Now, the [BiskitDef](#)⁶⁵² for a **Session** must contain information about each of the above six *properties*. For each one, it has to know the Data Type, its name, the label used to display it and so on. However, the **Session BiskitDef** must also store [meta-properties](#)⁶⁵⁴ of **Session** itself, such as its type ("Session"), its database table name and the text used to display the name "Session" in a number of contexts.

There are quite a few different types of property that a *Biskit* can store, and these are described in the next section, [Property Definitions](#)⁶⁴⁴. The *meta-properties* of a *Biskit* are much simpler, and here is what they look like in the editor:

Type	UserGroup
Parent	None
Group	
Primary Key	7
Version	0
Name Property	name
Sort Property	Use default sorting
Version Property	No version number
Created Property	
Updated Property	
Creator Property	
Updater Property	
Format	
Abstract	<input type="checkbox"/>
Enumerable	<input checked="" type="checkbox"/>
Hierarchy Property	
Label Lower	user group
Label Upper	User Group
Labels Lower	user groups
Labels Upper	User Groups
Null Value Label (Read-Only)	Use default
Null Value Label (Read-Write)	Use default
Tooltip	
Visibility	Users
Storage Mechanism	Static
Shareable Table	Not shareable with sub-types
Allows Deletion While Referenced	Does not allow deletion
Table Name	user_groups
ID Column Name	id

Here is a description of what each *meta property* is for:

Name	Data Type	Description
Type	String	This is the internal string that represents the <i>Biskit's</i> type.
Parent	BiskitDef	Some <i>Biskit's</i> extend others (in the programming subclass sense) and this indicates the <i>Biskit Type</i> that is the parent of this one.
Group	String	Only used by the Bakery ⁶⁵² itself, use any group name required on a <i>BiskitDef</i> . Then filter the <i>BiskitDefs</i> that are displayed by setting a filter <i>Biskit</i> group.
Primary Key	Integer	This is the unique database identifier for the <i>BiskitDef</i> .
Version	Integer	Used internally to manage changes to the <i>BiskitDef</i> . Increased by one every time a change is made.
Name Property	String	The name of a <i>property</i> on the <i>Biskit</i> that can be used to represent that <i>Biskit</i> . This is not used if Format is specified. Important to do this or the Calpendo chooses the name for each instance of the <i>Biskit</i> that is created.
Sort Property	String	The name of the <i>property</i> that should be used to sort <i>Biskits</i> of this type.
Version Property	String	The name of the <i>property</i> that should be used to hold the version number of the <i>Biskit</i> . Only those <i>properties</i> of type Int with an Automated Property Type of Create & Update will be used.
Created Property	String	The name of the <i>property</i> that should be used to hold the created date and time of the <i>Biskit</i> . Only those <i>properties</i> of type Datetime with an Automated Property Type of Create or Create & Update will be used.
Updated Property	String	The name of the <i>property</i> that should be used to hold the last modified date and time of the <i>Biskit</i> . Only those <i>properties</i> of type Datetime with an Automated Property Type of Update or Create & Update will be used.
Creator Property	String	The name of the <i>property</i> that should be used to hold the creator of the <i>Biskit</i> . Only those <i>properties</i> of type Biskit with Biskit Def set to User , with an Automated Property Type of Create or Create & Update will be used.
Updater Property	String	The name of the <i>property</i> that should be used to hold the upator of the <i>Biskit</i> . Only those <i>properties</i> of type Biskit with Biskit Def set to User , with an Automated Property Type of Update or Create & Update will be used.
Format	String	Lets the user specify a format for representing the <i>Biskit</i> when there isn't a single <i>property</i> that contains the name. See below ⁵⁴² for more details.
Abstract	Boolean	True if there can be no instances of this <i>Biskit Type</i> , only instances of subtypes.
Enumerable	Boolean	True if there are likely to be a large number of <i>Biskits</i> of this type, too many for display in a list.

Hierarchy Property	String (property name)	<p>Some <i>Biskits</i> represent a hierarchy. For example, there might be a Location Biskit, where the Location "Oxford" is a child of the Location "England", which is a child of "United Kingdom" which is a child of "Europe".</p> <p>A hierarchy is a <i>Biskit</i> that has a <i>property</i> containing a <i>Biskit</i> of the same type, representing the parent in the hierarchy, and also a <i>property</i> containing a set of its children. So, Europe has children "United Kingdom", "France", "Germany" and so on.</p> <p>When a user selects a <i>Biskit</i> from a hierarchy, they are presented with a "tree" widget making it easier for them to choose the right item. For this to work, Calpendo needs to know which <i>property</i> contains the children. This must be a OneToMany Biskit property.</p>
Label Lower	String	Text used to display the <i>Biskit Type</i> , when the context calls for a lower case label.
Label Upper	String	Text used to display the <i>Biskit Type</i> , when the context calls for an upper case label.
Labels Lower	String	Text used to display the <i>Biskit Type</i> , when the context calls for a plural lower case label.
Labels Upper	String	Text used to display the <i>Biskit Type</i> , when the context calls for a plural upper case label.
Null Value Label (Read Only)	String	The label to show for <i>Biskits</i> of the type represented by this <i>BiskitDef</i> when that <i>Biskit</i> is null, and we're displaying in a read only context.
Null Value Label (Read Write)	String	The label to show for <i>Biskits</i> of the type represented by this <i>BiskitDef</i> when that <i>Biskit</i> is null, and we're displaying in a read write context.
Tooltip	String	Text to display as a tool tip describing this type of <i>Biskit</i> .
Visibility	Nobody, Root, Admins, Users or Everybody	Indicates who can see that this <i>Biskit Type</i> exists.
Storage Mechanism	Static or Dynamic	Indicates whether this <i>BiskitDef</i> is defined statically, in the source code for Calpendo , or dynamically in the <i>Bakery</i> .
Shareable Table	Boolean	Shareable with sub-types if you want dynamic sub-types to be able to share this <i>BiskitDefs</i> database table.
Allows Deletion While Referenced	Boolean	Allows Deletion if you want <i>Biskits</i> of this type to be deletable even while something references them.
Table Name	String	The name of the database table for storing instances of this <i>BiskitDef</i> . The table name for a static <i>BiskitDef</i> cannot be modified, since it is defined in the source code for Calpendo . Also, not all static <i>Biskit Types</i> will have a table associated with them, for example if they are <i>abstract</i> or if they are only ever used as components of another <i>BiskitDef</i> .

ID Column Name	String	The name of the column which provides the primary key into the table. In a Master-Slave combination these must be different in the Master and each Slave .
----------------	--------	---

Biskit Format

Set the **Name** *property meta-property* of a *Biskit* to specify the name of the *property* that should be used to represent the *Biskit*. when something more complex than just the label is required. For example, a user has a login name, a given name and a family name, but any one alone can not be used to display the person's name. Instead, a combination of them is required. This includes being able to use the Biskit Type.



Biskit Format Definition

The syntax of the format is rather convoluted, and there will soon be an editor that will let you configure it without having to understand the syntax.

Dynamic Biskits

Use the *Bakery* to create new *Biskit* definitions. These will be *dynamic Biskits*. The user can use **Calpendo** to handle create, read, update and delete values in any table with the following restrictions:

- There must be an integer-valued primary key
- The new dynamic *Biskit* must contain only dynamic *properties*. This means that, if an extra *property* is added to the dynamic *Biskit*, the database table must contain a column for it. If the column doesn't already exist, then **Calpendo** can create the column for the user, or the user can create it.

Dynamic Biskits can be used in exactly the same way as **static Biskits**.

Creating Biskits

Biskits can have links to each other via *properties*, also some *meta-property* values of a *Biskit* depend on having appropriate *properties*, which means that creating *Biskits* and their *properties* is not a linear event. i.e. the user cannot start at the beginning and hope to move step by step to the end saving completed *Biskits* as they go. Sometimes the user will need to create a *Biskit* with its *properties*, save it with errors, create a second *Biskit*, save it with errors, go back to the first *Biskit*, set up new information now available in the first *Biskit* because the second *Biskit* has been created, re save the first *Biskit*, go back to the second *Biskit*, update that with information from the first *Biskit* and save. This means when creating your own *Biskit's* make sure there is a plan on how they are going to connect together.

Inheritance

Biskits can inherit from other *Biskits*, inheriting all their *properties*. To do this set the **Parent meta-property** to be the *Biskit* the new *Biskit* is inheriting from.

When a *Biskit* inherits from another *Biskit* it needs to be specified whether both *Biskits* will share the same table or whether the new *Biskit* will have a table of its own for its own *properties* (those it does not share with the parent). This is done by setting the **Share Super Type Table meta-property**. A *Biskit* can only share the table of the *Biskit* it inherits from, if that *Biskit* has the **Shareable Table meta-property** set to allow it. When inheriting from the **Booking Biskit Type** make sure that the parent and child share a table in order to be able to mutate *booking* from one type to the other effectively, (this is the default).

This may seem very similar to copying a *Biskit*, but there are certain major differences:

- When editing the child *Biskit* the parents *properties* are not seen.
- After adding a *property* to the parent the child automatically has access to the new *property*.
- When changing any *meta-properties* of *properties* in the parent the child *Biskit* will automatically inherit the changes. (e.g. Default Value).
- A *property* in the child cannot be created using the same name as a *property* in the parent.
- If the child *Biskit* has a sibling (same parent *Biskit*) and they all share the same table, do not create *properties* of the same name in both child and sibling. (Currently there are no checks for this but it will cause problems in the database).
- Automatic emails applied to a super-type will automatically apply to any sub-type.
- [Permissions](#)⁶⁵⁴ applied to a super-type will automatically apply to any sub-type but may be overridden in the sub-type.
- If there is a booking sub-type defined, then:
 - 1) [Booking Rules](#)²³⁶ will let you choose the sub-type they apply to. A [Booking Rule](#)⁶⁵² that applies to a super-type will also apply to all descendant types too.
 - 2) A *Booking Rule* that only applies to a particular subtype will only run on a *booking* that both is and was of that subtype or a descendant thereof. This means that if a booking is mutated from one subtype to another, it must be of a suitable type both before and after mutation for the *Booking Rule* to run.
 - 3) *Bookings* can be mutated from one **Booking** super/sub type to another. Any *property* information not shared by the two **Booking Biskit Types** will be lost.

6.18.3 Property Definitions

A **PropertyDef** is the definition of a [property](#)⁶⁵⁵ on a [Biskit](#)⁶⁵². In the example of a **Session**, described in [Biskit Definitions](#)⁵³⁹, we saw that a **Session** has 6 *properties*. Therefore, the [BiskitDef](#)⁶⁵² describing a **Session** contains 6 **PropertyDefs**.

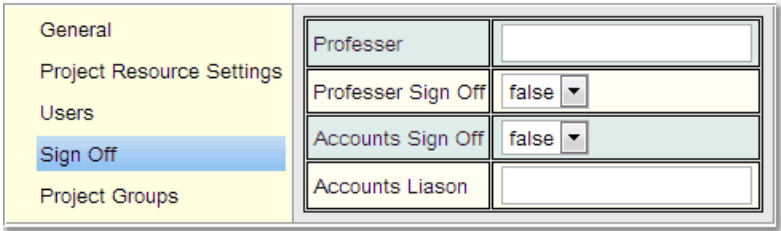
A **PropertyDef** itself has *properties*. For example, it needs to record the name and type of the *Biskit property* it represents. There are quite a few different *property* types:

Type	Description	Type-Specific Content
Biskit	<i>Properties</i> of type <i>Biskit</i> contain a reference ⁶⁵⁵ to another <i>Biskit</i> .	See Biskit Property Definitions ⁵⁴⁹ .
Boolean	Contains a true/false value	See Boolean Property Definitions ⁵⁵⁰ .
Date	Contains a date without a time element	None
DateRange	Contains a range of date/time values	None
Datetime	Contains a date and time	None
Double	A floating point number	See Double Property Definitions ⁵⁵¹ .
Int	The <i>property</i> contains a 32-bit signed integer. This may be interpreted via a drop-down selector if the PropertyDef indicates this is a Mapped Integer ⁵⁵² , and may allow you to set multiple values if it is a Bit Set ⁵⁵⁴ .	See Integer Property Definitions ⁵⁵² .
JavaEnum	The <i>property</i> contains a value that is from an enumeration defined in the source code. This is represented by a drop-down list.	See Java Enum Property Definitions ⁵⁵⁵ . Must specify the Java Enum Definition ⁵⁵⁵ .
Long	The <i>property</i> contains a 64-bit signed integer. Javascript does not directly support 64 bit values, and so long <i>properties</i> may cause performance problems because Calpendo has to emulate the behaviour of a 64 bit integer in the browser.	None
Set	The <i>property</i> contains a set of items. The PropertyDef also needs to define the type of the data contained in the set.	See Set Property Definitions ⁵⁵⁶ .
String	A text <i>property</i> value. The value may be constrained, by specifying a Mapped String ⁵⁶² , in which case the value would be edited using a drop-down list. Otherwise, it will be a free text entry.	See String Property Definitions ⁵⁵⁸ .
StringEnum	A <i>property</i> that contains a string constrained by a String Enumeration ⁵⁶⁴ . Note that these <i>properties</i> are deprecated and may be removed in a future version of Calpendo .	See String Enum Property Definitions ⁵⁶³ . Must specify the String Enumeration ⁵⁶⁴ .
UserDefined	Used when the actual type isn't known in advance. Normally used only for indirect <i>properties</i> .	None

The different *property* types have different requirements about what information must be stored. For example, an integer *property* needs to specify whether it is to have values constrained by being a [Mapped Integer](#)⁶⁵² or a [Bit Set](#)⁶⁵⁴, and in both cases it needs to know which particular [Mapped Integer](#)⁶⁵² should be used. But other *property* types do not need to know anything about [Mapped Integers](#)⁶⁵³.

The requirements of the individual *property* types are shown in the following sections, the rest of this section will focus on the *properties* they have in common.

Primary Key	0
Type	String ▼
Name	myProperty
Description	<input checked="" type="checkbox"/> None
Label	My Property
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	25
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic ▼
Formulaic	<input type="checkbox"/>
Column Name	my_property
Unique	<input type="checkbox"/>
Automated Property Type	None ▼
String Property Type	Single Line ▼
String Type	Unconstrained ▼
Default Value	Null ▼

Name	Description
Primary Key	This is the database ID for the PropertyDef . This cannot be changed, it is only for information purposes.
Type	The type of the <i>property</i> , integer, long, date etc
Name	The name of the <i>property</i> . This must be unique amongst all the <i>property</i> definitions on its <i>Biskit</i> , and contain only letters, numbers and the underscore. In particular, this cannot contain spaces.
Description	A description of the content of this property. This is only used for display in the Bakery ⁶⁵² so that anybody using the <i>Bakery</i> will be able to see this text. It's not used anywhere else.
Label	This is the text used to display the <i>property</i> name. This is used whenever a <i>Biskit's</i> detail is displayed, with each <i>property's</i> label show before its value.
Tooltip	This is a tool tip that should be displayed whenever the mouse hovers over this <i>property</i> when it is being displayed or edited in a <i>Biskit</i> .
Sort Order	Used to change the order in which <i>properties</i> are displayed. This cannot be modified directly. Instead, <i>property</i> sort order is modified by drag-and-drop in the <i>Bakery</i> .
Live	Used internally.
Min	The minimum value the <i>property</i> can take. This is only used for numeric and string values. For numeric properties, the minimum should be a number representing the minimum value the <i>property</i> should have. For string <i>properties</i> , the minimum should be the minimum length of the string. If a minimum is set the string is automatically required.
Max	Similar to Min , this specifies the maximum value the <i>property</i> can take and is only used for numeric and string values.
Attributes	These are defined in a separate table, below.
Rows	The number of rows used to display the value. Particularly useful for multi-line string values.
Columns	The number of columns used to display the value. Particularly useful for multi-line string values.
Group	<p>This can be any text required. In some contexts, <i>Biskits</i> are displayed with their <i>properties</i> in separate groups. In that case, it's the PropertyDef's Group string that defines the group that each <i>property</i> belongs to, and the text that should be used to name that group. If this is used, when viewing a <i>Biskit</i>, separate tabs each with the appropriate group name will be used to display the properties.</p> <p>This <i>property</i> is used if no layout has been defined in the layout editor for this Biskit Type⁶⁵². The Project Biskit will always use this <i>property</i> for its layout.</p> 

Storage Mechanism	Indicates whether the <i>property</i> is static, dynamic or indirect. See Property Storage Mechanisms ⁵³⁷ for more information.
Formulaic	Dynamic only. Allows a formula to be entered to calculate the value of this <i>property</i> . See the section on Formulae ⁵⁶⁴ for more details on the formatting.
Column Name	Only required for dynamic <i>properties</i> , this is the name of the database column used to hold the <i>property</i> .
Unique	Only required for dynamic <i>properties</i> , this indicates whether the value stored in this <i>property</i> should be unique. If it is, then when Calpendo is asked to generate SQL code to update the database schema, it will add the necessary commands to tell the database that the column contains unique values. This will make it illegal for the same non-null value to be present twice.
Automated Property Type	This specifies whether the <i>property</i> is automatically assigned a value, and when. The potential values are: None , Create , Update or Create & Update . If a property has a value other than None then the property will not be viewable when a <i>Biskit</i> is being edited.
Default Value	The value the user will see on creation. If this value is outside the Min and Max values above then the user will have to update before saving.

The *attributes* on a *PropertyDef* are all boolean values as follows:

Name	Description
Visible	Indicates whether the <i>property</i> should ever be displayed. If you make a <i>property</i> invisible, it will always be hidden.
Editable	Specifies whether this <i>property</i> can ever be modified. Some static <i>properties</i> are special and cannot be changed, but you may sometimes have dynamic <i>properties</i> that are calculated from some other source and inserted into the database. You can mark such <i>properties</i> as not being editable to protect them from accidental change.
Persistent	Indicates whether the <i>property</i> is stored in the database. Dynamic and indirect <i>properties</i> should always be persistent.
Null Allowed	Specifies whether null is an allowed value for the property. See below ⁵⁴⁸ for a discussion about Null Allowed and Required .
Required	Specifies whether it is mandatory to provide a value for the property. See below ⁵⁴⁸ for a discussion about Required and Null Allowed .
Visible In Biskit Detail	Specifies whether this <i>property</i> should be visible when the full detail of a <i>Biskit</i> is displayed.
Visible in Biskit List	Specifies whether this <i>property</i> should be visible when a list of <i>Biskits</i> is displayed.
Visible in Collection Editor	Specifies whether this <i>property</i> should be visible when displaying a collection of <i>Biskits</i> of this type. Normally, when editing a collection of <i>Biskits</i> , the number of <i>properties</i> displayed would be kept very small.

Required Fields And Null Allowed

Required and **Null Allowed** sound like they are exact opposites, and therefore **Calpendo** shouldn't support both of them. However, there is a subtle difference that means that it is sometimes useful to define a *property* that is both required and allows null. The **Null Allowed** attribute is checked when building an editor for a *property*. So, when displaying a drop-down for a *Mapped Integer*, *Mapped String*⁶⁵⁴, **Java Enum** or **String Enum**, a value representing **null** will be added if the **Null Allowed** attribute is set to **true**.

When a new *Biskit* is being created, the default value of a *property* displayed with a drop-down with **Null Allowed** set to **true** will be **null**, since that is the first value shown in the drop-down.

However, if a property is set to **Required**, then a value must be provided; it cannot be saved without a non-null value. For properties that have **Required** and **Null Allowed** both set to **true**, this means that the user is forced to choose one of the values from the drop-down as the default value of null cannot be left as the current value. A drop-down for a *property* that is **Required** but for which **Null Allowed** is **false** would be built without a value representing **null**, and so the user would not be forced to choose. Instead, the first value in the drop-down would be automatically selected unless a choice is made by the user.

For string properties, an empty string is a non-null value and so will satisfy **Required**. In order to make sure the user can not do this set a minimum length for strings where you want to make sure they enter something.

Large Numbers Of Properties And Tab Layout

If you are adding a large number of *properties* to a *Biskit* and wish them to be displayed in particular groups under specific tab names then use the **Group meta-property**. All *properties* with the same **Group meta-property** will be displayed under the same tab which will be labelled with the value of the **Group meta-property** unless a layout has been defined for this *Biskit* in the **Layout Editor**. (The **Project Biskit** will always use **Group** for its layout)

General	Professor	<input type="text"/>
Project Resource Settings	Professor Sign Off	false ▼
Users	Accounts Sign Off	false ▼
Sign Off	Accounts Liason	<input type="text"/>
Project Groups		

6.18.3.1 Biskit Property Definitions

A [property](#)⁶⁵⁵ of type [Biskit](#)⁶⁵² must specify the following information:

Biskit Def	None
Component	<input type="checkbox"/>
Biskit Property Type	To One
Reference Deletion Option	Cascade

Name	Meaning
BiskitDef	Specifies the type of <i>Biskit</i> this <i>property</i> will contain.
Component	Indicates whether the <i>Biskit</i> value is stored as a reference ⁶⁵⁵ to a <i>Biskit</i> stored somewhere else, or if its <i>properties</i> are stored directly as a component of its owning <i>Biskit</i> . Only used for static <i>properties</i> .
Biskit Property Type	To One, Many To One, Master to Slave or Slave To Master explained below ⁵⁴⁹ .
Inverse Property	Must be provided when this is a Many To One property, Master to Slave or Slave to Master , in which case this is the name of the <i>property</i> on the parent that points to the child BiskitDef ⁶⁵² .
Reference Deletion Option	Whether this objects of this <i>Biskit</i> type can be deleted if <i>referenced</i> by other <i>Biskits</i> in the DB. Cascade, No Action, Set null

Biskit Property Type

Value	Meaning
To One	The value of this <i>property</i> will <i>reference</i> a <i>Biskit</i> , but that <i>Biskit</i> will not know anything about us <i>referencing</i> them.
Many To One	The value of this <i>property</i> will <i>reference</i> a <i>Biskit</i> in a child-parent relationship. We are the child, and this <i>property</i> <i>references</i> the parent. The parent will also have a Set or List <i>property</i> that contains its child <i>Biskits</i> using a One To Many <i>property</i> . With this value of Biskit Property Type , we must also specify the Inverse Property that is the name of the parent's <i>property</i> that contains the children.
Master to Slave Slave To Master	When a <i>Biskit</i> is required which has a large number of <i>properties</i> , ie. in excess of a thousand in SQL, the DB cannot cope with this. In order to do this create Slave Biskit's , that is <i>Biskits</i> that are created, deleted and <i>referenced</i> with their Master . The Master Biskit would need a <i>property</i> of type <i>Biskit</i> which would point to each Slave and be a Master to Slave type, the Slave would need a <i>property</i> of type <i>Biskit</i> which would point to the Master and be a Slave to Master type. A Master can have many Slaves in order to reference as many <i>properties</i> as needed. The only caveat is that the Primary Key Column Name for any Slave Biskit must be different from the Masters (defaults to id , therefore needs to be changed), and different from any other Slave of that Master . The Inverse Property also needs to be set up in both <i>BiskitDefs</i> . See Creating a Master-Slave Biskit Relationship ⁵⁹⁷ example.

6.18.3.2 Boolean Property Definitions

A boolean *property*, as well as the normal settings required by a **PropertyDef**, may also specify the text to be displayed instead of **True** and **False**.

True Text	<input checked="" type="checkbox"/> Default
False Text	<input checked="" type="checkbox"/> Default

True Text	<input type="checkbox"/> Default	Yes
False Text	<input type="checkbox"/> Default	No

By un-ticking the **Default** option the user can then specify which text is to be used to display the value instead of **True/False**.

When used in a formula the values are **T** or **F**.

6.18.3.3 Double Property Definitions

A double *property*, as well as the normal settings required by a **PropertyDef**, may also specify its **Units**. Doubles are numbers with a decimal point.

Units	None
Default Value	<input type="checkbox"/> Length <input type="checkbox"/> Temperature <input type="checkbox"/> Time <input type="checkbox"/> Weight

There are four types of units **Length**, **Temperature**, **Time**, **Weight**. Users can also add their own units into **Calpendo**. Each type of unit has a number of options, and these are:

Name

Length

Units

Name	Short Name	Multiplier	Sub Units Per Unit	Offset	Sub Unit
Inches	in	0.0254	10	0	
Yards	yd	0.9144	10	0	
Miles	miles	1609.344	10	0	
Feet	ft	0.3048	12	0	Inches
Metres	m	1	10	0	
Millimetres	mm	0.001	10	0	
Centimetres	cm	0.01	10	0	
Feet (decimal)	ft	0.3048	10	0	

Name

Temperature

Units

Name	Short Name	Multiplier	Sub Units Per Unit	Offset	Sub Unit
Centigrade	C	1	10	0	
Kelvin	K	1	10	-273.15	
Fahrenheit	F	0.5555555555555556	10	-17.77777777777778	
Celsius	C	1	10	0	

Name

Time

Units

Name	Short Name	Multiplier	Sub Units Per Unit	Offset	Sub Unit
Seconds	s	1	10	0	
Minutes	mins	60	10	0	
Hours	hours	3600	10	0	
Days	days	86400	10	0	
Micro Seconds	us	0.000001	10	0	
Milli Seconds	ms	0.001	10	0	
Weeks	weeks	604800	10	0	
Months	months	2629800	10	0	
Years	years	31557600	10	0	

Name

Weight

Units

Name	Short Name	Multiplier	Sub Units Per Unit	Offset	Sub Unit
Stones	st	6.35029318	14	0	Pounds
Kilograms	kg	1	10	0	
Grams	g	0.001	10	0	
Ounces	oz	0.028349523125	10	0	
Pounds	lb	0.45359237	10	0	
Stones (decimal)	st	6.35029318	10	0	

6.18.3.4 Int Property Definitions

An integer (a whole number, no decimal information) *property*, as well as the normal settings required by a **PropertyDef**, must also specify its **Integer Type**. This indicates whether the content of the integer value should be constrained. The values allowed are:

Integer Type	Time of Day ▼
Default Value	Unconstrained
	Mapped Integer
	Bit Set
	Time of Day
	Time of Day (with seconds)

Value	Meaning
Unconstrained	The value can be anything.
Mapped Integer	The value will be one of the values specified by a Mapped Integer ⁶⁵² . The particular Mapped Integer ⁶⁵³ that constrains the value must be specified.
Bit Set	The value will be a Bit Set ⁶⁵⁴ containing values specified by a Mapped Integer ⁶⁵² . The particular <i>Mapped Integer</i> that constrains the value must be specified.
Time of Day	Stored in the database as the number of seconds since midnight. The only difference is the formatting on output, with or without seconds.
Time of Day (with seconds)	

6.18.3.4.1 Mapped Integers

A [Mapped Integer](#)⁶⁵³ is a mapping between a text display value and an integer value. It provides a way for a user to edit an integer *property* by seeing a drop-down list of text values, hiding the numerical values from them. This provides a means of generating drop-down boxes for entering values where there is a fixed set of valid values that can be entered, while storing an integer in the database.

Here is an example of a *Mapped Integer* defined in the database:

⊖ Mapped Int

Advanced Modes

CancellationReason

Http Status Code

Modes

PatternFlags

Role

Training

⊕ Mapped String

⊕ Tag Def

Name	CancellationReason
Display Group	
Null Value Label	No reason given

Values

Name	Value
Booked in error	1
Subject cancelled	2
Researcher unavailable	3
Operator unavailable	4
Equipment failure	5
Other	20

To add new values press the **Edit** button and then the **Add** button, the new entry will appear at the bottom, set the text string to be viewed and the number to store it as. Then press **Save**.

Name	CancellationReason
Display Group	
Null Value Label	No reason given

Values

Add Remove

Name	Value
Booked in error	1
Subject cancelled	2
Researcher unavailable	3
Operator unavailable	4
Equipment failure	5
Other	20

There can be as many *Mapped Integers* as needed in the database. Once the *Mapped Integer* is created, to use it, create an integer *property* on a [BiskitDef](#)⁶⁵² and set its **Integer Type** to be *Mapped Integer* and chose the *Mapped Integer* you have created. Use the **Display Group** to bring related items together in the same group to make finding them easier.

Integer Type	Mapped Integer
Mapped Int	None
Default Value	None

None
CancellationReason
Role

The database stores the values as integers. The definition of the *Mapped Integer* in the [Bakery](#)⁶⁵² just maps those numbers to a string for display. At a later date new items can be added to the mapping (new strings and new integers), the mapping can be changed by changing the text associated with a particular integer, then everything in the database that has that integer value will be displayed with the new string.

If there is a mapping where 1 displays as "A", and 2 displays as "B", and then this mapping is changed so that 2 displays as B and 3 displays as A, and there is no mapping for 1, then there won't be a way to display a text value for 1, and there won't be a way to search for it.

6.18.3.4.1.1 Bit Sets

A [Bit Set](#)⁶⁵² is a [BiskitDef](#)⁶⁵² integer property that uses a special sort of [Mapped Integer](#)⁶⁵³. To be used as a *Bit Set*, a *Mapped Integer* must map integer values that are between 0 and 31. It doesn't need to map all 32 possible values, but the values mapped must be within that range.

A *Bit Set* works by using binary values. Each *Bit Set* has a particular value, and when a number of bits have been set the value of the integer is the total of each of the individual bits values added up.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768

Therefore if bits 1, 5 and 7 are set, the value is 162. (2+32+128)

Bit Sets allow the user to combine up to 32 options in any combination required. In **Calpendo** we have one default bit set and that is for [User Roles](#)⁵⁵⁴.

There can be as many *Bit Sets* as needed in the database. Once the *Bit Set* is created as a *Mapped Integer*, to use it, create an integer [property](#)⁶⁵⁵ on a *BiskitDef* and set its **Integer Type** to be *Bit Set* and chose the *Mapped Integer* that has been created.

Integer Type	Bit Set
Mapped Int	None
Default Value	None
	CancellationReason
	Role

6.18.3.4.1.2 User Roles

There is a special [Mapped Integer](#)⁶⁵³ definition called [Role](#)⁶⁵⁶. This is used by the *Roles* bit set property of a **User**. Since it is used as a *Bit Set*, it may only have values of 0 to 31. Value zero is special in this *Mapped Integer*. Any user whose roles [property](#)⁶⁵⁵ has bit 0 set is a **Root** user and [Permissions](#)³¹⁴ do not apply to such a user.

Name	Role
Null Value Label	No roles

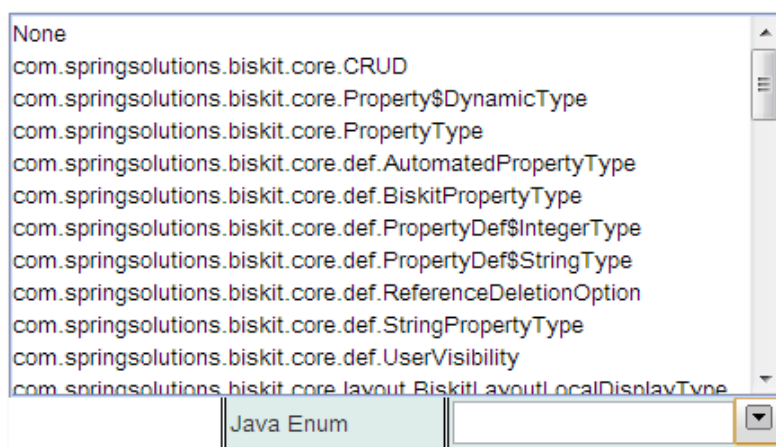
Values	
Name	Value
Root	0
Admin	1
User	2
Guest	3

So using the [Bit Set](#)⁶⁵⁴ bit table a user that has both **Admin** and **User** roles would have a value of 6, the 1 and 2 bits set. A **Guest** user would have the value 8, just the 3 bit set.

The user may add additional values to this *Mapped Integer*, and may rename any of them (including **Root**)

6.18.3.5 Java Enum Property Definitions

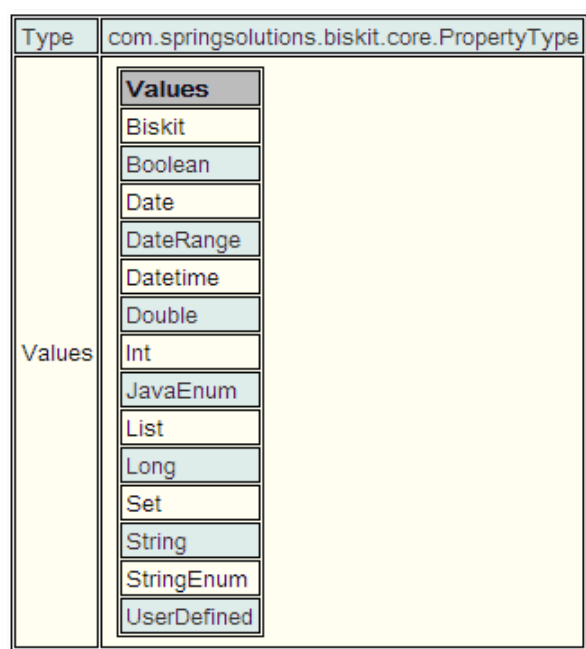
A **Java Enum** [property](#)⁶⁵⁵, as well as the normal settings required by a **PropertyDef**, must also specify its **Java Enum**. This indicates which of the **Java Enums** to be used.



6.18.3.5.1 Java Enum Definitions

A **Java Enum** is a representation of an enumeration defined in the original Java source code. End users should not create, change or delete any **Java Enum** defined in **Calpendo**. You may create [properties](#)⁶⁵⁵ of type **Java Enum** and use any of the existing enumerations, but you should do so carefully in case the enumeration changes or is deleted in a future release.

Properties of type **Java Enum** are edited with a drop-down list box showing the values from the enumeration. In special cases, such a drop-down list box might display only some of the values from the enumeration. However, it is not possible for a user-defined *property* to behave in this manner.



6.18.3.6 Set Property Definitions

A [property](#)⁶⁵⁵ of type **Set** must be of the sub-type [Biskits](#)⁶⁵². The following must then be specified:

Sub-type	Biskit
Sorted	<input type="checkbox"/>
Biskit Def	Equipment Attachment
Component	<input type="checkbox"/>
Biskit Property Type	One to Many
Inverse Property	None

Name	Meaning
Sub-type	The type of data stored by the <i>property</i> represented by this definition. The choice is Biskit or String . String is used when dealing with Tags .
Sorted	Indicates whether the contents of the Set are sorted or not.
BiskitDef	Specifies the type of <i>Biskit</i> the set will contain.
Component	Indicates whether the <i>Biskit</i> value is stored as a reference ⁶⁵⁵ to a <i>Biskit</i> stored somewhere else, or if its <i>properties</i> are stored directly as a component of its owning <i>Biskit</i> . Normally only used for static <i>properties</i> .
Biskit Property Type ⁶⁵⁷	One to Many, Many to Many Persisted, Many To Many Unpersisted, To Many , explained below ⁶⁵⁷ .
Reference Deletion Option	Cascade or Null . Cascade Indicates that the children need to be deleted when the parent is.
Inverse Property	This is the property on the Set Biskit which will point back to the parent <i>Biskit</i>

Biskit Property Type

Value	Meaning
One To Many	Indicates the set contains "child" <i>Biskits</i> that will all <i>reference</i> this <i>Biskit</i> as the "parent" in a Many To One Biskit property . With this value of <i>Biskit Property Type</i> , we must also specify the Inverse Property Name that is the name of the child's <i>property</i> that <i>reference</i> the parent.
Many To Many Persisted	This Set represents the persisted half of a Many To Many property . See Many To Many Properties ⁵⁵⁶ .
Many To Many Unpersisted	This Set represents the unpersisted half of a Many To Many property . See Many To Many Properties ⁵⁵⁶ .
To Many	This Set is a collection of <i>Biskits</i> , but the <i>Biskits</i> we contain do not <i>reference</i> us in any way.

Currently **Many to Many** (in all its forms) is not implemented for dynamic **Set properties**.

Many To Many Properties

Suppose a user can be associated with any number of [projects](#)⁶⁵⁴, and a *project* can have any number of users associated with it. This means there will be a collection of *projects* stored in a *property* on each user, and also a collection of users stored in a *property* on each *project*. Both sides of this relationship are known as **Many To Many properties**.

If a *project* is a member of a user's *projects*, then that same user would be a member of that same *project's* users and vice versa. This means it is not necessary to save both sides of this relationship to the database because of this symmetry. However, there are some occasions when it is necessary to know which side is persisted, and so one half is known as a **Many To Many Persisted**, and the other half is known as **Many To Many Unpersisted**.

6.18.3.7 String Property Definitions


[Property](#)⁶⁵⁵ definitions of type **String** must specify the following information:

String Property Type	Single Line
String Type	Single Line
Default Value	Password
	Multiline
	URL
	Email Address
	HTML

Name	Meaning
String Property Type ⁶⁵⁸	Defines the nature of the content of the string.
String Type ⁶⁵⁹	Indicates whether the content of the text value should be constrained.

String Property Type

This defines the nature of the content of the string, with possible values shown in this table:

Value	Meaning
Barcode	The text which defines a barcode.
Single Line	One line of text.
Password	A single line content that should be hidden when typed. Intended for passwords, but can be used for any text that should not be seen as it is typed.
Multiline	A text area with multiple lines. Typically, the PropertyDef would also specify the number of rows to display.
URL	<p>A single line of text that is expected to contain the web address of some page. When rendered in a read-only format, it displays with an HTML anchor element so that it will operate as a hyperlink.</p> <p>The content can contain a prefix of the form "http://" or "https://" or any other valid protocol recognised by all web browsers, but the protocol prefix is not required.</p>
Email Address	A single line of text that is expected to contain an email address. When rendered in a read-only context, this is displayed with a "mailto:" HTML link so that it will automatically create an email when clicked.
HTML	<p>A text area that contains HTML. When edited, this <i>property</i> will use an HTML editor, with tools for helping you to create valid HTML.</p> 
Colour	Three hexadecimal numbers that define the RGB colour to be displayed.

String Type

This indicates whether the content of the text value should be constrained. The values allowed are:

Value	Meaning
Unconstrained	The text value can be anything.
Mapped String	The value will be one of the values specified by a Mapped String ⁵⁶² . The particular Mapped String ⁶⁵⁴ that constrains the value must be specified.
Biskit Type	The value will be the type of a BiskitDef ⁶⁵² , and so will be configured as a drop-down listing all the known Biskit Types . ⁶⁵²
Tag	Allows the <i>property</i> to be used to store Tags . You must have defined a Tag Def . See Tag Properties ⁵⁶⁷

Barcode

This comes in two parts where PDFs can include barcodes, and also string-valued properties can be displayed in the UI as a barcode.

For examples of all the types of barcode available, see <http://barcode4j.sourceforge.net/examples.html>⁵⁵⁸

The format of the XML required inside the XSL FO file is described at <http://barcode4j.sourceforge.net/2.1/barcode-xml.html>⁵⁵⁸

A string-valued property can also be configured to display its content as a barcode in the browser.

- In the bakery, choose a String Property Type of "Barcode"
- Then select one of the barcode types

These properties are inherently read-only which means they would generally either formulaic properties (that is, generated from other properties) or be created by a workflow.

For details on the different types of barcode available, see <https://github.com/lindell/JsBarcode/wiki>⁵⁵⁸

Also supported are QR Codes, which provide a two-dimensional barcode.

The number of rows and columns assigned to a barcode property has an impact on how it is displayed:

- For one-dimensional barcodes:
 - the number of rows is used as the number of pixels high the barcode should be drawn (default 30).
 - the number of columns indicates the width of each of the thinnest lines drawn (default 1).

- For QR Codes:
 - the number of rows is not used
 - the number of columns indicates the width of each cell displayed in the code (default 1).

QR Code is a trademark of Denso Wave - for more, see <https://www.qrcode.com/>⁵⁵⁸ and <https://www.denso-wave.com/>⁵⁵⁹

Barcode Type

This indicates the type of Barcode to be stored and displayed. The values allowed are:

Value	Meaning
Automatic	Picks the type depending on the data to be stored.
CODE128	CODE128 is one of the more versatile barcodes. It has support for all 128 ASCII characters but does also encode numbers efficiently. It has three modes (A/B/C) but can switch between them at any time. CODE128 is the default barcode that will be chosen if nothing else is defined.
CODE128A	
CODE128B	
CODE128C	
EAN13	EAN comes in a variety of forms, most commonly used is EAN-13 (GTIN-13) that is used on world wide to marking the identity of products. EAN-13, UPC and EAN-8 all have the last digit being a check digit to verify the content that is encoded. This digit is considered a part of the number will be verified before generating the barcode. If the last digit of these barcodes are not specified it will automatically be calculated and added.
EAN8	
UPC	
CODE39	CODE39 is an old barcode type that can encode numbers, uppercase letters and a number of special characters (-, ., \$, /, +, %, and space). It has been a common barcode type in the past.
ITF14	TF-14 (Interleaved Two of Five) is the GS1 implementation of an Interleaved 2 of 5 bar code to encode a Global Trade Item Number. ITF-14 symbols are generally used on packaging levels of a product, such as a case box of 24 cans of soup. The ITF-14 will always encode 14 digits. The last digit of an ITF-14 barcode is a checksum. It is normally included but can automatically be calculated, if it is left out.
ITF	
MSI	MSI or Modified Plessey is a barcode developed by the MSI Data Corporation and is primarily used for inventory control, marking storage containers and shelves in warehouse environments. It supports digits 0-9. It provides automatic checksum calculation of Mod 10, Mod 11, Mod 1010 and Mod 1110.
MSI10	
MSI11	
MSI1010	
MSI1110	

pharmacode	Pharmacode is a barcode used in the pharmaceutical industry. It can encode numbers 3 to 131070.
QR Code (low error correction)	The symbol versions of QR Code range from Version 1 to Version 40. Each version has a different module configuration or number of modules. (The module refers to the black and white dots that make up QR Code.) "Module configuration" refers to the number of modules contained in a symbol, commencing with Version 1 (21 × 21 modules) up to Version 40 (177 × 177 modules). Each higher version number comprises 4 additional modules per side. Each QR Code symbol version has the maximum data capacity according to the amount of data, character type and error correction level. In other words, as the amount of data increases, more modules are required to comprise QR Code, resulting in larger QR Code symbols.
QR Code (medium error correction)	
QR Code (quartile error correction)	
QR Code (high error correction)	

6.18.3.7.1 Mapped Strings

A [Mapped String](#)⁶⁵⁴ is similar in concept to a [Mapped Integer](#)⁶⁵³, but instead of storing an integer in the database, a text value is stored instead. This means that a *Mapped String* is a mapping from one set of strings to another set of strings. When a **String property**⁶⁵⁵ is created on a [BiskitDef](#)⁶⁵², you can indicate that it should be constrained to contain values from a *Mapped String*. Then, when a user edits such a *property*, they will see a drop-down that lists the labels defined in the *Mapped String*, although the value stored in the database will be the associated text value.

Mapped Strings are similar to [String Enumerations](#)⁶⁶⁴, in that they both store values as text, and they both generate a drop-down selection list. *Mapped Strings* are the preferred mechanism though, because by decoupling the displayed labels from the values stored in the database, it makes things very easy if you ever want to change the way values are displayed to users. This is why **String Enumerations** are deprecated, and may be removed in a future version of **Calpendo**.

Here is an example of a *Mapped String* defined in the database:

The screenshot shows the Calpendo database configuration interface. On the left, a tree view shows the hierarchy: 'Mapped Int' (expanded) with sub-items like 'Advanced Modes', 'CancellationReason', 'Http Status Code', 'Modes', 'PatternFlags', 'Role', 'Training', and 'Mapped String' (expanded). Under 'Mapped String', 'BorderStyle' is selected. On the right, the configuration for 'BorderStyle' is shown. It includes a table for 'Name' and 'BorderStyle' with rows for 'Display Group' and 'Null Value Label'. Below this is a 'Values' section with a table listing various border styles and their corresponding database values.

Name	BorderStyle
Display Group	
Null Value Label	

Name	Value
Dashed	dashed
Dotted	dotted
Double	double
3D Grooved Border	groove
3D Inset Border	inset
No border	none
3D Outset Border	outset
3D Ridged Border	ridge
Solid	solid

To add new values press the **Edit** button and then the **Add** button, the new entry will appear at the bottom, set the name of enumeration and the value to store with that name. Then press **Save**. Use the **Display Group** to bring related items together in the same group to make finding them easier.

Name	BorderStyle
Display Group	
Null Value Label	

Values

Add Remove

Name	Value
Dashed	dashed
Dotted	dotted
Double	double
3D Grooved Border	groove
3D Inset Border	inset
No border	none
3D Outset Border	outset
3D Ridged Border	ridge
Solid	solid

6.18.3.8 String Enum Property Definitions

A **String Enum** property, as well as the normal settings required by a **PropertyDef**, must also specify its **String Enum**. This indicates which of the **String Enums** to be used.

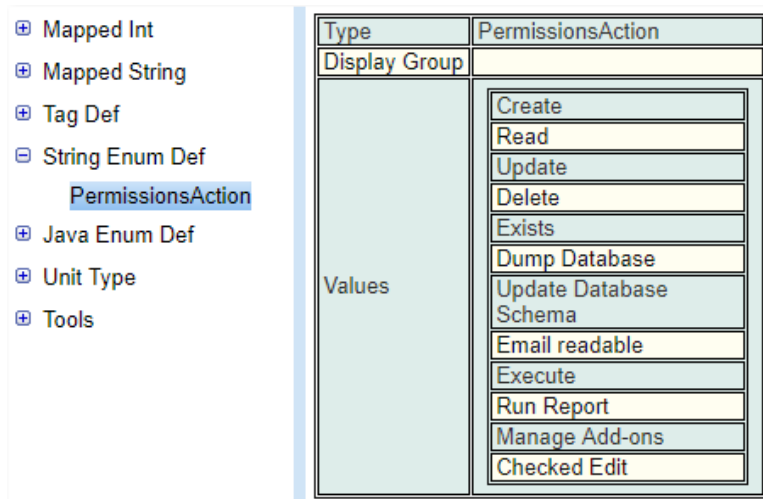
String Enum	None
Default Value	None

Department
PermissionsAction

6.18.3.8.1 String Enumerations

A **String Enumeration** is like a [Mapped String](#)⁵⁶² in which the name and value are always the same. A [property](#)⁶⁵⁵ with a type of **StringEnum** will be editable with a drop-down list box showing all the values in the enumeration.

String Enumerations are deprecated and *Mapped Strings* should be used instead. In some future release, **String Enumerations** may be removed. This is because, by tightly coupling the displayed text with the text stored in the database, it makes it very difficult to change the text that is displayed. Use the **Display Group** to bring related items together in the same group to make finding them easier.



6.18.4 Formulae

If the **Formulaic** check box is ticked then the value of this [property](#)⁶⁵⁵ will be calculated by a formulae.

Formulaic	<input checked="" type="checkbox"/>
Formula	<div></div>

The syntax for the formulae is based on MySQL, with one major exception, MySQL uses the column name in its formulae, most users of **Calpendo** would prefer to use the *property* name. In order to do this there is a special syntax, if the *property* name is in [], such as [durationInMinutes] then **Calpendo** will convert the *property* name to the correct column name. This also allows the user to specify [**biskit.property**] in order to access *properties* of related [Biskits](#)⁶⁵².

When creating formulae use all the standard number calculations * / + - , as well as setting up logic to determine the correct answer.

For instance there are two main logical options **if** and **case**:

Logic	Description
if (expr1, expr2, expr3)	If expr1 is true then expr2 will be returned otherwise expr3 is returned. (You cannot use the if statement)
case when expr1 then option1 when expr2 then option2 else option3 end	If expr1 is true do option1, otherwise if expr2 is true do option2, otherwise do option3. Case statements can be nested one inside the other.

An expression can include < > != (is not null) = () as well as AND or OR for example:

[costPerHour] > 1000 OR ([costPerHour] < 500 AND [costPerHour] > 275)

When dealing with strings there are a number of functions that can be used, here is a subset of the more useful ones:

Function	Description
length(str)	Return the length of the string
concat(str1,str2)	Return str 1 and str2 concatenated together
replace(str1,str2,str3)	In str1, find where str2 occurs, and replace it with str3
substr(str,pos)	Return the string that starts at the pos character of str.

When dealing with dates there are also many functions some of the more useful are:

Function	Description
datediff(date1, date2)	Returns the difference in whole days between two dates.
unix_timestamp(date)	Returns the number of seconds since midnight 1/1/70.
timediff(date1, date2)	Returns the difference between two times as a time string,
time_to_sec(time)	Returns the conversion of a time string to seconds,

Using **time_to_sec()** in conjunction with **timediff()** allows a calculation of the time in seconds between two times. i.e. **time_to_sec(timediff(date1,date2))**

Here is an example of a formulae using a nested case and **datediff()**.

In this example the value of a cost *property* of a [booking](#)⁶⁵² is determined depending on the status of a *booking*.

If the *booking* is denied, or cancelled with more than seven days to go then do not charge. If the *booking* goes ahead then charge the rate found in the *property costPerHour* in the [Project Resource Settings](#)⁶⁵⁴ for that [project](#)⁶⁵⁴ multiplying by the duration in minutes of the booking divided by 60 to give hours. If the *booking* was cancelled less than two days ago charge 50% otherwise charge 20%.

```
case
  when [status] = 'Denied' then 0
  when [status] = 'Approved' or [status] = 'Requested' then
    [projectResourceSettings.costPerHour] * [durationInMinutes] / 60
  else
    case
      when datediff(start_date, [cancelled]) >= 7 then 0
      when datediff(start_date, [cancelled]) >= 2 then
        0.2*[projectResourceSettings.costPerHour] * [durationInMinutes] / 60
      else 0.5*[projectResourceSettings.costPerHour] * [durationInMinutes] / 60
    end
  end
end
```

Known Formulae Problems

There are a number of issues with using formulae in *Biskits*. As the formulae has to be converted from what is written in **Calpendo** to MySQL there are currently some functions that will not work.

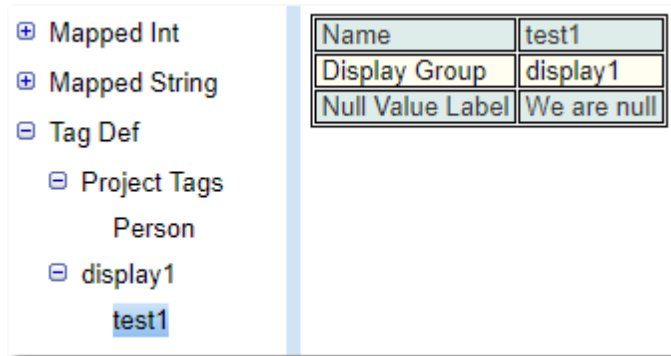
1. Any function that requires a string constant as a parameter. These constants do not currently convert from **Calpendo** to MySQL cleanly.
For example: **timestampdiff**(unit,datetime_expr1,datetime_expr2) where **unit** is one of **FRAC_SECOND** (microseconds), **SECOND**, **MINUTE**, **HOURL**, **DAY**, **WEEK**, **MONTH**, **QUARTER**, or **YEAR**. **Unit** does not convert properly.
2. The **if** statement does not work so use the **if()** function instead.
3. Use **'T'** and **'F'** for the value of Boolean properties.
4. When checking for null use "is not null" not != null as this is always False.

6.18.5 Tag Properties

Calpendo now has support for **Tags**. These are either **String [properties](#)**⁶⁵⁵ or a **Set property** that store a set of **Strings**, and the administrator can choose any value for the string, but will automatically be offered values that have previously been used for the same **Tag**.

To use this, in the **[Bakery](#)**⁵³⁷ there is now a new top-level item labelled **Tag Def** (after **Mapped String**) where the administrator can define the flavour of **Tags** that exist.

In here create the **Tags** required, they can have a **Display Group** and a **Null Value Label**.



Once the type of **Tag** has been defined, the administrator can add the ability to **Tag** a **[Biskit](#)**⁶⁵² by adding a new property to the **[Biskit Def](#)**⁶⁵².

To hold a single **Tag** this would be a property of **Type String** with the **String Type** set to **Tag**. Then choose the **Tag Definition** to be used. This will hold a list of the current **Tags** available in the **Tag Def** section.

To hold multiple **Tags**, add a **Set property**, specify the **Set** contains items of **Type String**, it will default to **Tag** being the type of the string. As above choose the **Tag Definition** to be used.

Tags are stored by name, they also have a **Deprecated** flag, and the **Definition** that defines them. An administrator might want to prevent certain values being offered as suggestions, and to do this they would search for **Tag biskits**, and modify the relevant one to mark it as deprecated. Alternatively, they can just delete it.

6.18.6 Bakery Editor

The **Bakery Editor** allows an **Admin** user to see, modify, create and delete [Biskits](#)⁶⁵² and their [properties](#)⁶⁵⁵. The **Bakery Editor** also allows access to any [Mapped Integers](#)⁶⁵³ or [Strings](#)⁶⁵⁴, any **String** and **Java Enumeration** definitions as well as the unit definitions. To access the **Bakery Editor** page it can be found at **Admin->Bakery**. However, the administrator may have configured **Calpendo** so that the menu is different.

When entering the **Bakery Editor**, in the left hand pane will be all the current definitions for *Biskits* and the additional definitions in an expanding tree structure. The right hand pane will hold the information for whatever is selected in the left hand pane.

Refresh Reload DB Configuration Update DB Schema... Validate Biskits Filter BiskitDef Group...	
Menu Item	Description
Refresh	Does a refresh of all the <i>Biskits</i> in the Bakery .
Reload DB Configuration	Tells the server to reconfigure according to the current dynamic <i>property</i> definitions you have. Make sure the database schema is up-to-date before running this.
Update DB Schema	Asks the server to generate a SQL script that can be used to create whatever columns are required for the dynamic <i>property</i> definitions. After the script has been generated, the user will have the option of applying it.
Validate Biskits	Checks that <i>Biskit</i> definitions are self consistent
Filter BiskitDef Group	Display only those <i>Biskits</i> in a particular group. By default no groups are set up, if your own <i>Biskets</i> are created they should be put into their own group in order to use this filter to find them quickly.

When changes have been made to the DB via the **Bakery Editor** the user **MUST** make sure to update the DB properly by using these buttons in the correct order:

1. Press **Update DB Schema** to generate a script to update the SQL database. Then apply the script to make the changes
2. .Press **Validate Biskit** to make sure any *Biskits* that have been changed have a self consistent definition.
3. Press **Reload DB Configuration** to reconfigure the server so that it is using the new DB schema.

Once this is done the **Calpendo** will now be able to use the changes implemented in the DB through the **Bakery Editor**.

The bottom menu provides buttons to deal with the expanding tree structure.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Use these in conjunction with the **Filter Bisket Group** button on the top menu to aid the viewing of the tree information.

View Mode

View mode allows the user to see each of the *Biskits* and their *properties*. You can also copy *properties* for later use in a different *Biskit*.

The screenshot shows the 'View Mode' interface with the following components:

- Top Bar:** Edit, Create, Create copy, Delete, References, History, Fullscreen, View Mode Tool Bar.
- Left Pane (Biskit Meta-Properties Definitions):**

Type	Calpendo Booking
Parent	None
Filter Group	
Primary Key	167
Version	0
Name Property	dateRange
Sort Property	Use default sorting
Version Property	version
Created Property	created
Updated Property	modified
Creator Property	
Updater Property	
Format	
Abstract	<input type="checkbox"/>
Enumerable	<input type="checkbox"/>
Hierarchy Property	
Label Lower	booking
Label Upper	Booking
Labels Lower	bookings
Labels Upper	Bookings
Null Value Label (Read-Only)	Use default
Null Value Label (Read-Write)	Use default
Tooltip	
Visibility	Everybody
Storage Mechanism	Static
Shareable Table	Shareable with sub-types
Allows Deletion While Referenced	Does not allow deletion
Table Name	bookings
ID Column Name	id
- Center Pane (List of the Biskit's Properties):**

Name	Type
resource	Biskit
booker	Biskit
owner	Biskit
title	String
project	Biskit
projectResourceSettings	Biskit
type	Biskit
status	JavaEnum
previousStatus	JavaEnum
templateApproved	Boolean
warned	Boolean
dateRange	DateRange
lastRepeat	DateRange
created	Datetime
cancelled	Datetime
modified	Datetime
durationInMinutes	Int
allDay	Boolean
description	String
repeat	Biskit
reminderTriggerTime	Datetime
reminderSent	Boolean
remindBooker	Boolean
remindOwner	Boolean
remindProjectOwner	Boolean
remindProjectUsers	Boolean
- Right Pane (The Selected Property's Meta-Properties):**

Primary Key	1149
Type	Biskit
Name	resource
Description	
Label	Resource
Tooltip	The resource being booked
Sort Order	0
Live	false
Min	
Max	
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input checked="" type="checkbox"/> Visible In Collection Editor
Rows	
Columns	
Group	
Storage Mechanism	Static
Automated Property Type	
Biskit Def	Resource
Component	<input type="checkbox"/>
Biskit Property Type	To One
Reference Deletion Option	No Action
Default Value	

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Items that are created will not exist in the database until you press the **Save** button and then follow the [procedure to validate, update and reload the DB](#)⁵⁶⁸.

Properties Tool Bar	Description
Copy	This button will take a copy of the selected <i>properties</i> and place them in a buffer to allow pasting into another <i>Biskit</i> . Multiple <i>properties</i> may be selected by using ctrl and shift .
References	Finds all references to the selected PropertyDef or any reference to a string property path that might refer to properties with the same name as this PropertyDef .

Edit Mode

Edit mode allows the user to edit a *Biskit* and all its *properties*. Once changes are saved, the DB will need to be [validated, updated and reloaded](#)⁵⁶⁸, in order for them to be permanent.

The screenshot displays the 'Edit Mode' interface. At the top left are 'Cancel' and 'Save' buttons. The interface is divided into three main sections:

- Edit Mode Tool Bar:** A list of properties and their values, including Type (Calpendo Booking), Parent (None), Filter Group (None), Primary Key (167), Version (0), Name Property (dateRange), Sort Property (Use default sorting), Version Property (version), Created Property (created), Updated Property (modified), Creator Property (None), Updater Property (None), Format (None), Abstract (false), Enumerable (false), Hierarchy Property (No suitable properties found), Label Lower (booking), Label Upper (Booking), Labels Lower (bookings), Labels Upper (Bookings), Null Value Label (Read-Only) (Use default), Null Value Label (Read-Write) (Use default), Tooltip, Visibility (Everybody), Storage Mechanism (Static), Shareable Table (Shareable with sub-types), Allows Deletion While Referenced (Does not allow deletion), Table Name (bookings), and ID Column Name (id).
- Properties:** A table listing properties and their types:

Name	Type
resource	Biskit
booker	Biskit
owner	Biskit
title	String
project	Biskit
projectResourceSettings	Biskit
type	Biskit
status	JavaEnum
previousStatus	JavaEnum
templateApproved	Boolean
warned	Boolean
dateRange	DateRange
lastRepeat	DateRange
created	Datetime
cancelled	Datetime
modified	Datetime
durationInMinutes	Int
allDay	Boolean
description	String
repeat	Biskit
reminderTriggerTime	Datetime
reminderSent	Boolean
- Properties Tool Bar:** A list of property settings for the selected property (resource):

Primary Key	1149
Type	Biskit
Name	resource
Description	None
Label	Resource
Tooltip	None The resource being booked
Sort Order	0
Live	false
Min	None
Max	None
Attributes	Visible, Editable, Persistent, Null Allowed, Required, Visible In Biskit Detail, Visible In Biskit List, Visible In Collection Editor
Rows	None
Columns	None
Group	None
Storage Mechanism	Static
Automated Property Type	
Biskit Def	Resource
Component	
Biskit Property Type	To One
Reference Deletion Option	No Action
Default Value	Null

A tooltip indicates a 'Drag Area for Drag and Drop of Properties to change sort order'.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Edit Mode Tool Bar	Description
Save	<p>Saves all the changes to the <i>Biskit</i> and its <i>properties</i>. In order to make these changes permanent follow the procedure to validate, update and reload the DB⁵⁶⁸.</p> <p>When saving a changed <i>Biskit</i>, an error message may occur:</p> <div data-bbox="438 1438 976 1615"> <p>Failed to Save BiskitDef</p> <p>BiskitDef is invalid and so cannot be saved</p> <p>OK Show Log Save Despite Errors</p> </div> <p>Look at the log (Show Log button) and then depending on the errors decide whether to Save Despite Errors. Sometimes <i>Biskits</i> need to be saved with errors due to the need to set up links between <i>Biskits</i>, and both <i>Biskits</i> need to be created before that can be done.</p> <p>With some errors there is not the choice of whether to Save Despite Errors, these errors must be fixed before saving because they can corrupt the database.</p>

Properties Tool Bar	Description
Add New	Adds a new <i>property</i> to the <i>property</i> list. The <i>property</i> is automatically selected and available for its meta properties ⁶⁵⁴ to be modified.
Cut	Multiple <i>properties</i> may be selected by using ctrl and shift
Copy	Multiple <i>properties</i> may be selected by using ctrl and shift .
Paste	Pastes the <i>properties</i> held in the buffer into the current <i>property</i> list.
Delete	Multiple <i>properties</i> may be selected by using ctrl and shift .
References	Finds all references to the selected PropertyDef or any reference to a string property path that might refer to properties with the same name as this PropertyDef .

	Name	Type
[drag]	projectCode	String
[drag]	type	Biskit
[drag]	status	JavaEnum
[drag]	owner	Biskit
[drag]	name	String
[drag]	description	Date
[drag]	usersPhone	String
[drag]	principalInvestigator	String
[drag]	finish	Date
[drag]	users	Set
[drag]	principalInvestigator	String
[drag]	start	Date
[drag]	ethics	String
[drag]	fundingAgency	String
[drag]	accountNumber	String

Properties are placed in the list of *properties* for a *Biskit* as defined by their **Sort Order meta-property**.

That **Sort Order meta-property** cannot be edited directly but changed by using the [drag] area, and dragging and dropping selected properties within the property list.

In the example three *properties* have been selected using the **shift** key and are being dragged up the list to be placed higher up.

Editing Mapped Int and Mapped String

When in edit mode for *Mapped Int* and *Mapped String* there are buttons to **Add** and **Remove** items. When an item is added, a new **Name Value** pair appear at the bottom of the list to put in the new mapping. To remove items select a **Name Value** pair first then press the **Remove** button.

Name	Value
Dashed	dashed
Dotted	dotted
Double	double
3D Grooved Border	groove
3D Inset Border	inset
No border	none
3D Outset Border	outset
3D Ridged Border	ridge
Solid	solid
Name	Value

Editing String Enum Def

When in edit mode for **String Enum Def** to add a new value, use the **Enter New Value** box and press **Enter** when finished. The new value is added to the bottom of the list. To remove items select the check box for the items to be removed and then press the **Remove** button. To move an item to a different position in the list use the [drag] column. Select the item to be dragged and then drag and drop it where required.

Type	Department
Enter new value:	
[drag]	<input type="checkbox"/> Anaesthetics
[drag]	<input type="checkbox"/> External
[drag]	<input type="checkbox"/> Clin Neurology
[drag]	<input type="checkbox"/> Clin Psychology
[drag]	<input type="checkbox"/> CVM
[drag]	<input type="checkbox"/> DPAG
[drag]	<input type="checkbox"/> Exp Psychology
[drag]	<input type="checkbox"/> FMRI
[drag]	<input type="checkbox"/> Neuroradiology
[drag]	<input type="checkbox"/> Nuffield Dept of Medicine
[drag]	<input type="checkbox"/> Nuffield Dept of Surgery
[drag]	<input type="checkbox"/> Pharmacology
[drag]	<input type="checkbox"/> Psychiatry
<input type="button" value="Remove"/>	

Editing Units

When in edit mode there is a list view, once a unit is selected there will be a single record view underneath the list view. For more information on [How to Edit Multiple Items At Once](#)¹⁴⁰ in the list view and [How to Edit A Single Item](#)¹⁴² in the record view, read the appropriate sections of the [Data Explorer](#)¹³⁹ chapter

Cancel Save

Name Length

Units

<input type="checkbox"/>	Name	Short Name	Multiplier	Sub Units Per Unit	Offset	Sub Unit
<input type="checkbox"/>	Inches	in	0.0254	10	0	
<input type="checkbox"/>	Yards	yd	0.9144	10	0	
<input type="checkbox"/>	Miles	miles	1609.344	10	0	
<input type="checkbox"/>	Feet	ft	0.3048	12	0	Inches
<input type="checkbox"/>	Metres	m	1	10	0	
<input checked="" type="checkbox"/>	Millimetres	mm	0.001	10	0	
<input type="checkbox"/>	Centimetres	cm	0.01	10	0	
<input type="checkbox"/>	Feet (decimal)	ft	0.3048	10	0	

↑ Edit Remove

Remove Add Apply Edit Cancel Create Create Copy Printable View

Unit 29

Name	Millimetres
Short Name	mm
Multiplier	0.001
Sub Units Per Unit	10
Offset	0
Sub Unit	

Tools

Under the "Tools" option at the top-level of the bakery, there's a tool to help locate references to any property path. Ttype in the name of a property (or something that used to be a property) or the name of a path that might have been used somewhere, such as in conditions. This could be something like "New.owner" or other things with a period in them as well as a simple property name and get all the references to this.

Enter a property name or path Check references

6.18.6.1 Bakery Editor Examples

6.18.6.1.1 Adding Properties

An example of adding a [property](#)⁶⁵⁵ to the [Biskit](#)⁶⁵² of type **Project** called **ProfessorSignOff**, of type **boolean** which is required to always have a value and starts with the value **False**.

1. Go to **Admin->Bakery**,
2. Click the + next to **Biskit Def** to open the **Biskit Tree**
3. Select the *Biskit* to add *properties* to (in this case **Base Project** then **Project**)
4. Press the **Edit** button to get the *Biskit* into edit mode

Name	Type
projectCode	String
type	Biskit
status	JavaEnum
owner	Biskit
description	String
ownerPhone	String
resourceSettings	Set
start	Date
finish	Date
users	Set
principalInvestigator	String
ethics	String
fundingAgency	String
accountNumber	String
version	Int
created	Datetime
updated	Datetime
serviceSettings	Set
ethicsAttachment	Biskit
fileAttachment	Biskit
self	Biskit

Primary Key	22
Type	String
Name	projectCode
Description	<input checked="" type="checkbox"/> None
Label	Project Code
Tooltip	<input type="checkbox"/> None A unique code for this projec
Sort Order	0
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input checked="" type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Static
Automated Property Type	
String Property Type	Single Line
String Type	Unconstrained
Default Value	Null

5. Press the **Add New** button to add a new *property*.

The screenshot shows the 'Add New' button in the top left corner. Below it is a table of existing properties. To the right is the configuration form for a new property named 'myProperty'. The form has the following fields and values:

Primary Key	0
Type	String
Name	myProperty
Description	<input checked="" type="checkbox"/> None
Label	My Property
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	22
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic
Formulaic	<input type="checkbox"/>
Column Name	my_property
Unique	<input type="checkbox"/>
Automated Property Type	None
String Property Type	Single Line
String Type	Unconstrained
Default Value	Null

Annotations:

- a) Choose Property Type (points to Type)
- b) Give the property a name (points to Name)
- c) Is it required to have a value (points to Required checkbox)
- d) Will it have a default value (points to Default Value)

6. Update the [meta properties](#)⁶⁵⁴ of the *property*. Most of them will not need to be changed but as a minimum

- Decide on the **Type** required.
- Give the *property* a name, this needs to be unique, **Calpendo** will update the **Label** and **Column Name** *meta properties* at the same time. Usually these can be left alone, but **Label** may need to be modified as this is what is displayed for this *property* in reports etc.
- When a *Biskit* of type **Project** is created by a user, the user is required to give this *property* a value. (**Null Allowed**, **Min** and **Max** also have an impact on data requirements and may be used instead of or in conjunction with **Required**.)
- As the *property* will have a default value when a *Biskit* of type **Project** is created.

The screenshot shows the Calpendo metadata editor with two panes. The left pane lists various properties, with 'professorSignOff' selected. The right pane shows the configuration for this property. Annotations highlight specific settings: 'Set To Boolean' points to the Type dropdown; 'Set to professorSignOff' points to the Name field; 'Required' points to the Required checkbox; and 'Set to False' points to the Default Value dropdown.

Name	Type
projectCode	String
type	Biskit
status	JavaEnum
owner	Biskit
description	String
ownerPhone	String
resourceSettings	Set
start	Date
finish	Date
users	Set
principalInvestigator	String
ethics	String
fundingAgency	String
accountNumber	String
version	Int
created	Datetime
updated	Datetime
serviceSettings	Set
ethicsAttachment	Biskit
fileAttachment	Biskit
self	Biskit
professorSignOff	Boolean

Primary Key	0
Type	Boolean
Name	professorSignOff
Description	None
Label	Professor
Tooltip	None
Sort Order	22
Live	false
Min	None
Max	None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input checked="" type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	None
Columns	None
Group	None
Storage Mechanism	Dynamic
Formulaic	
True Text	Default
False Text	Default
Column Name	professor_sign_off
Unique	
Automated Property Type	None
Default Value	Specified value false

7. Press the **Save** button.
8. If there are no errors then update the database:
 - a) Press **Update DB Schema** to implement changes in the DB, the following should pop up:

The dialog box titled 'Update Database Schema' displays the following database script:

```
alter table projects add column professor_sign_off char(1) default 'F'
```

Buttons for 'Cancel' and 'Apply Changes' are visible at the bottom.

9. Press the **Apply Changes** button to update the database. There will be a response saying how many rows affected. Press **OK** to finish.

- a) Press **Validate Biskits** button to check the database *Biskits*.
- b) Press **Reload Database Configuration**, to load the new database into **Calpendo**
- c) Refresh the browser

Remember if adding a large number of *properties* to a *Biskit* and they need to be displayed in particular groups under specific tab names then use the **Group meta-property**. All *properties* with the same **Group meta-property** will be displayed under the same tab which will be labelled with the value of the **Group meta-property**, unless a layout has been defined for this *Biskit* in the **Layout Editor**.

The screenshot shows the Calpendo interface with a sidebar on the left containing tabs: 'General', 'Project Resource Settings', 'Users', 'Sign Off' (selected), and 'Project Groups'. The main area displays a form for the 'Sign Off' tab with the following fields:

Professor	
Professor Sign Off	false
Accounts Sign Off	false
Accounts Liason	

6.18.6.1.2 Adding Formulaic Properties

This is an example of adding a formulaic [property](#)⁶⁵⁵ to the [Biskit](#)⁶⁵² of type **Booking**, which will be called **BookingCost** and will calculate the cost of the [booking](#)⁶⁵² depending on the duration of the *booking* and whether it was cancelled or not.

If a *booking* was **Denied**, or **Cancelled** more than 7 days before it was due, then no cost. If it was **Cancelled** more than 2 but less than 7 days ago, then 20% of cost, if it was **Cancelled** with less than 2 days to go then 50% of the cost otherwise full cost.

The formula used will be:

```
case
  when [status] = 'DENIED' then 0
  when [status] = 'APPROVED' or [status] = 'REQUESTED' then
    [projectResourceSettings.costPerHour] * [durationInMinutes]/60
  else
    case
      when datediff([dateRange.start], [cancelled]) >= 7 then 0
      when datediff([dateRange.start], [cancelled]) >= 2 then
        0.2*[projectResourceSettings.costPerHour] * [durationInMinutes]/60
      else 0.5*[projectResourceSettings.costPerHour] * [durationInMinutes]/60
    end
  end
end
```

[status] will refer to a *property* called **status** found on the **Booking Biskit**.

[projectResourceSettings.costPerHour] will refer to a *property* called **costPerHour** that is found on the *Biskit* that is [referenced](#)⁶⁵⁵ by the **projectResourceSettings** *property* of the **Booking Biskit**. In this case the type of *Biskit* referenced is **Project Resource Settings**.

[durationInMinutes] will refer to a *property* called **durationInMinutes** found on the **Booking Biskit**.

[dateRange.start] will refer to a *property* called **dateRange** found on the **Booking Biskit**. **dateRange** is of type **DateRange**, this type has some sub types, one of which is **start**. **start** holds the date and time of the start of the date range.

[cancelled] will refer to a *property* called **cancelled** found on the **Booking Biskit**.

datediff is a MySQL function that returns the difference between two dates as a whole number of days. For more accurate time differences use a combination of **timediff()** and **time_to_sec()** (i.e. **time_to_sec(timediff(date1,date2))** returns the number of seconds between the two dates).

We will be using the **costPerHour** *property* that can be set up for each [resource](#)⁶⁵⁵ a [project](#)⁶⁵⁴ can use. This way individual *projects* can be set up to pay different amounts for each *resource*.

1. Go to **Admin->Bakery**,
2. Press the **Open All** button to open the **Biskit Tree**
3. Select the *Biskit* to add *properties* to (in this case **Booking**)
4. Press the **Edit** button to get the *Biskit* into edit mode

5. Press the **Add New** button to add a new *property*.
6. Update the *meta properties* of the *property*. Most of them do not need to be changed but as a minimum
 - a) Change **Type** to double.
 - b) Change **Name** to **BookingCost**.
 - c) Make it not **Editable**.
 - d) Make it **Formulaic**.

The screenshot shows the 'Properties' dialog with a list of existing properties on the left and a configuration panel on the right. The 'BookingCost' property is selected in the list. The configuration panel on the right has the following settings:

- Primary Key:** ☐ (unchecked)
- Type:** Double (dropdown menu)
- Name:** BookingCost
- Description:** ☒ None
- Label:** Booking C
- Tooltip:** ☒ None
- Sort Order:** 25
- Live:** false
- Min:** ☒ None
- Max:** ☒ None
- Attributes:**
 - ☒ Visible
 - ☐ Editable (annotated with 'Set to not Editable')
 - ☒ Persistent
 - ☒ Null Allowed
 - ☐ Required
 - ☒ Visible In Biskit Detail
 - ☒ Visible In Biskit List
 - ☐ Visible In Collection Editor
- Rows:** ☒ None
- Columns:** ☒ None
- Group:** ☒ None
- Storage Mechanism:** Dynamic (dropdown menu)
- Formulaic:** ☒ (annotated with 'Set to Formulaic')
- Formula:** (empty text area)
- Units:** None (dropdown menu)

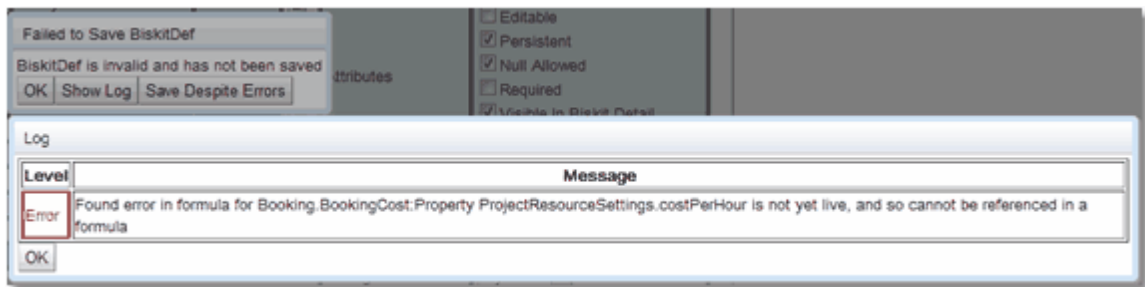
7. Enter the formula required.

The screenshot shows the 'Formula' field in the Properties dialog. The formula entered is:

```

case
  when [status] = 'DENIED' then 0
  when [status] = 'APPROVED' or [status] = 'REQUESTED' then [projectResourceSettings.costPerHour] *
    [durationInMinutes]/60
  else
    case
      when datediff([dateRange.start], [cancelled]) >= 7 then 0
      when datediff([dateRange.start], [cancelled]) >= 2 then 0.2*[projectResourceSettings.costPerHour] *
        [durationInMinutes]/60
      else 0.5*[projectResourceSettings.costPerHour] * [durationInMinutes]/60
    end
  end
end
  
```

8. Press the **Save** button. An error box will appear:



Check the error by pressing **Show Log**. If the error is complaining about a *property* not being live yet then press the **OK** button, and then the **Save Despite Errors** button.

9. As this is adding a formulaic *property* we only need to reload the database configuration.
- a) Press **Reload Database Configuration**, to load the new database into **Calpendo**
 - b) Refresh the browser

Go and check *bookings* to see if the **Booking Cost** is present. In this case if when viewing a *booking* in the [Booking Calendar](#)⁶⁵² at the bottom of the **Booking** will appear the **Booking Cost**.

6.18.6.1.3 Adding Properties For File Attachments

An example of adding a [property](#)⁶⁵⁵ to the [Biskit](#)⁶⁵² of type **Project** to hold an file attachment.

1. Go to **Admin->Bakery**,
2. Click the + next to **Biskit Def** to open the **Biskit Tree**
3. Select the *Biskit* to add *properties* to (in this case **Base Project** then **Project**)
4. Press the **Edit** button to get the *Biskit* into edit mode

Name	Type
projectCode	String
type	Biskit
status	JavaEnum
owner	Biskit
description	String
ownerPhone	String
resourceSettings	Set
start	Date
finish	Date
users	Set
principalInvestigator	String
ethics	String
fundingAgency	String
accountNumber	String
version	Int
created	Datetime
updated	Datetime
serviceSettings	Set
ethicsAttachment	Biskit
fileAttachment	Biskit
self	Biskit

Primary Key	22
Type	String
Name	projectCode
Description	<input checked="" type="checkbox"/> None
Label	Project Code
Tooltip	<input type="checkbox"/> None A unique code for this projec
Sort Order	0
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input checked="" type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Static
Automated Property Type	
String Property Type	Single Line
String Type	Unconstrained
Default Value	Null

5. Press the **Add New** button to add a new *property*.

Properties

Add New Cut Copy Paste Delete

Name	Type
[drag] projectCode	String
[drag] type	Biskit
[drag] status	JavaEnum
[drag] owner	Biskit
[drag] name	String
[drag] description	String
[drag] ownerPhone	String
[drag] start	Date
[drag] finish	Date
[drag] principalInvestigator	String
[drag] fundingAgency	String
[drag] ethics	String
[drag] accountNumber	String
[drag] resourceSettings	Set
[drag] users	Set
[drag] ProfessorName	String
[drag] SignOff	Boolean
[drag] SignOffDate	Date
[drag] ProfesorRole	String
[drag] Attachment	Biskit

Primary Key	0
Type	Biskit
Name	Attachment
Description	<input checked="" type="checkbox"/> None
Label	Attachment
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	19
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic
Formulaic	<input type="checkbox"/>
Column Name	attachment
Unique	<input type="checkbox"/>
Automated Property Type	None
Biskit Def	Attachment
Component	<input type="checkbox"/>
Biskit Property Type	To One
Reference Deletion Option	No Action
Default Value	Null

a) Change Type to Biskit

b) Change the name to Attachment

c) Change Biskit Def to Attachment

d) Change Biskit Property Type to To One

6. Update the [meta properties](#)⁶⁵⁴ of the *property*.
 - a) Change **Type** to **Biskit**.
 - b) Give the *property* a name, in this case **Attachment**.
 - c) Change **Biskit Def** to **Attachment**
 - d) Change **Biskit Property Type** to **To One**.
7. Press the **Save** button.
8. If there are no errors then update the database:
 - a) Press **Update DB Schema** to implement changes in the DB, a pop up will appear defining the database changes to be implemented.
9. Press the **Apply Changes** button to update the database. There will be a response saying how many rows affected. Press **OK** to finish.
 - a) Press **Validate Biskits** button to check the database *Biskits*.
 - b) Press **Reload Database Configuration**, to load the new database into **Calpendo**
 - c) Refresh the browser

6.18.6.1.4 Adding Properties For Created/Updated/Version

An example of adding a [properties](#)⁶⁵⁵ to a [Biskit](#)⁶⁵² created by a user to record when a *Biskit* was created, updated and store which version is currently in use.

1. Go to **Admin->Bakery**,
2. Click the + next to **Biskit Def** to open the **Biskit Tree**
3. Select the user created *Biskit* to add *properties* to
4. Press the **Edit** button to get the *Biskit* into edit mode
5. Press the **Add New** button to add a new *property*. First create a property called for example **Created**.

Properties

Add New Cut Copy Paste Delete

	Name	Type
[drag]	name	String
[drag]	serialNumber	String
[drag]	attachment	Biskit
[drag]	Created	Datetime

Primary Key	0	a) Set the Type to be DateTime
Type	Datetime	
Name	Created	b) Give the property a name
Description	<input checked="" type="checkbox"/> None	
Label	Created	
Tooltip	<input checked="" type="checkbox"/> None	
Sort Order	3	
Live	false	
Min	<input checked="" type="checkbox"/> None	
Max	<input checked="" type="checkbox"/> None	
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor	
Rows	<input checked="" type="checkbox"/> None	
Columns	<input checked="" type="checkbox"/> None	
Group	<input checked="" type="checkbox"/> None	
Storage Mechanism	Dynamic	
Formulaic	<input type="checkbox"/>	
Column Name	created	
Unique	<input type="checkbox"/>	
Automated Property Type	Create & Update	c) Set the Automated Property Type to Create & Update
Default Value	Null	

6. Update the [meta properties](#)⁶⁵⁴ of the *property*.
 - a) Change **Type** to **Date Time**.
 - b) Give the *property* a name, in this case **Created**.
 - c) Change **Automated Property Type** to **Create & Update**.
7. Do the same again to create a *property* for example called **Updated**.
8. And again to create a *property* for example called **Version**, this time make it of **Type Int**.

Type	Equipment
Parent	None
Group	<input type="checkbox"/> None <input checked="" type="checkbox"/> Nursery
Primary Key	132
Version	15
Name Property	name
Sort Property	Use default sorting
Version Property	Version
Created Property	Created
Updated Property	Updated
Format	<input checked="" type="checkbox"/> None
Abstract	<input type="checkbox"/>
Enumerable	<input checked="" type="checkbox"/>
Hierarchy Property	No suitable properties found
Label Lower	equipment
Label Upper	Equipment
Labels Lower	equipments
Labels Upper	Equipments
Null Value Label (Read-Only)	Use default
Null Value Label (Read-Write)	Use default
Tooltip	
Visibility	Everybody
Storage Mechanism	Dynamic
Shareable Table	Shareable with sub-types
Allows Deletion While Referenced	Does not allow deletion
Table Name	equipment
ID Column Name	id

a) Set Version Property to Version

b) Set Created Property to Created

c) Set Updated Property to Updated

New properties, Created, Updated, Version

Properties

	Name	Type
[drag]	name	String
[drag]	serialNumber	String
[drag]	attachment	Biskit
[drag]	Created	Datetime
[drag]	Updated	Datetime
[drag]	Version	Int

9. Set up the meta-property information

- a) Set **Version Property** meta-property to **Version**.
- b) Set **Created Property** meta-property to **Created**
- c) Set **Updated Property** meta-property to **Updated**

10. Press the **Save** button.

11. If there are no errors then update the database:

- a) Press **Update DB Schema** to implement changes in the DB, a pop up will appear defining the database changes to be implemented.

12. Press the **Apply Changes** button to update the database. There will be a response saying how many rows affected. Press **OK** to finish.

- a) Press **Validate Biskits** button to check the database *Biskits*.
- b) Press **Reload Database Configuration**, to load the new database into **Calpendo**
- c) Refresh the browser

6.18.6.1.5 Adding A New Yes-No Mapped Int Property

An example of adding a new [Mapped Int](#)⁶⁵³ to the Database for use in integer [properties](#)⁶⁵⁵, giving **Yes-No** as the drop down values rather than **True-False**.

1. Go to **Admin->Bakery**,
2. Open up the **Mapped Int** tree
3. Press the **Create** button to create a new *Mapped Int*
4. Update the [meta properties](#)⁶⁵⁴.
 - a) Change **Name** to **Yes No**
 - b) Change **Null Value Label** to **Not Selected**
5. Use the **Add** button under **Values** to create a new value
 - a) Change **Name** to **Yes**
 - b) Change **Value** to **1**
6. Use the **Add** button under **Values** to create a new value
 - a) Change **Name** to **No**
 - b) Change **Value** to **0**

Name	Yes No	5a) Change Name to Yes No
Null Value Label	Not Selected	5b) Change Null Value Label to Not Selected

Values

Add Remove Button to add new values

Name	Value
Yes	1
No	0

6a) Change Name to Yes 6b) Change Value to 1

7a) Change Name to No 7b) Change Value to 0

8. Press the **Save** button
9. Refresh the browser
10. Press the **Open All** button to open the **Biskit Tree**
11. Select the [Biskit](#)⁶⁵² to add [properties](#)⁶⁵⁵ to (in this case **Project** under **Base Project**)
12. Press the **Edit** button to get the *Biskit* into edit mode
13. Press the **Add New** button to add a new *property*.

14. Update the *meta properties* of the *property*

- a) Change **Type** to **Int**.
- b) Change **Name** to something relevant (in this case **SignedOff**)
- c) Change **Integer Type** to **Mapped Integer**
- d) Change **Mapped Int** to **Yes No**.
- e) Change **Default Value** to **Specified Value** and **Not Selected**

Properties

Add New Cut Copy Paste Delete

Name	Type
[drag] projectCode	String
[drag] type	Biskit
[drag] status	JavaEnum
[drag] owner	Biskit
[drag] name	String
[drag] description	String
[drag] ownerPhone	String
[drag] start	Date
[drag] finish	Date
[drag] principalInvestigator	String
[drag] fundingAgency	String
[drag] ethics	String
[drag] accountNumber	String
[drag] resourceSettings	Set
[drag] users	Set
[drag] SignedOff	Int

Primary Key: 0

Type: Int

Name: SignedOff

Description: ☒ None

Label: Signed Off

Tooltip: ☒ None

Sort Order: 15

Live: false

Min: ☒ None

Max: ☒ None

Attributes: ☒ Visible, ☒ Editable, ☒ Persistent, ☒ Null Allowed, ☐ Required, ☒ Visible In Biskit Detail, ☒ Visible In Biskit List, ☐ Visible In Collection Editor

Rows: ☒ None

Columns: ☒ None

Group: ☒ None

Storage Mechanism: Dynamic

Formulaic: ☐

Column Name: signed_off

Unique: ☐

Integer Type: Mapped Integer

Mapped Int: Yes No

Default value: Specified value Not Selected

a) Change Type to Int

b) Change Name to SignedOff

c) Change Integer Type to Mapped Integer

d) Change Mapped Int to Yes No

e) Change Default Value to Specified Value and Not Selected

15. Press the **Save** button. Press **Save Despite Errors** if required.

16. If there are no errors the database needs to be updated:

- a) Press **Update DB Schema** to implement changes in the DB, and then run the script to apply the changes
- b) Press **Validate Biskits** button to check the database *Biskits*
- c) Press **Reload Database Configuration**, to load the new database into **Calpendo**
- d) Refresh the browser

Looking at a **Project** [*BiskitDef*](#)⁶⁵² in the **Create Project** view we see:

Project Code	
Type	Please select a Project Type ▾
Status	Requested ▾
Owner	admin (admin) ▾
Name	Choose project name
Description	Describe your project here
Phone Number	
Start	
Finish	
Principal Investigator	
Funding Agency	
Ethics Approval Number	
Account Number	
Signed Off	Not Selected ▾

Not Selected

No

Yes

The **Default Value** could have been set up as either **No** or **Yes**, rather than **Not Selected**, also the order the values appear depends on the order of the *Mapped Int*, so if **No** has a higher value than **Yes** it will appear below **Yes**.

6.18.6.1.6 Creating A New Basic BiskitDef

An example of creating a [BiskitDef](#)⁶⁵² called **ProjectInfo** which has one [property](#)⁶⁵⁵ called **Name** which is of type **String**.

1. Go to **Admin->Bakery**.
2. Press the **Create** button to create a new *Biskit* and enter edit mode.
3. Update the [meta-properties](#)⁶⁵⁴ of the *Biskit*.
 - a) Change **Type** to be the new *BiskitDef* name, in this case **ProjectInfo**
 - b) Change **Filter Group** to help with finding all the new *BiskitDefs* using the **Filter Biskit Group...** button
 - c) In Properties **Add New**
 - i. Change **Name** to **Name**
 - d) Change **Name Property** on the *BiskitDef* to point to the new **Name** property

The screenshot displays the configuration interface for a new *BiskitDef* named 'ProjectInfo'. The interface is divided into two main panels: 'Meta-properties' on the left and 'Properties' on the right.

Meta-properties Panel:

- Type:** ProjectInfo (Callout: a) Change the type to ProjectInfo)
- Parent:** None
- Group:** ☐ None ☒ MyCompanies (Callout: b) Changed the Group to MyCompanies)
- Primary Key:** 0
- Version Number:** 0
- Name Property:** id (Callout: d) Change the Name Property to Name)
- Sort Property:** Use default sorting
- Format:** ☒ None
- Abstract:** ☐
- Enumerable:** ☒
- Hierarchy Property:** No suitable properties found
- Label Lower:** project info
- Label Upper:** Project Info
- Labels Lower:** project infos
- Labels Upper:** Project Infos
- Null Value Label (Read Only):** Use default
- Null Value Label (Read Write):** Use default
- Tooltip:**
- Visibility:** Everybody
- Storage Mechanism:** Dynamic
- Shareable Table:** Shareable with sub-types
- Allows Deletion While Referenced:** Does not allow deletion
- Table Name:** project_info
- Primary Key Column Name:** id

Properties Panel:

- Buttons:** Add New, Cut, Copy, Paste, Delete
- Table:**

Name	Type
(drag) Name	String

 (Callout: c) Add new property)
- Primary Key:** 0
- Type:** String
- Name:** Name (Callout: i) Change the Name to Name)
- Description:** ☒ None
- Label:** Name
- Tooltip:** ☒ None
- Sort Order:** 0
- Live:** false
- Min:** ☒ None
- Max:** ☒ None
- Attributes:**
 - ☒ Visible
 - ☒ Editable
 - ☒ Persistent
 - ☒ Null Allowed
 - ☐ Required
 - ☒ Visible In Biskit Detail
 - ☒ Visible In Biskit List
 - ☐ Visible In Collection Editor
- Rows:** ☒ None
- Columns:** ☒ None
- Group:** ☒ None
- Storage Mechanism:** Dynamic
- Formulaic:** ☐
- Column Name:** name
- Unique:** ☐
- String Property Type:** Single Line
- String Type:** Unconstrained
- Default Value:** Null

6. Press the **Save** button.
7. If there are no errors the database will need to be updated:
 - a) Press **Update DB Schema** to implement changes in the DB, and then run the script to apply the changes
 - b) Press **Validate Biskits** button to check the database *Biskits*.
 - c) Press **Reload Database Configuration**, to load the new database into **Calpendo**
 - d) Refresh the browser

6.18.6.1.7 Creating A Set Of Biskits (Attachments)

An example of creating a Set of [Biskits](#)⁶⁵² for this example we will create a Set of Attachments, to store a number of documents.

1. Go to **Admin->Bakery**.
2. Click the + next to **Biskit Def** to open the **Biskit Tree**
3. Select any [Biskit](#)⁶⁵².
4. Press the **Create** button to create a new *Biskit* and enter edit mode. First we need to create the *BiskitDef* to hold the attachments.
5. Update the [meta-properties](#)⁶⁵⁴ of the *Biskit*.
 - a) Change **Type** to be the new *BiskitDef* name, in this case **ProjectAttachment**.
 - b) Change **Group** to help find all the new *BiskitDefs* using the **Filter Biskit Group...** button
 - c) In **Properties Add New**
 - i. Change **Name** to **attachment**
 - ii. Change **Type** to **Biskit**
 - iii. Change **Biskit Def** to **Attachment**

Type	ProjectAttachment
Parent	None
Group	<input type="checkbox"/> None <input type="checkbox"/> Nursery
Primary Key	0
Version Number	0
Name Property	FileName
Sort Property	Use default sorting
Version Property	No version number
Created Property	None
Updated Property	None
Format	<input checked="" type="checkbox"/> None
Abstract	<input type="checkbox"/>
Enumerable	<input checked="" type="checkbox"/>
Hierarchy Property	No suitable properties found
Label Lower	project attachment
Label Upper	Project Attachment
Labels Lower	project attachments
Labels Upper	Project Attachments
Null Value Label (Read Only)	Use default
Null Value Label (Read Write)	Use default
Tooltip	
Visibility	Everybody
Storage Mechanism	Dynamic
Shareable Table	Shareable with sub-types
Allows Deletion While Referenced	Does not allow deletion
Table Name	project_attachment
Primary Key Column Name	id

Properties
Add New Cut Copy Paste Delete

Name	Type
(drag) FileName	Biskit

Primary Key	0
Type	Biskit
Name	FileName
Description	<input checked="" type="checkbox"/> None
Label	File Name
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	0
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic
Formulaic	<input type="checkbox"/>
Column Name	file_name
Unique	<input type="checkbox"/>
Automated Property Type	None
Biskit Def	Attachment
Component	<input type="checkbox"/>
Biskit Property Type	To One
Reference Deletion Option	No Action
Default Value	Null

ii) Change Type to Biskit

i) Change Name to FileName

iii) Change Biskit Def to Attachment

© 2019-2021 Exprodo Software

6. Create a new *property*.
 - a) Change **Type** to **Biskit**.
 - b) Change **Name** to **project**
 - c) Change **Biskit Def** to **Project**
 - d) Change **Biskit Property Type** to **Many To One**.
 - e) Make sure **Required** is ticked (not shown in this diagram)

Properties

Add New Cut Copy Paste Delete

Name	Type
[drag] FileName	Biskit
[drag] Project	Biskit

New Project Property

a) Change Type to Biskit

b) Change Name to Project

Primary Key	0
Type	Biskit
Name	Project
Description	<input checked="" type="checkbox"/> None
Label	Project
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	1
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic
Formulaic	<input type="checkbox"/>
Column Name	project
Unique	<input type="checkbox"/>
Automated Property Type	None
Biskit Def	Project
Component	<input type="checkbox"/>
Biskit Property Type	Many to One
Inverse Property	None
Default Value	Null

c) Change Biskit Def to Project

d) Change Biskit Property Type to Many To One

7. Press the **Save** button. There will be an error:

Failed to Save BiskitDef

BiskitDef is invalid and has not been saved

OK Show Log Save Despite Errors

Attributes

Log

Level	Message
Error	ProjectAttachment.Project is manyToOne Project but no inversePropertyName set

OK

Check the error by pressing **Show Log**. If the error is complaining about a no inversePropertyName being set press the **OK** button, and then the **Save Despite Errors** button.

9. In **Properties Add New**.
 - a) Change **Type** to **Set**.
 - b) Change **Name** to **attachments**
 - c) Change **Biskit Def** to **Project Attachment**
 - d) Change **Biskit Property Type** to **One To Many**
 - e) Change **Reference Deletion Option** to **Set Null**
 - f) Change **Inverse Property** to **project**

Add New

Cut

Copy

Paste

Delete

Name	Type
projectCode	String
type	Biskit
status	JavaEnum
owner	Biskit
description	String
ownerPhone	String
resourceSettings	Set
start	Date
finish	Date
users	Set
principalInvestigator	String
ethics	String
fundingAgency	String
accountNumber	String
version	Int
created	Datetime
updated	Datetime
serviceSettings	Set
ethicsAttachment	Biskit
fileAttachment	Biskit
self	Biskit
attachments	Set

Primary Key

0

Type	Set
Name	attachments
Description	<input checked="" type="checkbox"/> None
Label	Attachment
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	22
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic
Formulaic	<input type="checkbox"/>
Column Name	attachments
Unique	<input type="checkbox"/>
Automated Property Type	None
Collection Subtype	Biskit
Sorted	<input type="checkbox"/>
Biskit Def	Project Attachment
Component	<input type="checkbox"/>
Biskit Property Type	One to Many
Reference Deletion Option	Set null
Inverse Property	Project

New attachments Property

c) Change BiskitDef to Project Attachment

d) Change Biskit Property Type to One To Many

e) Change Reference Delete Option to Set Null

f) Change Inverse Property to Project

a) Change Type to Set

b) Change name to attachments

10. Press the **Save** button, and this should save without an error.

11. Select the **Project Attachment BiskitDef**, press the **Edit** button to get back into edit mode.

12. Select the **Project** property.
a) Change **Inverse Property** to **attachments**.

The screenshot shows the 'Properties' dialog box for a BiskitDef. The 'Name' field is 'Project' and the 'Type' is 'Biskit'. The 'Primary Key' is '1396'. The 'Type' is 'Biskit'. The 'Name' is 'Project'. The 'Description' is 'None'. The 'Label' is 'Project'. The 'Tooltip' is 'None'. The 'Sort Order' is '1'. The 'Live' property is 'false'. The 'Min' and 'Max' values are 'None'. The 'Attributes' section includes checkboxes for 'Visible', 'Editable', 'Persistent', 'Null Allowed', 'Required', 'Visible In Biskit Detail', 'Visible In Biskit List', and 'Visible In Collection Editor'. The 'Rows' and 'Columns' are 'None'. The 'Group' is 'None'. The 'Storage Mechanism' is 'Dynamic'. The 'Formulaic' checkbox is unchecked. The 'Column Name' is 'project'. The 'Unique' checkbox is unchecked. The 'Automated Property Type' is 'None'. The 'Biskit Def' is 'Project'. The 'Component' checkbox is unchecked. The 'Biskit Property Type' is 'Many to One'. The 'Inverse Property' is 'Attachments'. The 'Default Value' is 'Null'. A callout box points to the 'Inverse Property' field with the text 'a) Change Inverse Property to Attachments'.

13. Press the **Save** button.

14. If there are no errors the database will need to be updated:

- Press **Update DB Schema** to implement changes in the DB, and then run the script to apply the changes
- Press **Validate Biskits** button to check the database *Biskits*.
- Press **Reload Database Configuration**, to load the new database into **Calpendo**
- Refresh the browser

6.18.6.1.8 Creating A Hierarchy Of BiskitDef

An example of creating a hierarchy of [BiskitDef](#)⁶⁵² similar to **Location**. Where a parent can have many children of the same type of *BiskitDef* and a child will have only one parent. We will create one called **Organisation**.

1. Go to **Admin->Bakery**.
1. Click the + next to **Biskit Def** to open the **Biskit Tree**
2. Select any [Biskit](#)⁶⁵².
3. Press the **Create** button to create a new *Biskit* and enter edit mode.
5. Update the [meta-properties](#)⁶⁵⁴ of the *Biskit*.
 - a) Change **Type** to be the new *BiskitDef* name, in this case **Organisation**.
 - b) Change **Filter Group** to help find all the new *BiskitDefs* using the **Filter Biskit Group...** button
 - c) In **Properties Add New**
 - i. Change **Name** to **name**
 - d) Change **Name Property** on the *BiskitDef* to point to the new **name** [property](#)⁶⁵⁵

Type	Organisation
Parent	None
Group	<input type="checkbox"/> None <input type="checkbox"/> MyCompany
Primary Key	0
Version Number	0
Name Property	id
Sort Property	Use default sorting
Format	<input checked="" type="checkbox"/> None
Abstract	<input type="checkbox"/>
Enumerable	<input checked="" type="checkbox"/>
Hierarchy Property	No suitable properties found
Label Lower	organisation
Label Upper	Organisation
Labels Lower	organisations
Labels Upper	Organisations
Null Value Label (Read Only)	Use default
Null Value Label (Read Write)	Use default
Tooltip	
Visibility	Everybody
Storage Mechanism	Dynamic
Shareable Table	Shareable with sub-types
Allows Deletion While Referenced	Does not allow deletion
Table Name	organisation
Primary Key Column Name	id

Properties	
Name	Type
(drag) Name	String
Primary Key	0
Type	String
Name	Name
Description	<input checked="" type="checkbox"/> None
Label	Name
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	0
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic
Formulaic	<input type="checkbox"/>
Column Name	name
Unique	<input type="checkbox"/>
String Property Type	Single Line
String Type	Unconstrained
Default Value	Null

6. Create a new *property*.
 - a) Change **Type** to **Biskit**.
 - b) Change **Name** to **parent**
 - c) Change **BiskitDef** to *BiskitDef*
(**Organisation** is not know yet because we have not saved so use this as a holder)
 - d) Leave **Biskit Property Type** as **To One**.

Properties

Add New Cut Copy Paste Delete

Name	Type
[drag] Name	String
[drag] Parent	Biskit

New Parent Property

a) Change Type to Biskit

b) Change Name to Parent

c) Change Biskit Def to Biskit Def (temporary)

d) Do not change Biskit Property Type

Primary Key	<input type="checkbox"/>
Type	Biskit
Name	Parent
Description	<input checked="" type="checkbox"/> None
Label	Parent
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	1
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible in Biskit Detail <input checked="" type="checkbox"/> Visible in Biskit List <input type="checkbox"/> Visible in Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic
Formulaic	<input type="checkbox"/>
Column Name	parent
Unique	<input type="checkbox"/>
Biskit Def	Biskit Def
Component	<input type="checkbox"/>
Biskit Property Type	To One
Reference Deletion Option	No Action
Default Value	Null

7. Create a new *property*
 - a) Change **Type** to **Set**.
 - b) Change **Name** to **children**
 - c) Change **BiskitDef** to **BiskitDef**
(**Organisation** is not known yet because we have not saved so use this as a holder)
 - d) Leave **Biskit Property Type** as **One To Many**

8. Press the **Save** button. There will be an error:

Check the error by pressing **Show Log**. If the error is complaining about a *property* not having the correct *BiskitDef* press the **OK** button, and then the **Save Despite Errors** button.

9. Press the **Edit** button to get back into edit mode.

10. Select the **Parent** property
- Change **BisketDef** to **Organisation**.
 - Change **Biskit Property Type** to **Many To One**
 - Change **Inverse Property** to **children**

Properties

Add New Cut Copy Paste Delete

Name	Type
[drag] Name	String
[drag] Parent	Biskit
[drag] Children	Set

Primary Key: 1247

Type: Biskit

Name: Parent

Description: ☒ None

Label: Parent

Tooltip: ☒ None

Sort Order: 1

Live: false

Min: ☒ None

Max: ☒ None

Attributes: ☒ Visible, ☒ Editable, ☒ Persistent, ☒ Null Allowed, ☐ Required, ☒ Visible In Biskit Detail, ☒ Visible In Biskit List, ☐ Visible In Collection Editor

Rows: ☒ None

Columns: ☒ None

Group: ☒ None

Storage Mechanism: Dynamic

Formulaic: ☐

Column Name: parent

Unique: ☐

Biskit Def: Organisation

Component: ☐

Biskit Property Type: Many to One

Inverse Property: Children

Default Value: Null

a) Change Bisket Def to Organisation

b) Change Biskit Property Type to Many To One

c) Change Inverse Property to Children

11. Press the **Save** button, there will be an error, **Save Despite Error**. This is needed so that when the **children** property is set up it finds the completed **parent** property to set up **Inverse Property**
12. Press the **Edit** button to get back into edit mode.

13. Select the **Children** property.
- Change **BasketDef** to **Organisation**.
 - Change **Biskit Property Type** to **One To Many**
 - Change **Reference Deletion Option** to **Set Null**
 - Change **Inverse Property** to **parent**

Add New		Cut	Copy	Paste	Delete
Name	Type				
name	String				
parent	Biskit				
children	Set				
Primary Key		1584			
Type		Set ▼			
Name		children			
Description		<input checked="" type="checkbox"/> None			
Label		Children			
Tooltip		<input checked="" type="checkbox"/> None			
Sort Order		2			
Live		false			
Min		<input checked="" type="checkbox"/> None			
Max		<input checked="" type="checkbox"/> None			
Attributes		<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor			
Rows		<input checked="" type="checkbox"/> None			
Columns		<input checked="" type="checkbox"/> None			
Group		<input checked="" type="checkbox"/> None			
Storage Mechanism		Dynamic			
Formulaic		<input type="checkbox"/>			
Column Name		children			
Unique		<input type="checkbox"/>			
Automated Property Type		None ▼			
Collection Subtype		Biskit ▼			
Sorted		<input type="checkbox"/>			
Biskit Def		Organisation			
Component		<input type="checkbox"/>			
Biskit Property Type		One to Many			
Reference Deletion Option		Set null ▼			
Inverse Property		Parent ▼			

a) Change Biskit Def to Organisation

b) Change Biskit Property Type to One To Many

c) Change Reference Deletion Option to Set null

d) Change Inverse Property to Parent

14. Change the **Hierarchy Property meta-property** of the **Organisation Biskit** to be **children**.
15. Press the **Save** button.
16. If there are no errors the database will need to be updated:
- Press **Update DB Schema** to implement changes in the DB, and then run the script to apply the changes
 - Press **Validate Biskits** button to check the database *Biskits*.
 - Press **Reload Database Configuration**, to load the new database into **Calpendo**
 - Refresh the browser

6.18.6.1.9 Creating A Master-Slave Biskit Relationship

An example of creating a **Master-Slave Biskit**⁶⁵² relationship. This type of relationship is needed if there are going to be a large number of pieces of data to be stored in effectively one table. MySQL has a limit of about 1000 columns to a table, so if more are needed, *Biskits* that are completely linked together will need to be created. Where if a record is created in one an equivalent record will be created in the other and they will be referenced through the same key.

The **Master-Slave** that will be created will have only one *property*⁶⁵⁵ on the **Master** (**Name** of type **String**), once this has been created then all the *properties* needed, can be added to each of the *Biskits*. A **Master** could have many **Slave Biskits** as required, a *property* needs to be set up on the **Master** to point to each **Slave**. Each **Slave** just needs one *property* to point to the **Master**. The **Primary Key Column Name** in the **Slave** MUST be different to that in the **Master**. If there are multiple **Slaves** then the **Primary Key Column Name** in each **Slave** MUST be different.

1. Go to **Admin->Bakery**.
2. Click the + next to **Biskit Def** to open the **Biskit Tree**
3. Select any *Biskit*.
4. Press the **Create** button to create a new *Biskit* and enter edit mode.

5. Update the [meta-properties](#)⁶⁵⁴ of the *Biskit*.
 - a) Change **Type** to be the new [BiskitDef](#)⁶⁵² name, in this case **PatientInfo**.
 - b) Change **Group** to make it easy to find all the new *BiskitDefs* using the **Filter Biskit Group...** button
 - c) In **Properties Add New**
 - i. Change **Name** to **Name**
 - d) Change **Name Property** on the *BiskitDef* to point to the new **Name** property
6. Create a new *property*.
 - a) Change **Type** to **Biskit**.
 - b) Change **Name** to **Slave**.
 - c) Change **Biskit Def** to **Biskit Def** (The other *BiskitDef* is not know yet because we have not created it so use this as a holder).

The screenshot shows the Calpendo Properties editor with two panels. The left panel is for a property named 'PatientInfo' and the right panel is for a property named 'Slave'. Both panels have various configuration options, and several annotations are present:

- Left Panel (PatientInfo):**
 - Annotation 'a) Change Type to PatientInfo' points to the 'Type' field.
 - Annotation 'b) Change Group to MyCompany' points to the 'Group' field.
 - Annotation 'd) Change Name Property to new Name' points to the 'Name Property' field.
- Right Panel (Slave):**
 - Annotation 'a) Change type to Biskit' points to the 'Type' field.
 - Annotation 'b) Change name to Slave' points to the 'Name' field.
 - Annotation 'c) Change Biskit Def to Biskit Def' points to the 'Biskit Def' field.

Annotations 'c) New Name property' and 'New Slave Property' point to the 'Name' and 'Slave' fields in the 'Properties Add New' dialog box.

7. Press the **Save** button.

8. Press the **Create** button to create a new *Biskit* and enter edit mode.
9. Update the *meta-properties* of the *Biskit*.
 - a) Change **Type** to be the new *BiskitDef* name, in this case **PatientInfo2**
 - b) Change **Group** to make it easy to find all the new *BiskitDefs* using the **Filter Biskit Group...** button
 - c) Change **Primary Key Column Name** to id1 (must be different from the **Master BiskitDef Primary Key Column Name**)
10. Create a new *property*.
 - a) Change **Type** to **Biskit**.
 - b) Change **Name** to **Parent**
 - c) Change **BiskitDef** to **Patient Info**
 - d) Change **Biskit Property Type** to **Slave To Master**

The screenshot displays two configuration windows side-by-side. The left window is for a 'PatientInfo2' property, and the right window is for a 'Biskit' property. Annotations with callout boxes point to specific fields in both windows.

Left Window (PatientInfo2):

- Type:** PatientInfo2 (Annotation: a) Change Type to PatientInfo2)
- Parent:** None
- Group:** MyCompany (Annotation: b) Change Group to MyCompany)
- Primary Key:** 165
- Version Number:** 3
- Name Property:** id (Annotation: c) Change iPrimary Key Column Name to id2)
- Sort Property:** Use default sorting
- Format:** None
- Abstract:** ☐
- Enumerable:** ☒
- Hierarchy Property:** No suitable properties found
- Label Lower:** patient info2
- Label Upper:** Patient Info 2
- Labels Lower:** patient info2s
- Labels Upper:** Patient Info 2s
- Null Value Label (Read Only):** Use default
- Null Value Label (Read Write):** Use default
- Tooltip:**
- Visibility:** Everybody
- Storage Mechanism:** Dynamic
- Shareable Table:** Shareable with sub-types
- Allows Deletion While Referenced:** Does not allow deletion
- Table Name:** patient_info2
- Primary Key Column Name:**

Right Window (Biskit):

- Primary Key:** 1243
- Type:** Biskit (Annotation: a) Change Type to Biskit)
- Name:** Parent (Annotation: b) Change Name to Parent)
- Description:** None
- Label:** Parent
- Tooltip:** None
- Sort Order:** 0
- Live:** false
- Min:** None
- Max:** None
- Attributes:**
 - ☒ Visible
 - ☒ Editable
 - ☒ Persistent
 - ☒ Null Allowed
 - ☒ Required
 - ☒ Visible in Biskit Detail
 - ☒ Visible in Biskit List
 - ☐ Visible in Collection Editor
- Rows:** None
- Columns:** None
- Group:** None
- Storage Mechanism:** Dynamic
- Formulaic:** ☐
- Unique:** ☒
- Biskit Def:** Patient Info (Annotation: c) Change Biskit Def to Patient Info)
- Component:** ☐
- Biskit Property Type:** Slave To Master (Annotation: d) Change Biskit Property Type to Slave To Master)
- Reference Deletion Option:** Cascade
- Inverse Property:** No suitable properties found

11. Press the **Save** Button. There will be an error, **Save Despite Errors**.

12. Select the **PatientInfo Biskit** and press the **Edit** button to get back into edit mode.

13. Select the **Slave property**.

- a) Change **Bisket Def** to **PatientInfo2**.
- b) Change **Biskit Property Type** to **Master To Slave**
- c) Change **Inverse Property** to **Parent**

Type	PatientInfo
Parent	None
Group	<input type="checkbox"/> None MyCompany
Primary Key	164
Version Number	2
Name Property	id
Sort Property	Use default sorting
Format	<input checked="" type="checkbox"/> None
Abstract	<input type="checkbox"/>
Enumerable	<input checked="" type="checkbox"/>
Hierarchy Property	No suitable properties found
Label Lower	patient info
Label Upper	Patient Info
Labels Lower	patient infos
Labels Upper	Patient Infos
Null Value Label (Read Only)	Use default
Null Value Label (Read Write)	Use default
Tooltip	
Visibility	Everybody
Storage Mechanism	Dynamic
Shareable Table	Shareable with sub-types
Allows Deletion While Referenced	Does not allow deletion
Table Name	patient_info
Primary Key Column Name	id

Properties	
Name	Type
[drag] Name	String
[drag] Slave	Biskit

Properties	
Add New Cut Copy Paste Delete	
Primary Key	1241
Type	Biskit
Name	Slave
Description	<input checked="" type="checkbox"/> None
Label	Slave
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	1
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic
Formulaic	<input type="checkbox"/>
Unique	<input checked="" type="checkbox"/>
Biskit Def	Patient Info 2
Component	<input type="checkbox"/>
Biskit Property Type	Master To Slave
Reference Deletion Option	Cascade
Inverse Property	Parent

a) Change Bisket Def to Patient Info 2

b) Change Biskit Property Type to Master To Slave

c) Change Inverse Property to Parent

14. Press the **Save** button. There will be an error, **Save Despite Errors**.

15. Select the **PatientInfo2 Biskit** and press the **Edit** button to get back into edit mode.
 16. Select the **Parent property**.
 a) Change **Inverse Property** to **Slave**

Type	PatientInfo2
Parent	None
Group	<input type="checkbox"/> None <input checked="" type="checkbox"/> MyCompany
Primary Key	165
Version Number	5
Name Property	id
Sort Property	Use default sorting
Format	<input checked="" type="checkbox"/> None
Abstract	<input type="checkbox"/>
Enumerable	<input checked="" type="checkbox"/>
Hierarchy Property	No suitable properties found
Label Lower	patient info2
Label Upper	Patient Info 2
Labels Lower	patient info2s
Labels Upper	Patient Info 2s
Null Value Label (Read Only)	Use default
Null Value Label (Read Write)	Use default
Tooltip	
Visibility	Everybody
Storage Mechanism	Dynamic
Shareable Table	Shareable with sub-types
Allows Deletion While Referenced	Does not allow deletion
Table Name	patient_info2
Primary Key Column Name	id2

Properties					
Add New Cut Copy Paste Delete					
	<table border="1"> <tr> <th>Name</th> <th>Type</th> </tr> <tr> <td>Parent</td> <td>Biskit</td> </tr> </table>	Name	Type	Parent	Biskit
Name	Type				
Parent	Biskit				
Primary Key	1243				
Type	Biskit				
Name	Parent				
Description	<input checked="" type="checkbox"/> None				
Label	Parent				
Tooltip	<input checked="" type="checkbox"/> None				
Sort Order	0				
Live	false				
Min	<input checked="" type="checkbox"/> None				
Max	<input checked="" type="checkbox"/> None				
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input checked="" type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor				
Rows	<input checked="" type="checkbox"/> None				
Columns	<input checked="" type="checkbox"/> None				
Group	<input checked="" type="checkbox"/> None				
Storage Mechanism	Dynamic				
Formulaic	<input type="checkbox"/>				
Unique	<input checked="" type="checkbox"/>				
Biskit Def	Patient Info				
Component	<input type="checkbox"/>				
Biskit Property Type	Slave To Master				
Reference Deletion Option	Cascade				
Inverse Property	Slave				

a) Change
Inverse Property
to Slave

17. Press the **Save** button.
 18. If there are no errors the database will need to be updated:
 a) Press **Update DB Schema** to implement changes in the DB, and then run the script to apply the changes
 b) Press **Validate Biskits** button to check the database *Biskits*
 c) Press **Reload Database Configuration**, to load the new database into **Calpendo**
 d) Refresh the browser

6.18.6.1.10 Creating An Inheriting BiskitDef (Booking)

An example of creating a [*Biskit*](#)⁶⁵² that inherits from another *Biskit*. In this case **Resource**. Please note that when editing the new *Biskit* the user cannot see the [*properties*](#)⁶⁵⁵ that it inherits from the parent *Biskit*.

1. Go to **Admin->Bakery**.
1. Click the + next to **Biskit Def** to open the **Biskit Tree**
2. Select any *Biskit*.
3. Press the **Create** button to create a new *Biskit* and enter edit mode.
5. Update the [meta-properties](#)⁶⁵⁴ of the *Biskit*.
 - a) Change **Type** to be the new [BiskitDef](#)⁶⁵² name, in this case **ExtendedResource**.
 - b) Change **Parent** to **Resource**
 - c) Change **Group** to make it easy to find all the new *BiskitDefs* using the **Filter Biskit Group...** button
 - d) Change **Name Property** on the *BiskitDef* to have the same value as the **Name Property** of the parent
 - e) Decide whether new *BiskitDef* will share a table with its parent by changing **Share Super Type Table** (it is best if they do share a table).
6. In Properties **Add New**
 - a) Change **Name** to PatientInfo
 - b) Change **Type** to Int

Type	Scanners	a) Change Type to Scanners
Parent	Booking	b) Change Parent to Booking
Group	<input type="checkbox"/> None MyCompany	c) Change Group to MyCompany
Primary Key	0	
Version Number	0	
Name Property	dateRange	d) Change Name to be same as parent - dateRange
Sort Property	Use default sorting	
Version Property	No version number	
Created Property	None	
Updated Property	None	
Format	<input checked="" type="checkbox"/> None	
Abstract	<input type="checkbox"/>	
Enumerable	<input checked="" type="checkbox"/>	
Hierarchy Property	No suitable properties found	
Label Lower	scanners	
Label Upper	Scanners	
Labels Lower	scannersss	
Labels Upper	Scannersss	
Null Value Label (Read Only)	Use default	
Null Value Label (Read Write)	Use default	
Tooltip		
Visibility	Everybody	
Storage Mechanism	Dynamic	
Share Super Type Table	Share database table	e) Decide whether new Biskit Type will share a table
Shareable Table	Shareable with sub-types	
Allows Deletion While Referenced	Does not allow deletion	

Properties

Add New
Cut
Copy
Paste
Delete

Name	Type
[drag] PatientNo	Int

6. Create new properties for the Biskit

Primary Key	0
Type	Int
Name	PatientNo
Description	<input checked="" type="checkbox"/> None
Label	Patient No
Tooltip	<input checked="" type="checkbox"/> None
Sort Order	27
Live	false
Min	<input checked="" type="checkbox"/> None
Max	<input checked="" type="checkbox"/> None
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input checked="" type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input type="checkbox"/> Visible In Collection Editor
Rows	<input checked="" type="checkbox"/> None
Columns	<input checked="" type="checkbox"/> None
Group	<input checked="" type="checkbox"/> None
Storage Mechanism	Dynamic
Formulaic	<input type="checkbox"/>
Column Name	patient_no
Unique	<input type="checkbox"/>
Automated Property Type	None
Integer Type	Unconstrained
Default Value	Null

7. Press the **Save** button.
8. If there are no errors the database will need to be updated:
 - a) Press **Update DB Schema** to implement changes in the DB, and then run the script to apply the changes
 - b) Press **Validate Biskits** button to check the database *Biskits*
 - c) Press **Reload Database Configuration**, to load the new database into **Calpendo**
 - d) Refresh the browser

6.19 Layout Editor

The **Layout Editor** allows the user to define how the *properties*⁶⁵⁵ of a *Biskit*⁶⁵² will be displayed. This is required because *Biskits* could have a large number of *properties*, numbering in the hundreds, and in order for users to find those *properties* to update them, it is necessary to organise them.

The **Layout Editor** allows the user to organise the *properties* of a *Biskit* into multiple **Property Groups**, **Property Groups** may themselves contain child **Property Groups**.

Property Groups at the same level may be organised in one of five ways, **Vertical Tabs**, **Vertical Tabs (Sorted by Name)**, **Horizontal Tabs**, **Captions** and **Flow**. **Property Groups** may have a heading and may also be hidden.

The *properties* contained within a **Property Group** may also be organised in one of four ways, **Simple Table**, **Multi Column Table**, **Radio Button Table** and **Custom HTML**.

For each table, row, column or cell the standard CSS can be changed allowing the user to set such things as font, background colour, border size, type and colour and many other attributes.

Finally for any **Property Group**, **Notes** can be set up to be created and then displayed in a **Notes Waterfall**.

The **Layout Editor** is split into four sections in two panes.

1. The left hand pane shows a list of [BiskitDefs](#)⁶⁵² that have had layouts created for them, when a *BiskitDef* is expanded all the layouts defined for that *Biskit* will be shown. Any in red are currently not enabled.
2. The right pane is split into three areas:
 - a. Top is the layout menu and the header information for the current layout. (**Top Pane**)
 - b. Bottom left shows the organisation of the **Property Groups** and *properties* for this layout in the form of a tree. (**Organisation Pane**)
 - c. Bottom right displays information depending on what is selected in the organisation tree in the bottom left pane. (**Information Pane**)
 - i. If a **Property Group** is selected then the header information for that **Property Group** is viewed along with any *properties* assigned to that **Property Group**.
 - ii. If a **Properties** branch is selected then the *properties* are shown in a list
 - iii. If a single *property* is selected then information about that *property* will be shown.

The screenshot displays the Calpendo Layout Editor interface. The left pane shows a list of BiskitDefs: DatabaseUpgrade, TestSubject, Visit1, Visit1 Layout (selected), Visit2, Visit3, and Visit4. A callout points to the 'Visit1 Layout' entry, stating 'Left Pane showing Biskit Defs and their layouts'. The top pane, titled 'Layout Menu', contains a table with layout header information for 'Visit 1'.

Displayed Type	Visit 1
Enabled	true
Unassigned Properties Display Location	Show In Main Table
Version	10
Created	8 Apr 2015 11:09
Updated	8 Apr 2015 11:09

A callout points to this table, stating 'Layout header information'. The organisation pane shows a tree view of the layout structure. A callout points to the tree, stating 'Tree showing organisation of property groups and properties'. The information pane displays a table of properties for the selected 'PSQI Totals' property group.

Name	PSQI Totals
Hidden	false
Multiline Text In Tab	In main table with other properties
Property Grouping Method	Captions
Heading	
Property Table Type	Simple Table

A callout points to the 'Property Table' tab, stating 'Displayed information of the item selected in the tree to the left, in this case a property group'. Below the table is a list of properties: PSQIDURAT, PSQIDISTB, PSQILATEN, PSQIDAYDYS, PSQISLPQUAL, PSQIMEDS, PSQIHSE, and PSQI.

Left Pane

The Left Pane shows the **Layout Tree** (see image above). Showing for each *BiskitDef* the layouts that are available. It also has a tool bar which allows the user to open and close all the branches to the tree.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Top Pane With Layout Toolbar

The **Top Pane** shows the header information for a layout. To create a new layout press the **Create** button and then choose the *BiskitDef* the layout is for.

Property	Description
Displayed Type	The <i>BiskitDef</i> the layout is for.
Enabled	Whether this layout is currently to be used. Disabled layouts are shown in the layout tree in the left pane in red.
Unassigned Properties Display Location	Specifies how all unassigned <i>properties</i> will be displayed. Either Show in Main Table where the <i>properties</i> are shown in a table above the tabs section or Shown in Property Group where the <i>properties</i> are shown as the first tab in the tabs section with a name for the tab assigned by the user.
Version	Current version number of the layout.
Created	When the layout was created.
Updated	Last time the layout was updated.

The **Top Pane** also has the **Layout** toolbar which enables the user to edit, create and delete layouts as well as being able to preview the layout already defined.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

Layout Toolbar Item	Description
Preview	This will display a preview of the current layout as defined. When in edit mode this preview will be up to date and you do not need to save first. The preview will display initially in read-write mode, but the user can change the mode using a toggle at the top of the view to see what the display would look like in read-only mode. Some information is only displayed in read-only mode. The information in the <i>properties</i> in the preview is constructed in a random manner to give the user an idea of what the layout would look like with data and is not read from the database.

By pressing the **Preview** button a view of the current layout showing the different areas will appear. Use this preview to walk around the layout checking how it works.

The **Flow** grouping method, means the name of a "tab" is ignored and is intended for when a caption or a box is not required around a set of properties. It provides a cleaner look when there are multiple sections displayed together. This can be combined with multiple columns to allow display of many areas at once.

Multiple Columns

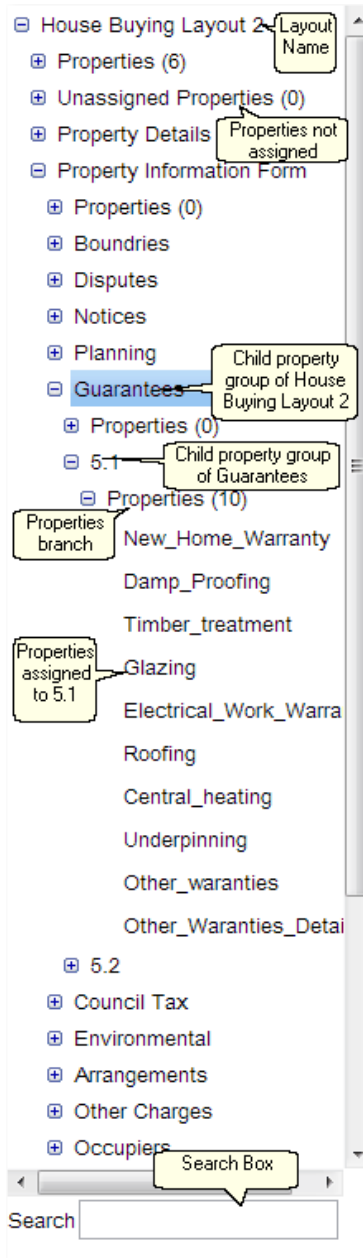
If either **Captions** or **Flow** are chosen then the user also has the choice of whether to display the property groups in one or multiple columns. If multiple columns are chosen the user can define the minimum width and height to display each property group in.

Name	Project Layout
Hidden	false
Multiline Text In Tab	In main table with other properties
Property Grouping Method	Captions
Display in columns	Display children in multiple columns
Minimum Width	300
Minimum Height	300
Heading	

Shown below are the results of displaying in multiple columns.

Project Calpendo ID Number 2 Title test Running Title Describe your project here Short Project Description Owner admin (admin) Phone Number Project (SB) Number Status Approved Type Proposed Start Of MRI Study Proposed Completion of Study Co-Researchers Students No project contacts found	MRI MRI Scanner Required Please select Will this study involve acquiring Phantom Data Only Proposed Number of Subjects to be Scanned Proposed number of Scans Per Subject Total Number Scans Proposed MRI Time Per Subject Please select	Other Ancillary Equipment Other Planned Investigations Other Relevant Information	PI Principal Investigator PI Phone PI Fax PI Email PI Address 1 PI Address 2 PI City PI State PI Post Code
	Billing Grant Code Billing Address 1 Billing Address 2 Billing City Billing State Billing Post Code	Submission Ethics Application Attached Ethics Application Ethics Approval Letter Attached Ethics Approval Submit For Authorisation	For Office Use Only Settings Project Users Protocol Verification Checklist Admin Team Users This project has no users associated with it

Organisation Pane



This pane shows how the *properties* of a *Biskit* have been organised into **Property Groups** and child **Property Groups** as a tree structure.

The layout is defined at the top of the pane, the layout will have a **Properties** and **Unassigned Properties** branch.

Property Groups are then created. When a **Property Group** is created it will automatically have a **Properties** branch which will initially be empty.

Property Groups can be reordered by dragging and dropping the group elsewhere within its parent.

To move a **Property Group** from its current parent to another parent they can be dragged and dropped or by using cut and paste. Cut the **Property Group**, select the new parent, paste the **Property Group**. Then reorder as above.

Hidden **Property Groups** are displayed in red.

The user in edit mode can move *properties* from one **Properties** branch to another by dragging and dropping. All the *properties* in a **Properties** branch can be moved by dragging the branch to a different **Properties** branch. Also *properties* can be dragged from a property list in the **Information Pane** and dropped into a **Properties** branch in the **Organisation Pane**.

Properties can be reordered within a **Property Group** by dragging and dropping the *properties*.

The search box can be used to find a *property*. Once the user starts typing a name into the search box, a list of possible *properties* will appear, as the user continues typing the possible list will reduce, until only one remains. The **Properties** branch containing that *property* will be expanded in the tree and the *property* will be highlighted.

If what has been typed does not match any possible *property* the search box will be outlined in red and the search text will also go red.

Information Pane

The **Information Pane** will have information about the item that is currently selected in the **Organisation Pane**. There are three types of items that can be selected

1. A **Property Group**.
2. A **Properties** branch
3. A *property*

Copy	
Name	PSQI Totals
Hidden	false
Multiline Text In Tab	In main table with other properties
Property Grouping Method	Captions
Heading	
Property Table Type	Simple Table
Property Table Notes	
PSQIDURAT	psqiDURAT
PSQIDISTB	psqiDISTB
PSQILATEN	psqiLATEN
PSQIDAYDYS	psqiDAYDYS
PSQISLPQUAL	psqiSLPQUAL
PSQIMEDS	psqiMEDS
PSQIHSE	psqiHSE
PSQI	psqi

If a **Property Group** is selected then information about that group is displayed. Any *properties* associated with the **Property Group** are listed under the **Property Table** tab showing their **Name** and **Label**.

The **Notes** tab allows the user to define whether this **Property Group** contains a **Notes** field and/or a .

The **Copy** button allows the user put a copy of this **Property Group** into the buffer for later pasting.

Information	Description
Name	Name of the Property Group
Hidden	Whether the Property Group should be hidden from view.
Multiline Text in Tab	Whether any Multiline text will be displayed in a Separate Tab (original functionality) or In main table with other properties (new functionality).
Property Grouping Method	How the child Property Groups of this Property Group should be grouped/displayed. Vertical Tabs, Vertical Tabs (Sorted by Name), Horizontal Tabs, Captions and Flow . To see examples of these layout types look at the Preview ⁶⁰⁶ image above.
Heading	Any heading to be displayed with this Property Group . These can have HTML style and class attributes defined such as <code><i> </i></code> .
Property Table Type	How Properties belonging to this Property Group will be displayed. Simple Table, Multi Column Table, Radio Button Table and Custom HTML .

Name	Type
New_Home_Warranty	Int
Damp_Proofing	Int
Timber_treatment	Int
Glazing	Int
Electrical_Work_Warranty	Int
Roofing	Int
Central_heating	Int
Underpinning	Int
Other_waranties	Boolean
Other_Waranties_Details	String

Property Def	New Home Warranty
--------------	-------------------

If a **Properties** branch to the tree is selected then the *properties* grouped here will be displayed as a list, showing their **Name** and **Type**.

If an individual *property* is selected then the *properties* **Label** will be viewed. This is so the user can see what will be written for this *property* in the layout.

Property Def	Preop Hernia
Show Full Detail	Show In Property Group Preop Hernia

If the *property* selected is of type *Biskit* then there will also be an option of how to display the *Biskit* information. Either as a **Show Simple**, (as a single element in a table) or **Show in Property Group** with a name assigned by the user (treating the *Biskit* as the first **Property Group**. If the *Biskit* has a layout defined it will be used, otherwise all the *properties* will be displayed as a simple table.)

Primary Key	298
Type	Biskit
Name	subjectStatus
Description	
Label	Subject Status
Tooltip	
Sort Order	0
Live	true
Min	
Max	
Attributes	<input checked="" type="checkbox"/> Visible <input checked="" type="checkbox"/> Editable <input checked="" type="checkbox"/> Persistent <input type="checkbox"/> Null Allowed <input type="checkbox"/> Required <input checked="" type="checkbox"/> Visible In Biskit Detail <input checked="" type="checkbox"/> Visible In Biskit List <input checked="" type="checkbox"/> Visible In Collection Editor
Rows	
Columns	
Group	
Storage Mechanism	Static
Automated Property Type	
Biskit Def	Subject Status
Component	<input type="checkbox"/>
Biskit Property Type	To One
Reference Deletion Option	No Action
Default Value	

If a *property* is selected from the unassigned *properties* branch the information from the [Bakery](#)⁵³⁷ about the *property* is displayed for information. This information can **NOT** be edited here even if the layout is in edit mode. To edit the details of a *property* the user will need to go to the [Bakery](#)⁵³⁷.

Information Pane Edit Mode

On entering edit mode what is viewed in the **Information Pane** will vary depending on what is selected in the **Organisation Pane**.

If a **Property Group** is selected then the header information for the **Property Group** will be shown in an editable form. See the [table](#)⁶⁰⁸ in the previous section for a description of the information displayed.

Property Grouping Method	Property Table Type
PSQIDURAT	psqiDURAT
PSQIDISTB	psqiDISTB
PSQILATEN	psqiLATEN
PSQIDAYDYS	psqiDAYDYS
PSQISLPQUAL	psqiSLPQUAL
PSQIMEDS	psqiMEDS
PSQIHSE	psqiHSE
PSQI	psqi

There will also be an edit toolbar.

Property Group Toolbar Item	Description
Add New	Adds a new Property Group at the same level as the currently selected Property Group .
Add New Child	Adds a new Property Group as a child of the currently selected Property Group .
Cut	Cuts the currently selected Property Group , removing it from its current place in the tree and puts it in the edit buffer overwriting any current contents.
Copy	Takes a copy of the currently selected Property Group and puts it in the edit buffer overwriting any current contents.
Paste	Pastes the current buffer contents as a child of the currently selected Property Group . When pasting a Property Group care needs to be taken if the Property Group has any <i>properties</i> associated with it. <i>Properties</i> can only appear once in a layout and the editor will remove <i>properties</i> that clash with the pasted <i>properties</i> .
Delete	Deletes the selected Property Group . Any <i>properties</i> associated with that Property Group will be moved to the Unassigned Properties branch.
Show Cell CSS Editors	This will display in each cell of the <i>properties</i> a little icon allowing the user to start up a CSS editor for that cell. For more on the CSS editor read the section on CSS Editors ⁶⁰³ .

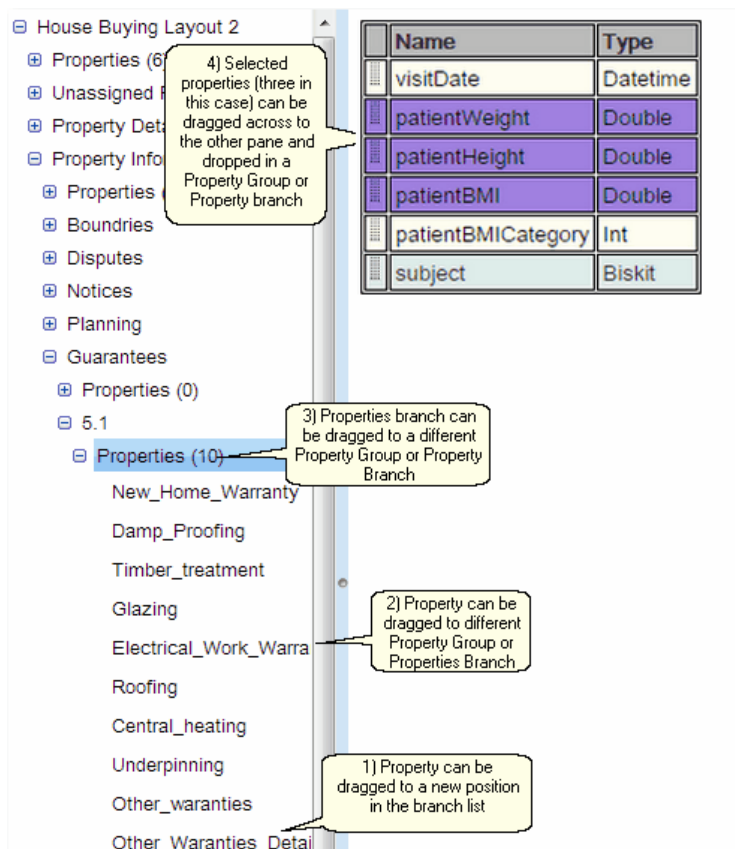
Moving Properties

If a **Properties** branch is selected then a list of *properties* will be shown in the **Information Pane**.

There are four ways to move *properties* within a combination of the **Organisation Pane** and the **Information Pane**

1. An individual *property* can be selected in the **Organisation Pane** and dragged to a new position in the list.
2. An individual *property* can be selected in the **Organisation Pane** and dragged and dropped into a different **Properties** branch.
3. A **Properties** branch can be selected in the **Organisation Pane** and dragged and dropped into a different **Properties** branch, all the *properties* in the dragged branch will then transfer to the new branch.
4. In the **Information Pane** using shift and ctrl, a number of *properties* can be selected from the viewed list and dragged across to a **Properties** branch in the **Organisation Pane**.

In all the above cases dropping the *properties* on a **Property Group** name will place the *properties* in the **Properties** branch for that **Property Group**.



Property Table Types

There are four different ways to display a list of *properties*.

1) Simple Table

Name	5.1
Hidden	false
Property Grouping Method	Captions
Heading	Does the property benefit from any of the following guarantees or
Property Table Type	Simple Table

New Home Warranty	New_Home_Warranty
Damp Proofing	Damp_Proofing
Timber Treatment	Timber_treatment
Glazing	Glazing
Electrical Work	Electrical_Work_Warranty
Roofing	Roofing
Central Heating	Central_heating
Underpinning	Underpinning
Other Warranties	Other_warranties
If yes give details	Other_Warranties_Details

Information in a **Simple Table** will be displayed with the *property* label and the *property* value.

2) Radio Button Table

Name	Kitchen
Hidden	false
Property Grouping Method	Captions
Heading	
Property Table Type	Radio Button Table

	Not Applicable	Yes	No
Hob	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Extractor hood	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fitted oven and grill	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fitted microwave	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Tumble dryer	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fridge-freezer	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Freezer	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Free standing oven	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Dishwasher	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Washing machine	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Information in a **Radio Button Table** is displayed showing the property labels and then a choice of values. This type of table is limited to the following

- a. All *properties* must be of the same type.
- b. They must be one of the following types:
 - i. *Biskit* valued.
 - ii. [Mapped Integer](#)⁶⁵³.
 - iii. [Mapped String](#)⁶⁵⁴.


3) Multi-Column Table

Row Labels	<input type="radio"/> Default row labels <input checked="" type="radio"/> Custom row labels <input type="radio"/> No row labels	
Number of Property Columns	2	
Show Column Headings	<input checked="" type="checkbox"/>	
Column Label Alignment	Centre	
Row Label Alignment	Right	

Custom (editable) Row Labels		Editable Column Headings
No symptoms. I am attending for sterilisation	B1Sterilisation	B1SterilisationSymptomAge
Pelvic Pain	B1PelvicPain	B1PelvicPainSymptomAge
Painful Periods	B1PainfulPeriods	B1PainfulPeriodsSymptomAge
Infertility	B1Infertility	B1InfertilitySymptomAge
Ovarian Cyst	B1OvarianCyst	B1OvarianCystSymptomAge
Painful Intercourse	B1PainfulIntercourse	B1PainfulIntercourseSymptomAge
Pain On Opening Bowels	B1PainOnOpeningBowels	B1PainOnOpeningBowelsSymptomAge
Bowel Upset eg: Constipation, Diarrhoea	B1BowelUpset	B1BowelUpsetSymptomAge
Pain On Passing Urine	B1PainOnPassingUrine	B1PainOnPassingUrineSymptomAge
Other Urinary Problems	B1OtherUrinaryProblems	B1OtherUrinaryProblemsSymptomAge
Other	B1Other	B1OtherSymptomAge

Add in new empty cell after the current cell

Information in a **Multi-Column Table** will be displayed with the *property* label and the *property* value. If multiple columns are being displayed only the labels of the *properties* required for the number of rows being used will be displayed.

Multi-Column Table	Description
Row Labels	Allows the user to define how row labels are displayed. The choices are 1) Default row labels 2) Custom row labels 3) No row labels.
Number of Property Columns	The number of columns used to display the <i>properties</i> . The first <i>property</i> in the list will have its value displayed at the top of the first column, the next <i>property</i> value at the top of the next column and so on until the columns are filled up. The next available <i>property</i> value will then be displayed in the second row of the first column, <i>property</i> values will then fill in the second row before moving onto the next row and so on until all <i>property</i> values have been displayed
Show Column Headings	Displays column headings into which the user may place their own labels.
Column Label Alignment	Defines whether the column labels are left, centre or right justified.
Row Label Alignment	Defines whether the row labels are left, centre or right justified.
	Adds a new empty cell after the current cell. Useful for filling out tables which have a variable number of rows.

4) Custom HTML

Enable building of a table structure using HTML.

Name	Suffered From
Hidden	false <input type="button" value="v"/>
Property Grouping Method	Horizontal Tabs <input type="button" value="v"/>
Heading	Has the property ever suffered from any of the following
Property Table Type	Custom HTML <input type="button" value="v"/>

```

<table>
<tr>
<th><div id="Suffered_Flooding_1-label"/></th>
<th><div id="Suffered_Rising_Damp-label"/></th>
</tr>
<tr>
<td><div id="Suffered_Flooding_1-value"/></td>
<td><div id="Suffered_Rising_Damp-value"/></td>
</tr>
</table>

```

Property attribute

Property name

Other Display Options

Boolean *properties* have additional options, click on the *property* to get

Property Def	Suffered Flooding
Boolean Type	LISTBOX
Boolean True Text	true
Boolean False Text	false
Boolean Null Text	null

The display type can be either a list box or a check box and the user can define the text to be used for **True**, **False** and **Null**.

Biskit *properties* have additional options, click on the *property* to get

Property Def	Status
Show Full Detail	Show Simple
	Show Simple
	Show In Property Group

The details of the *Biskit* can be displayed either as a simple table of all the *properties* (**Show Simple**) or in a separate **Property Group (Show in Property Group)** using the current layout type, **Horizontal**, **Vertical** or **Caption**. Also if a layout has been defined for the *Biskit* then the layout definition will be used for the *properties* of the *Biskit*.

Set *properties* have additional options, click on the *property* to get

Property Def	Samples
One To Many Type	Multi-Edit ▼
One To Many Import Header	name,type

The **Set** can be edited in the **Regular** way, one at a time, or **Multi-Edit** (or **Multi-Edit** (on click)) can be set up, so multiple children of the can be created and edited together. If the **One To Many Import Header** is setup with the properties of the **Multi Edit** then there will be the option to **Download** and **Upload**.

The **Multi-Edit** of sets has a number of options:

The screenshot shows a web interface titled "Testing Multi". At the top, there is a toolbar with the following buttons: Delete (minus icon), Edit (pencil icon), Add (plus icon), Add Many... (plus icon with a grid), Copy (two overlapping squares), Paste (square with plus), Download (downward arrow), and Upload (upward arrow). Below the toolbar is a table with three columns: "Name", "Dates", and "A Number". The table contains six rows. The first row is highlighted with a blue border, indicating it is selected. Each row has a small square checkbox in the first column, which is checked for the first row. The table is set against a light gray background.

Delete - Select a number of rows using the the checked boxes and the minus will remove them all.

Edit - Edits the ticked children in a popup. (Checked editing)

Add - Adds a new row to the bottom.

Add Many - provides a pop-up to decide how many rows need to be added.

Copy - Copy the selected cell(s). Use shift to select multiple cells.

Paste - Paste the copied cells.

Download - Download the table as a CSV file.

Upload- Use a csv file to load information into the table.

Notes

The user can add areas to the layout which enable the storing of notes. In order to do this a *BiskitDef* needs to be created to hold the note information. This *BiskitDef* must have at least two *properties*. Optionally the user can also define a *property* to hold the **Category** of note to be stored.

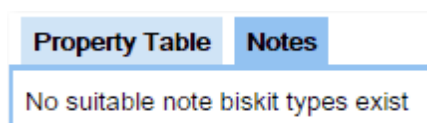
Property Type	Description
String	This <i>property</i> will hold the Note information. If the StringPropertyType is HTML then the note will support Rich Text Format .
Biskit	This property will define the parent <i>BiskitDef</i> that the Note is for. This <i>property</i> must be an Automated Property Type , set to Create and Update , the <i>Biskit Def</i> must be set to the parent <i>Biskit Type</i> and the Biskit Property Type must be set to To One .
Biskit	This property will define the Category BiskitDef that the Note is associated with. If used the Category is stored with the note and is used to determine which Waterfall the note will be displayed in.

Other *properties* may be added to the *BiskitDef* such as automated properties to hold the created and updated time, the creator, the updater and the version number as required.

There may be more than one **Note BiskitDef** if required. A single **Note BiskitDef** can be used to hold notes created in multiple places in the layout, or the user may define a different **Note BiskitDef** for each place notes are going to be added, or a combination of both methodologies.

There is also an addon which makes it very simple to add notes to *Projects*. It creates all the required *Biskits* etc and initializes notes, so all the administrator needs to do is enable dynamic notes.

Once an appropriate BiskitDef has been created the Notes tab will change from



to



use the drop down to

enable **Notes**.

Once enabled the user will see the following in the **Notes** tab.

Note Biskit Type: This needs to be set to the *Biskit Type* that will hold the note information.

Owner Property: This needs to be set to the *property* that points to the owning *Biskit Type*. This is the *Biskit Type* that the notes can be attached to in the Layout Editor.

Note Property: This needs to be set to the *property* that will hold the note information.

Add Note Button Label: This defines the string to be put on the button that will add the note to the database.

Add Note Heading: This defines the string that will be used as a heading to the add note area.

This information is used to determine how the add note area is defined. Below you can see an example of an add note area.

So far the area to write and add notes has been defined. Next the user needs to define an area to display all the notes that have been written to the DB, this is called a **Notes Waterfall**.

Once the **Waterfall** is enabled the user can define what heading will appear above the **Waterfall** in the layout.

Finally notes can be stored under a number of **Category**'s. In order to set up the **Category**'s a [hierarchical BiskitDef](#)⁵⁹² needs to be defined, this is where a parent can have many children of the same type of *BiskitDef* and a child will have only one parent. The minimum properties would be:

Property Type	Description
String	This <i>property</i> will hold the Category name.
Set	This property will point to the children's <i>BiskitDef</i> as a Set. These will be of the same <i>Biskit Type</i> as the parent.
Biskit	This property will point to the parents <i>BiskitDef</i> . This will be of the same <i>Biskit Type</i> as the child.

Category Property	Category
Category	

Please select a Note Category

- Person
- Communications and events
- Facilities
- Finance
- Human Resources
- Research

Once the **Category** is enabled the user will need to set the **Category property**. This is the property on the **Note BiskitDef** that points to the **Category BiskitDef**.

Then the user will need to define which of the **Category's** the current **Notes** section will save **Notes** as. Any **Waterfall** enabled in this **Notes** section, will only display **Notes** of this **Category**. If no **Category** is enabled then the **Waterfall** section will display all **Notes** no matter what their **Category**. An **Add Note** can define a **Category** to create a **Note** under in one place in the **Layout** and a **Notes Waterfall** can be enabled with that **Category** defined somewhere else in the **Layout** if required.

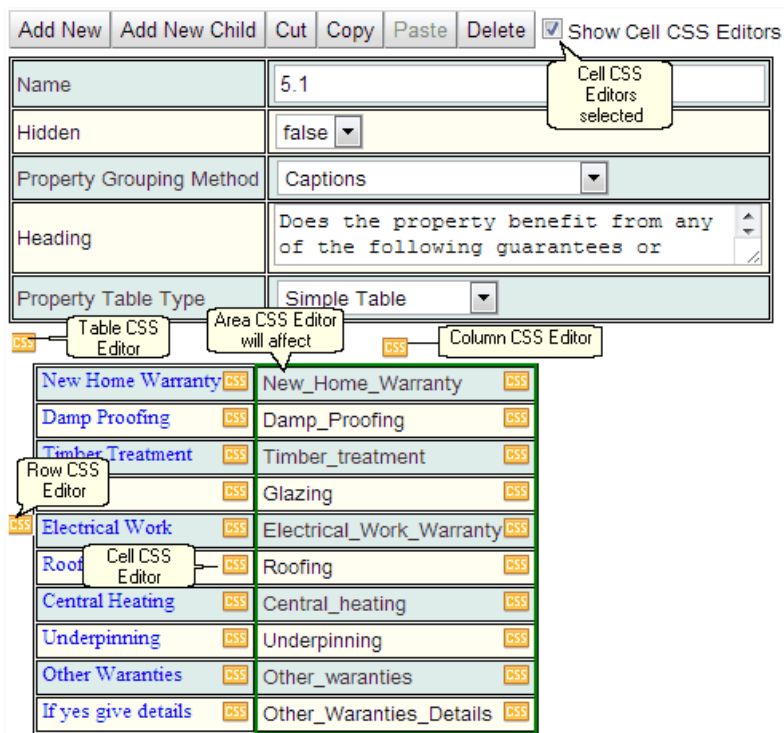
[Permissions](#) ⁶⁵⁴ may be assigned to **Category's** to stop/allow viewing/editing of notes. A *Permission* assigned to a **Category** in a hierarchy will be inherited by all its children.

CSS Rule Editors

The user can define their own CSS information for each Table, Row, Column or Cell using the **CSS Editors**. These editors are displayed as small icons with CSS written on them. In order to bring up the icons:

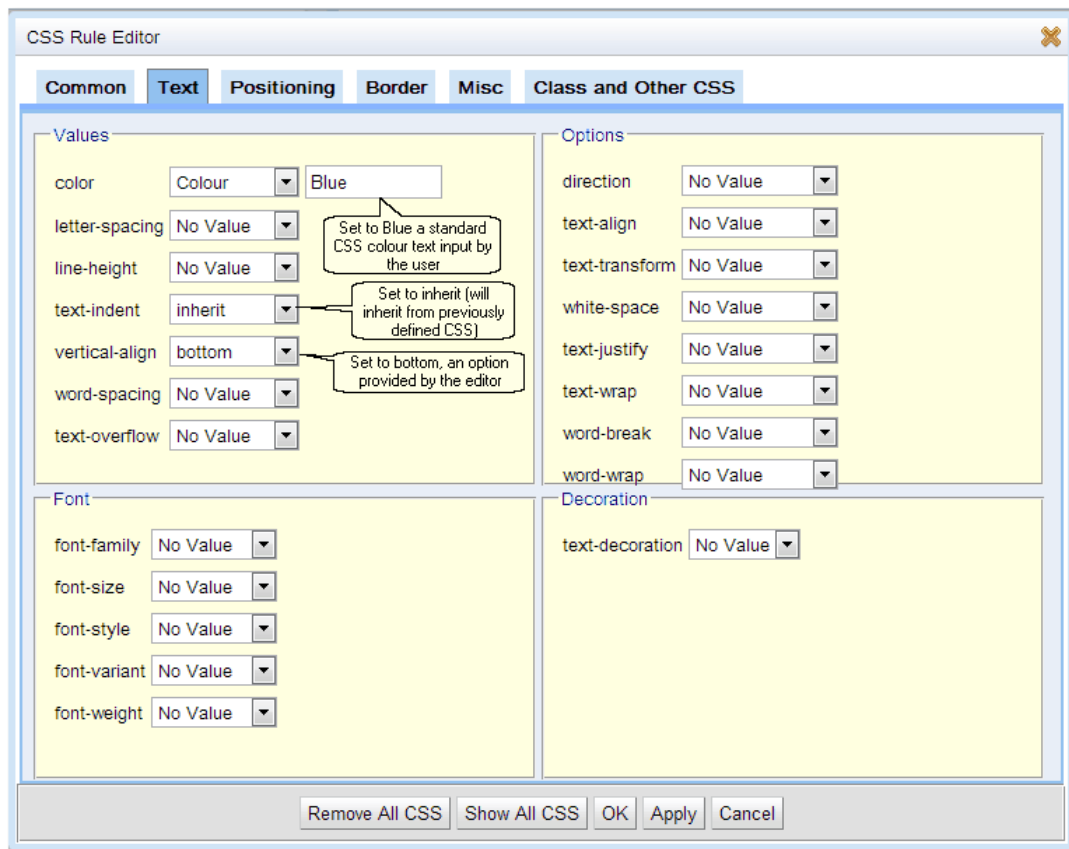
1. For a row, move the cursor to just to the left of the start of the row.
2. For a column, move the cursor to be just above the column.
3. For a table, move the cursor to be just outside the top left hand corner of the table.
4. For a cell, check the **Show Cell CSS Editors** box in the **Property Group Toolbar**.

When selecting a **CSS Rule Editor** the area to be affected will have a green border.



In the example above the first column has had the text colour set to Blue, the font-family set to Times New Roman and the font-size set to small.

When the **CSS Rule Editor** is opened up:



The possible CSS values are organised into different tabs to help with finding the required CSS value. When the drop down button is pressed for a value the user will have a choice of options. Some will require the user to input the appropriate CSS string (see Colour above). The user can also specify inherit which means the CSS will be inherited from previously defined CSS by the user, this could be in [Global Preferences](#)⁶¹⁰ or elsewhere in **Calpendo** where CSS can be defined.

For a complete description of the standard toolbar buttons read the [Toolbar Button Standard Definition](#)¹⁰⁴ chapter.

CSS Editor Toolbar Item	Description
Remove All CSS	Removes all CSS that the user has applied using the editor to this particular object.
Show All CSS	Shows in a pop up window the CSS that has been defined for this particular object.

For more information and help on using CSS and HTML a site we recommend is <http://www.w3schools.com/>

6.20 Processes

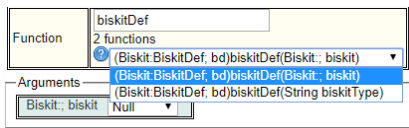
This provides for multi-step editing of [Biskits](#)⁶⁵² where each step is a page with **Next** and **Previous** buttons. The procedure can also have other buttons (**Help**, **Reset**, **Exit**) . There can be defined a standard flow from one **Step** to another, or use [Workflows](#)⁶⁵⁶ to control which **Step** follows on, depending on what was done in the previous **Step**.

Overview of how to set up a **Process**.

1. In the [Bakery](#)⁶³⁷ create a [BiskitDef](#)⁶⁵² and populate with the required [Properties](#)⁶⁵⁵.
2. Create a layout(s) in the [Layout Editor](#)⁶⁰³.
3. Create a **Process Definition** to define how to handle your *BiskitDef*.
 - a. Create a number of **Steps**, within each define which layout is going to be used and additional information such as which buttons will be shown, and who can edit this **Step**.
4. Create any [Workflows](#)³³⁴ required to manage moving between **Steps** that is not handled by the **Process Definition**.

Creating the BiskitDef

Create the *BiskitDef* in the [Bakery](#)⁵³⁷, make sure it is using its own table, populate with the required properties. There can be a *property* of Type **Biskit**, Name **currentStep**, and Biskit Def of **Step**, **Biskit** Property Type **To One**. If such a *property* is defined then whenever a **Process** is re edited the editing will start at the step that was last used. If no such *property* exists re editing the **Process** will always start at the first step.



Creating the Layouts

This can be:

1. A single layout defining all the tabs to be required,
2. Multiple layouts one for each **Step**, with the properties not required on a **Step** put in a **Property Group** that is then defined as hidden.
3. Either of the previous two with some additional layouts defining special combinations of *properties* that would be required in addition to the defining layout.

Each **Step** has access to two layouts, which will be combined to be a single layout for that **Step**. Any *properties* found in both layouts will be displayed once using the information from the primary layout.

Displayed Type	Process Test
Enabled	true
Unassigned Properties Display Location	Show In Main Table
Version	2
Created	19 Oct 2016 11:28
Updated	19 Oct 2016 11:28

Process Test Layout

Properties (0)

Unassigned Properties (0)

Step 1

Step 1 Information

Properties (3)

Step 2

Properties (3)

Tab 3

Properties (1)

Tab for hidden properties

Copy

Name	Step 1
Hidden	false
Multiline Text In Tab	Separate Tab
Property Grouping Method	Horizontal Tabs
Heading	
Property Table Type	Simple Table

Property Table

Notes

Test 1	test1
Test 2	test2
Test 3	test3

Creating the Process Definition

Using a [Search](#)¹¹⁸ page create **Process Definition** *Biskit* for the Process.

Name	
Enabled	true ▼
Process BiskitDef	Please select...
Initial Step	No default step ▼
Run New Process	Cannot create new process without initial step defined

Give the **Process Definition** a name and select the **Process BiskitDef**. The **Initial Step** will be set up once the **Steps** have been created. Once the **Initial Step** is set up then the **Run New Process** will hold a link to create a new **Process Biskit** of type **Process BiskitDef**.

Add the steps required using the **Add Step** button.

Add Step	Cut	Copy	Paste	Delete
Name	New			
Enabled	true ▼			
Writeability	Writeable ▼			
Target Path	The process is the target			
Layout	No layout selected			
Secondary Layout	No secondary layout			
Sort Order	1			

Information	Description
Name	The name of the step.
Enabled	Whether this step is enabled for use.
Writeability	Writable or ReadOnly . If ReadOnly then nothing is editable.
Target Path	Normally refers to the current <i>Biskit</i> , but can be used to point to <i>Biskits</i> that are referenced from this one.
Layout	The Layout or sub part thereof to be used for this Step .
Secondary Layout	The Layout or sub part thereof to be used as a subsidiary layout for this Step .
Sort Order	The order the Steps will be listed in the left pane.

For each **Step** there are a number tabs which provide additional functionality for that **Step**.

Buttons

Buttons	Navigation Bars	Conditions	Help Text	Editors	Non-Editors	People & Changes
Button	Label	Enabled	Visible	Default Follow-on Step		
Previous Button	Previous	<input type="checkbox"/>	<input type="checkbox"/>	No default step ▼		
Next Button	Next	<input type="checkbox"/>	<input type="checkbox"/>	No default step ▼		
Reset Button	Reset	<input type="checkbox"/>	<input type="checkbox"/>	No default step ▼		
Finish Button	Finish	<input type="checkbox"/>	<input type="checkbox"/>	No default step ▼		
Help Button	Help	<input type="checkbox"/>	<input type="checkbox"/>	No default step ▼		

Define for each button, what label will be used, whether the button can be seen, and whether it can be pressed or is greyed out. Finally specify which **Step** will be the default if the button is pressed.

Navigation Bars

Please select where the buttons should be displayed

Show Top	true ▼
Show Bottom	false ▼
Show Left	false ▼
Show Right	false ▼

Define where the buttons will be displayed.

Conditions

There are no conditions [Add condition](#) +

The [Conditions](#)⁶⁵³ define whether or not you can go to the **Next Step**. If the Conditions are considered not to be met then the user will not be allowed to move to the **Next Step**.

HelpText

Rich text editor toolbar and formatting options.

Define the text that will appear when the **Help** button is pressed.

Editors

Specify the users that are allowed to edit this step, or leave empty to allow anybody to do so
There are no entries here
[Add](#) +

Use the add button to specify the users that are allowed to edit this step, or leave empty to allow anybody to do so.

Non-Editors

Define the text that will be shown to anybody who tries to view this **Step** that is not deemed to be a valid editor.

People & Changes

This view shows who created or updated this **Step** and when.

Creating the Workflows

Each Workflow will need to start with a [Process Workflow Event](#)³⁴⁰.

Trigger Type	Process Workflow Event
Name	New
Enabled	true ▼
Children to Run	All children ▼
Sort Order	0
Last Ran	

Target Type	Process Test
Finish Events	false ▼
Next Events	false ▼
Previous Events	false ▼
Reset Events	false ▼
Help Events	false ▼

- Set the target type to your *BiskitDef* (so this event is triggered only by changes to *BiskitDef*)
- Choose whether you want to be triggered by clicking next, previous etc.

Workflow creation advice:

- If a **Process Event** is defined to handle the next button, and inside that there is a **BiskitUpdate Action** to go from **Step 1** to **Step 2**, and a sibling that goes from **Step 2** to **Step 3**, then on pressing **Next** from **Step 1**, this will jump straight to **Step 3**.
- That's because it will run **Step 1** to **Step 2 Action** first, then look for the next **Action**.
- If the next **Action** then has a condition to check if the current **Step** is 2, it will fire because the current **Step** has just been changed to be 2.
- In order to stop this have one event for each change. That is, one **Process Event** that contains everything to move from **Step 1** to **Step 2**. Then configure each **Event** to ignore events caused by "**This Workflow**".

6.21 Backing Up The Database

To take a backup of the database, use the **Database Dump** page, which is by default on the menu as **Admin-->Database Dump**.

Before asking for the backup, there are the following options:

Option	Description	
Include structure	If ticked, the backup will contain SQL statements that will create the database tables.	
Include data	If ticked, the backup will contain SQL statements containing the data from the database.	
Add drop if exists	If ticked, the before each database table is created, a statement is added to drop that table if it already exists.	
Compress	If ticked, the backup will be compressed before being downloaded.	
Include all tables	This drop down allows the choice of one of the following options:	
	Option	Description
	Include all Tables	Dumps all tables
	Include all tables except history and statistics	Dumps all tables except history and statistics
	Select Tables	Provides a simple choice of tables to be dumped.
	Use Custom per-table settings	See below for complete description

When all the settings are as required, just press the **Dump database** button to initiate the backup. When the backup is complete, the browser should download the backup file.

Calpendo limits how often a database backup may be taken, by setting **Minimum time between database backups** on the General tab of the [Global Preferences](#)⁵⁰⁹.

Use Custom Per-Table Settings

If the **Use Custom per-table settings** option is chosen, then **Calpendo** will get a list of all the tables in the database and allow the way each table is backed up to be customised.

Table Name	Use global settings	Include Structure	Add drop if exists	Include Data	Select By Column	Min Column Value	Max Column Value
audit_log	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
audit_log_properties	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
biskit_defs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
booking_properties	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
booking_types	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
bookings	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
bookmark_elements	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
bookmark_properties	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
bookmarks	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			

For each table, there are the following options:

Option	Description
Use global settings	If ticked, then the other options for this table will be ignored.
Include structure	Allows the global setting of Include structure for this table to be overridden.
Add drop if exists	Allows the global setting of Add drop if exists for this table.
Include data	Used to override the global setting of Include data for this table to be overridden.
Select by column	If partial data is to be downloaded, then the user can nominate a column of the table and specify minimum and/or maximum values of that column that should be included in the download.
Min Column Value	If the user wants to download partial data and specifies Select by column , then the Min column value is the minimum value of the selected column that should be included in the download.
Max Column Value	If the user wants to download partial data and specifies Select by column , then the Max column value is the maximum value of the selected column that should be included in the download.

6.22 Creating A Read Only Copy

If **Calpendo** is installed on your own servers, it may be necessary to create a backup instance that runs against a copy of the database. Normally such a copy would want to be configured to be read-only so that people can connect to it to access information, but cannot change anything.

There are three ways to place your **Calpendo** into a read-only mode:

1. Set up the access rights for the database user so that it does not have insert [Permission](#)⁶⁵⁴ to the **exprodo_events** table. A [system event](#)⁶⁵⁶ is created at boot time, and if the insert fails, this is used as a trigger to assume a read-only mode.
2. Set the [read-only mode](#)⁵²⁵ setting in the [Global Preferences](#)⁵⁰⁹ [General](#)⁵²⁵ tab.
3. Change the read-only mode by directly modifying the database. Here is an example of the SQL you would use if you are using MySQL to store the **Calpendo** database:
- 4.

```
update config_properties set boolval='T' where name='readOnlyMode';
```

To manually modify the database to disable read-only mode that is triggered by setting the *global preferences*, run this SQL:

```
update config_properties set boolval='F' where name='readOnlyMode';
```

Requirements For A Read-Only Database

If you create a database and configure **Calpendo's** database user so that it has no insert or update *Permissions*, then **Calpendo** will not work properly. To make it work, you must at least have the following *Permissions* (this is expressed in MySQL code):

```
GRANT SELECT ON `calpendo`.* TO 'calpendo_ro'@'localhost' identified by 'thepassword';
GRANT INSERT, UPDATE ON `calpendo`.`audit_log_properties` TO 'calpendo_ro'@'localhost';
GRANT INSERT, UPDATE ON `calpendo`.`sessions` TO 'calpendo_ro'@'localhost';
GRANT INSERT, UPDATE ON `calpendo`.`user_settings` TO 'calpendo_ro'@'localhost';
GRANT INSERT, UPDATE ON `calpendo`.`audit_log` TO 'calpendo_ro'@'localhost';
GRANT INSERT, UPDATE ON `calpendo`.`user_resource_info` TO 'calpendo_ro'@'localhost';
GRANT INSERT, UPDATE ON `calpendo`.`login_attempts` TO 'calpendo_ro'@'localhost';
```

where **calpendo_ro** is the name of the database user.

Consequences of Putting Calpendo Into Read-Only Mode

In read-only mode, the following apply:

- Reminders emails are disabled. This is because **Calpendo** cannot track which reminders have or have not been sent.
- **Automatic Emails** do not run
- When using the [Bookings Calendar](#)⁶⁵², it does not offer the ability to edit anything
- Normally, when a [repeat](#)⁶⁵⁵ [booking](#)⁶⁵² or [repeat Time Template](#)⁶⁵⁶ passes a due time, a new instance of the *booking* or *Time Template* is created so that **Calpendo** keeps a record of what *bookings* or *Time Templates* were in effect. It means that all past records should be accurate. In read-only mode, no such modification takes place. This means that the record of the past is not the same as it would be for a read-write **Calpendo**.

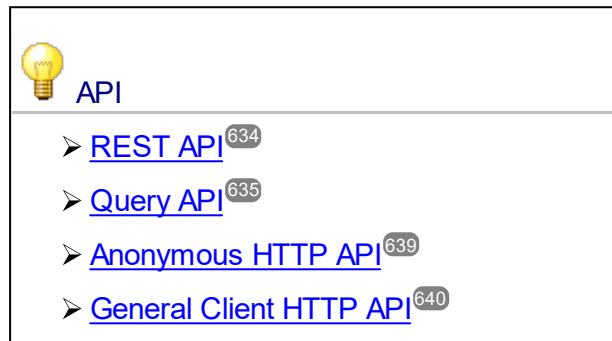
Part



7 API

There are four different ways of providing an external program that makes incoming requests to **Calpendo**. You can choose the one that's most appropriate to the task you need to perform.

We have a [REST API](#)⁶³⁴ that provides RESTful access in JSON, XML or HTML, but currently only in read-only form. There's a read-only [Query API](#)⁶³⁵ that lets perform complex data queries by specifying conditions to filter which data to return as well as the columns you want. A [Workflows](#)⁶³⁴ [Anonymous HTTP event](#)⁶⁴² is triggered whenever an http call is made to a subset of URLs, which enables you to run a workflow in response to a web browser or other program. Note that this is an [anonymous API](#)⁶³⁹ which means there is no logging in for this interface and you will be responsible for providing your own security arrangements. Finally, the [API used by our standard web client](#)⁶⁴⁰ is a JSON API that you can access, although only a small subset of the API is published. Amongst other things, this will allow you to run a workflow *and* go through authentication.



7.1 REST API

The REST API is accessible over HTTP/HTTPS (although HTTP would usually be disabled) and currently provides **read-only** access to the database. It can serve data in JSON, XML and HTML formats. It is available on your own system by appending "/webdav" to your normal login URL. This means it will look like:

`https://yourcalpendo/webdav`

An example can be seen by pointing the web browser at <https://demo.calpendo.com/webdav> and logging in with the username "admin" and password "admin".

After going to the above address, click on items and navigate through your data. Accessing via a normal web browser means you will be looking at the HTML formatted output.

Accessing The API Over wget

In order to automate the API, you can use a utility like wget. Here is an example of requesting JSON-formatted output:

```
wget -O - --header="Accept: application/json" --user=USER --password=PASSWORD http
```

The URL /webdav will show all available services that can be discovered by following the links in the output (see [HATEOAS](#)).

The query API /webdav/q (described in the next section) is not shown by /webdav because the format of its URLs are not discoverable from the UI. Currently, the only service available at the top level is

/webdav/b

which lists all the known [Biskit Types](#)⁶⁵², and provides a URL showing how to list each of them. It is likely that there will be other services available under /webdav in the future (for example, to run reports that have been saved).

Example showing all instances of [Project](#)⁶⁵⁴:

```
wget -O - --header="Accept: application/json" --user=USER --password=PASSWORD http
```

For details of the REST API, you can use its discoverability features to follow links directly from calls to /webdav/.

7.2 Query API

The query API is accessible over HTTP/HTTPS (although HTTP would usually be disabled) and provides **read-only** access to the database using queries where you can specify arbitrarily complicated conditions to select the right data and also choose which data paths should be included in the response. It can serve data in JSON, XML and HTML formats. It is available on your own system by appending "/webdav/q" to your normal login URL. This means it will look like:

<https://yourcalpendo/webdav/q>

Conditional Queries

URLs are of the form:

```
webdav/q/TYPE[?paths=PATHS]
```

or

```
webdav/q/TYPE/EXPR[?paths=PATHS]
```

where

TYPE := a *Biskit Type*

EXPR := BOOLOP/EXPR/.../EXPR/CLOSE | PATH/RELATION/VALUE

BOOLOP:= AND | OR | NAND | NOR

CLOSE := "CLOSE"

PATH := [property path](#)⁶⁵⁵ valid from [Biskit](#)⁶⁵² of type TYPE

RELATIO := EQ | NE | LT | LE | GT | GE

N

| CONTAINS | DOES_NOT_CONTAIN

| STARTS_WITH | DOES_NOT_START_WITH

| ENDS_WITH | DOES_NOT_END_WITH

| INCLUDES_ALL | DOES_NOT_INCLUDE_ALL

| INCLUDES_ANY | DOES_NOT_INCLUDE_ANY

| MEMBER_OF | NOT_MEMBER_OF

| CONTAINS_MEMBER | DOES_NOT_CONTAIN_MEMBER

VALUE := value that is valid format for the [property](#)⁶⁵⁶ referenced by PATH value or the text string `__NULL__` which is used to indicate a value of null.

PATHS := [PATH[,PATH[,PATH[...]]]]

PATH := PROPERTY[.PROPERTY[.PROPERTY[...]]]

PROPER:= the name of a *property*

TY

Note that any CLOSE items at the end of the URL can be omitted.

The relations EQ | NE | LT | LE | GT | GE stand for:

- equal
- not equal
- less than
- less than or equal
- greater than
- greater then or equal

For example:

```
webdav/q/Workflow/updated/GT/20140526T1200
```

returns the contents of all **Workflows** that have been modified since noon on May 26, 2014.

```
webdav/q/Workflow/AND/updated/GT/20140526T1200/name/CONTAINS/Thing
```

returns the contents of all **Workflows** that have been modified since noon on May 26, 2014 and whose name contains the text "Thing".

Also, note that if a query is made to ask for data for a [repeatable](#)⁶⁵⁵ *Biskit*, then those *Biskits* with a *repeat* will be expanded into each instance of the *repeat* provided the query specifies a range within which the results should be returned. This is necessary to avoid expanding an infinite set of results. The server will attempt to identify the minimum value of the start date/time and the maximum value of the finish date/time. If it can work out both of those from the conditions set on the query, then all *Biskits* with a *repeat* will be expanded.

For example, in Calpendo if you search for bookings like this:

```
/webdav/q/Booking/AND/id/eq/88/dateRange.finish/gt/20140501-00:00/dateRange.start/
```

then the bookings will be expanded.

Applicability Of Relations

- **CONTAINS, DOES_NOT_CONTAIN, STARTS_WITH, DOES_NOT_START_WITH, ENDS_WITH** and **DOES_NOT_END_WITH** all apply to String-valued *properties* and apply case insensitive comparisons.
- **INCLUDES_ALL, DOES_NOT_INCLUDE_ALL, INCLUDES_ANY, DOES_NOT_INCLUDE_ANY, GAINED_VALUES, NOT_GAINED_VALUES, LOST_VALUES** and **NOT_LOST_VALUES** only apply to integer *properties* that are defined as a bit set.
- **MEMBER_OF** and **NOT_MEMBER_OF** applies to applies to two sorts of paths:
 1. A path that leads to a group.

For Calpendo, this means it applies to Calpendo.CalpendoUser, Calpendo.Project and Calpendo.Resource, and the value compared against would be UserGroup, Calpendo.ProjectGroup and Calpendo.ResourceGroup respectively.

For other Exprodo applications, this would apply only to ExprodoUser and the value compared against would be a UserGroup.

For example:

```
webdav/q/ExprodoUser/self/MEMBER_OF/UserGroup-7
```

2. A path that leads to a collection, or through a collection of biscuits.

For example:

```
webdav/q/ExprodoUser/self/MEMBER_OF/UserGroup-7.users
webdav/q/Calpendo.CalpendoUser/self/MEMBER_OF/Calpendo.Project-
4.users
webdav/q/Calpendo.CalpendoUser/self/MEMBER_OF/Calpendo.ProjectGroup-
7.projects.owner
webdav/q/Calpendo.CalpendoUser/self/MEMBER_OF/Calpendo.ProjectGroup-
7.projects.users
```

- **CONTAINS_MEMBER** and **DOES_NOT_CONTAIN_MEMBER** apply only to groups. For **Calpendo**, this means it applies to **UserGroup**, **ProjectGroup** and **ResourceGroup** and the value being tested is a **CalpendoUser**, **Project** or **Resource**. For other **Exprodo** applications, it applies only to **UserGroup** and the value being tested is an **ExprodoUser**.

Note that **MEMBER_OF**, **NOT_MEMBER_OF**, **CONTAINS_MEMBER** and **DOES_NOT_CONTAIN_MEMBER** all require that a value that is a *Biskit* is provided. This can be done by specifying a string of the form "biskitType-PK" where biskitType is the *Biskit's Type*, and PK as its primary key.

Time Zones

The dates are formatted like this:

2017-06-12T17:00Z

The "Z" is short for "Zulu" and is the standard indication that a time is in UTC. This is close to UK time, but not the same thing.

If the UK is on summer time, they are at UTC+1. There's an inconsistency - the output is all in UTC, but the dates put into the query are interpreted as local time.

When asking for "all booking entries for Sunday (06/11) starting from midnight to 11:59 pm"

This would be:

```
AND/dateRange.start/ge/20170611-00:00/dateRange.start/lt/20170612-00:00/
```

Refining Searches

Choosing bookings for particular resources such as MRI1 and MRI2, can be done either by the name of the resources, or by their ID.

To do this by name (pseudo code format):

start >= 20170611-00:00 and start < 20170612-00:00 and (resource.name = MRI1 or resource.name = MRI2)

and by resource id (pseudo code format):

start >= 20170611-00:00 and start < 20170612-00:00 and (resource.id = 1 or resource.id = 2)

if MRI1 and MRI2 have IDs 1 and 2.

Therefore, the full query line for these are:

```
webdav/q/Calpendo.Booking/AND/dateRange.start/GE/2017061100:00/dateRange.start/LT/
```

or

```
webdav/q/Calpendo.Booking/AND/dateRange.start/GE/2017061100:00/dateRange.start/LT/
```

Query Data Through Collections

- In a single API call, the user can ask for all the users that own a project in a particular Project Group, or all the users associated with projects in a particular Project Group.
- For example, to find all the users that belong to projects in project group number 7:

```
webdav/q/Calpendo.CalpendoUser/self/MEMBER_OF/Calpendo.ProjectGroup-7.projects.use
```

- and to find all the owners of projects in project group 9:

```
webdav/q/Calpendo.CalpendoUser/self/MEMBER_OF/Calpendo.ProjectGroup-9.projects.own
```

7.3 Anonymous HTTP API

The anonymous HTTP API is a mechanism that allows you to create a web server whose content is generated by workflows. Examples of the kind of things you might use this for are:

- Show a web-form to users and store data they enter into the database
- Respond to a card reader that notifies Calpendo whenever a card has been swiped so that you can enable or disable an instrument or open a door.
- Have a workflow send out emails that includes hyperlinks for a user to click on, so the user can then enter or view additional information.
- Perform pretty much any kind of operation you want.

For more details, see the workflows description of the [Anonymous HTTP Workflow Event](#)³⁴².

7.4 General Client HTTP API

The standard web client communicates with the server using JSON-formatted messages. You can use standard tools within the browser to see the content of the requests and responses. However, we do not publish a specification for most of it because it is subject to change. We do plan on publishing an example of using this API to run a workflow.

Part



8 Calpendo Installation Guide

If your **Calpendo** is to be hosted on **calpendo.com**, then you will not need to read these instructions as it will be installed for you. You only need to read this installation guide if you are going to install **Calpendo** on your own server.

Hardware and Software Requirements

Operating system

- Windows
- MacOS
- Linux (preferred)

Server Software

- Java 8 or Java 11
- MySQL 5.6 or later or MariaDB 10.1.2 or later
- Tomcat 7 minimum
- Apache HTTP Server (tested on 2.2 and 2.4, although any version should work)

Server Hardware

- RAM 4GB minimum, 8GB recommended
- Hard Disk 10GB minimum, but requirements depend on usage, particularly if you configure the system to store file uploads.
- Recent & fast CPU recommended
- Both 32 bit and 64 bit supported
- You can use a virtualised server if required

Client Software

- Chrome - any version, but recent versions preferred
- FireFox - any version, but recent versions preferred
- Safari - any version, but recent versions preferred
- Microsoft Edge
- Internet Explorer IE 9 or later, but IE11 is much better than IE9. (IE8 mostly works but is extremely slow)

Requirements

You should have received a file archive that contains the following files:

```
INSTALL/apache/http-redirect-www.yoursite.com.conf
INSTALL/apache/https-www.yoursite.com.conf
INSTALL/apache/http-testonly-www.yoursite.com.conf
INSTALL/apache_with_mod_jk/http-redirect-www.yoursite.com.conf
INSTALL/apache_with_mod_jk/mod_jk.conf
INSTALL/apache_with_mod_jk/https-www.yoursite.com.conf
INSTALL/apache_with_mod_jk/http-testonly-www.yoursite.com.conf
INSTALL/apache_with_mod_jk/README.txt
INSTALL/apache_with_mod_jk/workers.properties
INSTALL/apache_with_webauth/http-redirect-www.yoursite.com.conf
INSTALL/apache_with_webauth/https-www.yoursite.com.conf
INSTALL/pGina/dotNet45/pGina.Plugin.Exprodo.dotNet45.dll
INSTALL/pGina/dotNet45/Newtonsoft.Json.dll
INSTALL/pGina/dotNet40/pGina.Plugin.Exprodo.dotNet40.dll
INSTALL/pGina/dotNet40/Newtonsoft.Json.dll
INSTALL/user_tracker/...
INSTALL/README.txt
INSTALL/tomcat6/conf/server.xml
INSTALL/tomcat6/conf/tomcat-users.xml
```

The **Calpendo** directory contains the program itself. The other files are examples for configuring **Apache Tomcat 7** and **Apache** web server, described further below.

Installation Overview

You will need:

- A database server, **MySQL** or **MariaDB**. Please email support@exprodo.com if you would like to use a different database server.
- A servlet engine, eg **Tomcat 7**, **Jetty** etc
- **Java JDK 8** or **JDK 11** (or possibly just a **JRE**, depending on the version of your servlet engine)
- An **Apache HTTP** server may be required, or else you can serve directly from the servlet engine. The config files are use **Apache HTTP** server as a front end to **Tomcat**.

Once those things are in place, you need to:

- Copy the **Calpendo** directory into the servlet engine's `webapps` directory
- Change a configuration file so it knows how to connect to the database
- Configure the servlet engine and **Apache**
- Start the servlet engine and **Apache**
- Connect to the server from a browser to make sure it is accepting connections.

Install MySQL

MySQL 5.6 or later or **MariaDB 10.1.2**.

Create The Database And A Mysql User

Open a connection to **MySQL** with:

```
mysql -u root -p
```

or (depending on your operating system) and configuration:

```
sudo mysql
```

and then at the `mysql` prompt:

```
mysql> create database calpendo;
mysql> grant all privileges on calpendo.* to calpendo@localhost
    -> identified by 'somepassword';
mysql> exit;
```

You may choose a different name for the database and/or user if you wish. Later on, you will configure **Calpendo** so it can locate the database and log in - you will just put the connection information into a URL (in file `hibernate.cfg.xml`).

It is not necessary to load any tables into this new database. When **Calpendo** boots for the first time, it will see the tables are missing and will automatically create the required tables and populate some of them with default data.

Install A Java JDK

Follow the installation instructions for the **JDK** for your platform. This needs to be at least **JDK 8**. Note that the **JDK** is required, not just the **JRE** (this is a requirement for **Apache Tomcat**, the servlet server, rather than **Calpendo** itself).

Install A Java Servlet Server

This should be **Apache Tomcat**, or **Jetty** or some other server that supports the **Servlet 2.5** specification. **Calpendo** has only been tested with **Tomcat 7, 8** and **9**, and so this is the preferred server. A future **Calpendo** release is likely to use **Jetty** instead (and also include a much easier installation and upgrade mechanism). The rest of this section assumes you are using **Tomcat** version 7.

You should read and follow the standard documentation for **Tomcat** to install it. Once installed, you will need to edit two files in **Tomcat's** `conf` directory, `server.xml` and `tomcat-users.xml`. You should find samples of each of these with the **Calpendo** distribution. The important parts of these are:

`tomcat-users` This needs to contain a `<user...>` line so that somebody can connect to **Tomcat's** web management pages. The user name and password can be anything and will only ever be used by a human. You may want more than one entry here, depending on who will do the administration.

`server.xml` The sample provided defines a `<Connector ...>` entry on port 8080 that allows users to connect directly to **Tomcat**. A standard **Tomcat** `server.xml` would contain this entry, although the sample has the necessary config so that it supports compression. If all access is to be made through **Apache**, then this connector on port 8080 could potentially be turned off. Leave it on for now so you can test the installation before we bring **Apache** into the equation.

There's also a `<Connector...>` on port 8009 with protocol **AJP/1.3**. This is the means by which **Apache** talks to **Tomcat**.

Once done, check that you can connect to **Tomcat** and access its management pages. Go to the web page:

`http://servername:8080/`

where `servername` is the name of the server you installed **Tomcat** on. There should be a link on there to the management pages so you can ask **Tomcat** to start **Calpendo** (after **Calpendo** is installed).

Install Calpendo

Calpendo is delivered as a directory in war file format. However, there's a file inside the directory (`WEB-INF/classes/hibernate.cfg.xml`) that you will need to modify to tell **Calpendo** how to connect to the database.

The root directory in the archive is called `Calpendo`. You should put this directory into **Tomcat's** `webapps` directory. Then edit the file

`webapps/Calpendo/WEB-INF/classes/hibernate.cfg.xml`

The entries you need to configure are:

```
<property name="hibernate.connection.username">
    calpendo
</property>
<property name="hibernate.connection.password">
    somepassword
</property>
<property name="hibernate.connection.url">
    jdbc:mysql://localhost/calpendo
</property>
<property name="hibernate.connection.driver_class">
    org.mariadb.jdbc.Driver
</property>
<property name="hibernate.dialect">
    com.springsolutions.exprodo.core.server.persistence.ExprodoMySQL5Inno
    DBDialect
</property>
```

You need to change the username, password and url to match the user and database name you created earlier.

You should now ask **Tomcat** to start **Calpendo** (in case it isn't done automatically). Go to the web management page, linked from:

`http://servername:8080/`

and see if **Calpendo** has started, and if not, ask for it to be started.

Once this is done, then you should test to see whether **Calpendo** is working with **Tomcat**. Open a web browser at this url:

`http://servername:8080/Calpendo/com.springsolutions.calpendo.Calpendo/program.html`

All being well, you will see a **Calpendo** login screen. If you don't, then you'll need to check to see what errors are on screen or in the **Tomcat** logs.

If you do see the login screen, then try to log in with **username** `root` and a blank **password**.

Install Apache HTTP Server

Follow the **Apache** documentation as appropriate for your operating system in order to install the **Apache HTTP** server. Once installed, you should make sure your **Apache** configuration loads modules **mod_proxy**, **mod_proxy_http**, **mod_rewrite**, **mod_deflate** and **mod_ssl**. The location of your modules may vary from that shown below, so you may need to alter this sample configuration:

```
LoadModule proxy_module /usr/lib/apache2/modules/mod_proxy.so
LoadModule proxy_http_module /usr/lib/apache2/modules/mod_proxy_http.so
LoadModule rewrite_module /usr/lib/apache2/modules/mod_rewrite.so
LoadModule deflate_module /usr/lib/apache2/modules/mod_deflate.so
LoadModule ssl_module /usr/lib/apache2/modules/mod_ssl.so
```

For **mod_deflate**, you will need to add config like this:

```
<IfModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css
    text/javascript
</IfModule>
```

This allows **Apache** to `gzip` the files it serves. This has a marked impact on the size of **Calpendo javascript** files in particular.

You will now need to configure the **Calpendo** aspects of **Apache**. You will have your own requirements here, but here is the configuration you will need to provide access to **Calpendo** from a URL of "`https://yourServer/Calpendo`".

```
RewriteEngine On
RewriteRule ^/Calpendo/Service(.*) /Calpendo/com.springsolutions.calpendo.Calpendo
RewriteRule ^/Calpendo/program.html(.*) /Calpendo/com.springsolutions.calpendo.Cal
RewriteRule ^/Calpendo/$ /Calpendo/com.springsolutions.calpendo.Calpendo/program.h
RewriteRule ^/Calpendo/(.*) /Calpendo/com.springsolutions.calpendo.Calpendo/$1 [PT]
ProxyRequests Off
<Proxy *>
    Order allow,deny
    Allow from all
</Proxy>
ProxyPass /Calpendo http://localhost:8080/Calpendo
ProxyPassReverse /Calpendo http://localhost:8080/Calpendo
```

The above configuration should be placed inside a **VirtualHost** declaration. If you want to provide both **SSL** and **non-SSL** access to **Calpendo**, then you should place the above configuration in both the **SSL** and **non-SSL VirtualHost** areas. At this point, accessing **Calpendo** from "`https://yourServer/Calpendo`" should work, possibly unless you are using an operating system that uses **SELinux**, as described below.

Note that previous versions of this installation guide differed in two ways from the above:

- Previously, it was suggested that you should use **mod_jk** to connect **Apache HTTP** server with your servlet container (such as **Apache Tomcat**). This is no longer recommended. In particular, **mod_jk** isn't available for recent versions of **Red Hat Linux** and **CentOS**. That is why the above recommended **Apache** configuration uses **mod_proxy_http**.

- It used to be a requirement that **http** access was required, at least from `localhost`, even when you only wanted **https** access to be remotely accessible. This requirement no longer exists so you only need to configure the **SSL VirtualHost** with the above directives if you only want to provide **SSL** access.

SELinux Settings

If you are running a version of **Linux** that uses [SELinux](#), then you may find that your **Apache HTTP** server is prevented from opening further **http** connections, thereby stopping the above from working. You can check whether you will need to do this by running this command:

```
getsebool httpd_can_network_connect
```

If this returns zero or off, then you will need to change this setting by running this command:

```
setsebool -P httpd_can_network_connect 1
```

You can read about the `httpd_can_network_connect` setting on the [httpd_selinux man page](#). If your `httpd_can_network_connect` setting is zero or off, then the result will be an error in your **Apache** logs similar to this:

```
(13)Permission denied: proxy: HTTP: attempt to connect to 127.0.0.1:8080 (localhost)
```

Rebooting

Finally, you will need to arrange for **MySQL**, **Tomcat** and **Apache** to be started each time your operating system boots. You should follow the standard method for your platform. For **Windows**, this means they should all have been installed as services.

Advanced Apache Configuration

Sometimes, you may have an external-facing **Apache HTTP** server through which all of your traffic flows (eg on a firewall) and then internally you may have another **Apache HTTP** server that has the `mod_jk` module installed. In this situation, with two **Apache HTTP** servers and one **Tomcat** server, there is some extra configuration to do. Without any extra configuration, **Tomcat** will always be given the IP address of the firewall **Apache HTTP** server as the source of each request instead of that of the user's computer. The problem with such a situation is that the login attempts that **Calpendo** will record will all show the firewall as the source of every attempt. For defence against any attack, it would be better to change this.

You can set up forwarding of **HTTP** requests from the firewall to the inner **Apache HTTP** server in two ways: by configuring a reverse proxy with the **ProxyPass** directive, or with `mod_rewrite` and the `[P]` directive.

If you use **ProxyPass**, then you can make a configuration change on the firewall **Apache HTTP** server by adding the directive

```
ProxyPreserveHost On
```

to the appropriate part of the firewall's **Apache HTTP** server configuration.

This situation is described here:

```
http://tomcat.apache.org/connectors-doc/generic\_howto/proxy.html
```

You may be able to achieve the desired effect with **SetEnv** or **SetEnvIf** directives instead of the above (but I have not tested this).

Once you have done this, you can check whether this is working as expected in one of two ways. First, you can look at entries that appear in the **login_attempts** table whenever somebody tries to login to **Calpendo**. Secondly, you can check from **Calpendo** by using either the [Search](#)¹¹⁸ page or the [Data Explorer](#)¹³⁹ page.

Upgrading Calpendo

Upgrades follow the following path:

1. Stop **Tomcat**
2. Take a backup of your MySQL database, this should be done before every upgrade (and *must* be done before major upgrades) because **Calpendo** applies database updates automatically when a new version is first run.


```
mysqldump -u root --password=somepassword calpendo > calpendo.saved.sql
```
3. Move your existing `Calpendo` directory out of tomcat's webapps directory to somewhere else.
4. Unpack the contents of the new tar.gz file, and move the `Calpendo` directory from there into your tomcat's webapps directory.
5. Copy the hibernate.cfg.xml file from your old `Calpendo/WEB-INF/classes/` directory to the new directory
6. Make sure the old `Calpendo` directory is not in tomcat's webapps directory. (If there are two `Calpendo` directories in the webapps directory at the same time, then both will try to run. If they point to the same database, then things may go wrong. If they are different versions of **Calpendo**, then things will probably go wrong.)
7. Open a web browser to confirm that **Calpendo** is accepting connections

Rolling Back a Failed Database Upgrade

If you should want to roll back the database because an upgrade failed in some way, then you should *not* load you back-up over the top of the upgraded database. That's because there could be tables for the upgrade that remain in place, and so it might cause problems with doing the upgrade again (since it may do the wrong thing since it will detect the new tables or columns exist and assume it doesn't need to do an upgrade).

The recommended way to roll back the database after an upgrade is to drop the database, recreate it, and then load the backup.

Installing Files Required For PDF Generation

Calpendo supports two different means of generating PDF files. The older of these requires installation of `groff` and `mom` and relies on system calls outside Java to run it. We no longer recommend this, and instead much prefer to use [XSL FO](#) as the route towards generating PDF files. This means you do not need to install `groff` and should only consider using this method if you are very familiar with `groff` and `mom`.

Ubuntu

```
$ sudo apt-get install groff
```

The `groff` package includes both the full set of 'mom' macros needed for creating documents and the extras for creating PDF files.

CentOS

```
$ sudo yum install groff
$ sudo yum install groff-perl
```

The `groff` package includes the full set of 'mom' macros needed for creating documents.

The `groff-perl` package includes the extras for creating PDF files.

Part



IX

9 Glossary

Term	Definition
Abdicated	The process of indicating that you will not use a booking ⁶⁵² slot allocated to you by a Time Template ⁶⁵⁶ , so that it can be used by other people.
Acceptable	A Time Template ⁶⁵⁶ is acceptable when bookings ⁶⁵² matching the Time Template ⁶⁵⁶ should be allowed, and should be given the status of Requested.
Action	See Permission Action ⁶⁵⁴ .
Applies To	Defines the list of users that are will affected. This list is overruled by the Does Not Apply To ⁶⁵³ list. ie. If a user is in both lists then the Does Not Apply To ⁶⁵³ is used.
Approved Booking	A booking ⁶⁵² whose status is approved.
Approved User	A user that has been approved for access to the system.
Approved Project	A project ⁶⁵⁴ that has been approved for use by an administrator.
Automatic Approval	A Time Template ⁶⁵⁶ has a state of automatic approval when bookings ⁶⁵² matching the Time Template ⁶⁵⁶ should be allowed, and automatically given the status of Approved.
Automatic Denial	A Time Template ⁶⁵⁶ has a state of automatic denial when bookings ⁶⁵² matching the Time Template ⁶⁵⁶ should not be allowed.
Bakery	An editor which allows the user to create, delete and update BiskitDefs ⁶⁵² and their properties ⁶⁵⁵ . The Bakery also allows editing of Mapped Strings ⁶⁵⁴ , Mapped Integers ⁶⁵³ and Units ⁶⁵⁶ .
Best Possible	A booking status ⁶⁵³ that indicates that Calpendo should create an approved booking ⁶⁵² if possible, and a requested booking ⁶⁵⁵ otherwise.
Biskit	A Biskit is an object stored in the database that has a custom type. It will have whatever properties ⁶⁵⁵ are specified by its BiskitDef ⁶⁵² .
BiskitDef	Defines the properties ⁶⁵⁵ associated with a particular type of Biskit ⁶⁵² and how those properties ⁶⁵⁵ should be stored in the database.
Biskit Type	Each Biskit ⁶⁵² has a type. Biskits ⁶⁵² with the same type, have the same property ⁶⁵⁵ definitions. See BiskitDef ⁶⁵² .
Bit Set	A Bit Set is a type of integer property ⁶⁵⁵ that results in a drop-down that can take multiple values.
Booker	The user who has made a particular booking ⁶⁵² .
Booking	A reservation of a resource ⁶⁵⁵ at a particular time. Bookings may also be repeatable ⁶⁵⁵ .
Booking Approval Process	The process by which a booking ⁶⁵² becomes approved.
Bookings Calendar	A page which provides a graphical view of bookings ⁶⁵² by day, week or month.
Booking Rule	A restriction relating to which bookings ⁶⁵² can be made.

Booking Rule Type	The different types of booking ⁶⁵² regulations. E.g. Simple, Double, Advanced, Total Time, Number of Bookings etc.
Booking Rule Validator	A tab in the Booking Rule Editor that provides a means of testing Booking ules ⁶⁵² .
Booking Status	The current condition of the booking ⁶⁵² . There are five possible statuses Requested, Approved, Tentative, Denied or Cancelled.
Booking Type	A user-defined Booking ⁶⁵² taxonomy.
Bookmark	A collection of resources ⁶⁵⁵ to be displayed in the Bookings Calendar ⁶⁵² . Each user can create their own personal bookmarks, and there may also be system wide bookmarks.
Condition	Allows the user to specify under what circumstances an action such as generating an email, granting or refusing permission ⁶⁵⁴ etc. will occur. Conditions work logically together using information from the database to reach a True/False result.
Data Explorer	A page that shows Biskits .
Data Type	See Biskit Type ⁶⁵² .
Default Booking Status	A globally set up preference that specifies the status of a booking ⁶⁵² if no Time Template ⁶⁵⁶ is available.
Does Not Apply To	Defines the list of users that an operation (Rule ⁶⁵² , Permission ⁶⁵⁴ etc) will not affect. This list overrules the Applies To ⁶⁵² list. ie. If a user is in both lists then the Does Not Apply To is used.
Event	An Exprodo SDM concept which defines a when a task ⁶⁵⁶ needs to occur.
FAQ	Frequently Asked Questions, this page lists the FAQ's for the facility.
Far Term Time Time Template	Time Templates ⁶⁵⁶ allow the option of having a different Time Template used beyond a certain time period. This is usually to restrict bookings ⁶⁵² beyond this time period, this is known as the Far Term Time Template.
Global Preferences	Specifies settings that change the way Calpendo operates.
Group Report	A report that lets you break down data by any number of properties ⁶⁵⁵ .
History	The audit log kept by Calpendo that shows what has been changed, when and by whom.
Layout Editor	The Layout Editor allows the user to define how the properties ⁶⁵⁵ of a Biskit ⁶⁵² will be displayed.
List Report	A report that shows a list of Biskits ⁶⁵² .
Mapped Integer	A Mapped Integer relates text labels to integer values. An integer property on a Biskit ⁶⁵² may use a Mapped Integer so that a user would see a drop-down with each of the text labels when they edit it. The related integer value is stored in the database rather than the text. See also Bit Set ⁶⁵² .

Mapped String	Similar in concept to a Mapped Integer ⁶⁵³ , but instead of storing an integer in the database, a text value is stored instead. This means that a Mapped String is an association of one set of strings to another set of strings so that you see a drop-down of text values, but the value stored in the database does not have to be the same as the displayed value.
Meta-Property	1) Attributes that define how a Biskit ⁶⁵² or a Biskit's properties ⁶⁵⁵ work. 2) A type of property used in Conditions ⁶⁵³ when you want to specify a relation ⁶⁵⁵ about who performs an action or when.
Network Metrics	A record of how long communications between web browsers and the Calpendo server take.
Nobody	A special user used for Permissions ⁶⁵⁴ to control what information may be written into emails.
Near Term Time Template	Time Templates ⁶⁵⁶ allow the option of having a different Time Template known as the near term Time Template used when getting close to a particular booking ⁶⁵² time. This allows unused booking ⁶⁵² slots to be used.
Owner	The user that owns a booking ⁶⁵² , bookmark ⁶⁵³ or project ⁶⁵⁴ . Bookings ⁶⁵² may be owned by users other than the booker ⁶⁵² .
Permission	How the administrator determines which user may have access to what information and under which circumstances. Permissions may have many layers allowing fine grained control over who can perform what action ⁶⁵² on which Biskit ⁶⁵² .
Permission Action	An operation that can be performed on a Biskit ⁶⁵² . The most common actions are Create, Read, Update and Delete.
Process	A means of editing a Biskit ⁶⁵² in multiple steps
Project	A project allows an administrator to link together users, resources ⁶⁵⁵ and costs as well as other customised information to enable them to organise who can make bookings ⁶⁵² for particular resources and how they will be charged.
Project Approval Process	The process by which new projects ⁶⁵⁴ become approved.
Project Code	A unique string used to define a project ⁶⁵⁴ .
Project Group	A way of grouping similar projects ⁶⁵⁴ together for use in Rules ⁶⁵² , Permissions ⁶⁵⁴ , Time Templates ⁶⁵⁶ or Automatic Emails. A project ⁶⁵⁴ may belong to many groups. See also Project Type ⁶⁵⁵ .
Project Resource Settings	The list of resources ⁶⁵⁵ that a project ⁶⁵⁴ may use, including information that may be added such as the price or booking ⁶⁵² duration for each resource ⁶⁵⁵ .
Project Service Settings	The list of services ⁶⁵⁵ that a project ⁶⁵⁴ may use, including information that may be added such as the price.
Project Status	The current status of a project. This can be Requested , Approved , Denied , Unbookable or Terminated .

Project Template	A project ⁶⁵⁴ that would normally have a status of Unbookable, and is used to set default values for newly created projects ⁶⁵⁴ .
Project Type	A way of grouping similar projects ⁶⁵⁴ together. A project has only one type. See also Project Group ⁶⁵⁴ .
Property	An individual piece of data attached to a Biskit ⁶⁵² . A property will have a name, a type and a label as well as other meta properties ⁶⁵⁴ defining how it is organised.
Property Path	When adding a property ⁶⁵⁵ to a report, or placing a condition ⁶⁵³ on a property ⁶⁵⁵ , you can choose a property ⁶⁵⁵ of a property ⁶⁵⁵ . This is known as a property path. For example, if you want a Permission ⁶⁵⁴ to prevent creating reports that are scheduled to run daily, you would need a condition ⁶⁵³ on the the report schedule's repeat type. This is found in the property path reportSchedule.repeat.repeatType .
References	Biskit ⁶⁵² properties ⁶⁵⁵ may have various types, such as integer and string. One such type is Biskit ⁶⁵² , which means that a Biskit ⁶⁵² may reference another Biskit ⁶⁵² . The References for a Biskit ⁶⁵² is the list of all such referencing Biskits ⁶⁵² .
Relation	Provides a means of comparing the value of a property ⁶⁵⁵ to another value e.g. equals, less than etc.
Repeat	Something that occurs at regular time intervals.
Requested Booking	A booking ⁶⁵² that has been created but not yet approved ⁶⁵² .
Requested Project	A project ⁶⁵⁴ that has been created but not yet approved ⁶⁵² .
Requested User	A user that has been created but not yet approved ⁶⁵² .
Resource	A room, person, instrument, or anything else that must be booked before it can be used.
Resource Group	A way of grouping similar resources ⁶⁵⁵ together. A resource ⁶⁵⁵ may belong to many groups. See also Resource Type ⁶⁵⁵ .
Resource Type	A way of grouping similar resources ⁶⁵⁵ together. A resource ⁶⁵⁵ has only one type. See also Resource Group ⁶⁵⁵ .
Resource Usage	The amount of time a resource ⁶⁵⁵ is actually used as opposed to the time it is booked ⁶⁵² for.
Resource Usage Calendar	A page which provides a graphical view of resource usage ⁶⁵⁵ by day, week or month.
Resource Usage Recorder	A page that allows you to indicate when a resource ⁶⁵⁵ is actually used. Also known as an Electronic Log.
Settings	See User Settings ⁶⁵⁶ .
Service Order	A request for a Service ⁶⁵⁵ , an Order will have a status and a creator
Service Provider	The provider of a Service ⁶⁵⁵ . A provider may provide many different Services ⁶⁵⁵ .
Services	The ability to set up different tasks that a user may request
Single Item Report	Shows a report of a single item of the appropriate Biskit Type ⁶⁵² .

Summary Report	A report like a spreadsheet pivot table that can automatically count or sum data.
System Event	Various things that happen within Calpendo are recorded as a System Event. For example, a system event is recorded each time an email is sent or an error is detected.
System Usage Statistics	These are statistics that may optionally be kept to track how much each user has been using Calpendo .
Task	An Exprodo SDM concept of an individual action that needs to be performed. See also Event .
Template	See Time Template ⁶⁵⁶ or Project Template ⁶⁵⁵ .
Time Template	Allows the administrator to assign an acceptability rating to bookings ⁶⁵² depending on when they are booked, who makes the booking and the project ⁶⁵⁴ the booking is for.
Time Templates Calendar	A page which provides a graphical view of Time Templates ⁶⁵⁶ by day, week or month.
Trigger	A Workflow Event or Action
Units	A method of displaying different measurement types such as feet and metres.
User Approval Process	The process by which new users become approved.
User Group	A way of grouping similar users together. A user may belong to many groups. See also User Type ⁶⁵⁶ .
User Project Association	A list of projects ⁶⁵⁴ the user is associated with.
User Roles	Users may be assigned roles, and then Permissions ⁶⁵⁴ can be configured to use those Rules. For example, by default, users with the Admin role ⁶⁵⁶ can modify more things than users without the Admin role.
User Settings	Specifies settings that change the way Calpendo operates for this user. These will override the Global Preferences ⁶⁵³ .
User Status	The current status of the users account. This may be one of Requested, Normal, Password must be reset at next login, Blocked or Denied.
User Type	A way of grouping similar users together. A user has one type. See also User Group ⁶⁵⁶ .
Workflow	A way of automating additional functionality in Calpendo .
Workflow Action	A request to do something during a Workflow
Workflow Event	A method of triggering a Workflow

Index

- A -

Abdicate 63
 Actual Resource Usage
 Enabling For A Particular Resource 287
 Actual Usage
 pGina 296
 pGina Authentication 306
 pGina Authentication (Calpendo Server Side) 308
 pGina Authorisation 309
 pGina Computer Monitoring 304
 pGina Configuration 302
 pGina Exprodo Plugin Services 304
 pGina Exprodo Plugin Settings 303
 pGina Gateway 310
 pGina Initial Installation 297
 pGina Notification 312
 pGina Plugin Order 313
 pGina Unblocking the DLL files 300
 pGina Updating Plugin 299
 Recording 293
 Admin Role
 Introduction To 163
 Projects That Can Be Booked 157
 Receiving New User Request Emails 213
 Advanced Booking Rule
 API 271
 API
 Accessing Using wget 634
 Applicability Of Relations 634
 Conditional Queries 634
 Appearance
 Global Preferences For 510
 Approving
 Booking Requests 151
 New User Requests 167
 Project Requests 158
 Attachment, File
 Attaching Documents to Biskits 216
 Creating A Set Of Bisket Def 588
 Example Adding Properties For 580
 Authentication Methods 499

Basic 500
 Editor 505
 Exprodo 500
 External 502
 External Attribute Map 504
 External Proxy 502
 IMAP 500
 LDAP 501
 Local 499
 SMTP 500

Automatic Email
 See Workflow Actions Email 379

- B -

Backing Up The Database 628
 Use Custom Per-Table Settings 629
 Bakery 537
 Biskit Definitions 539
 Biskit Format 542
 Biskit Inheritance 543
 Biskit Property Definitions 549
 Bit Sets 554
 Boolean Property Definitions 550
 Check Reserved Words Preference 525
 Double Property Definitions 551
 Edit Mode 570
 Editor 568
 Editor Example 574
 Example Adding A New Yes-No Mapped Int Property 584
 Example Adding Created/Updated/Version Properties 582
 Example Adding File Attachment Properties 580
 Example Adding Formulaic Properties 577
 Example Adding Properties 574
 Example Booking Cost Formulae 577
 Example Creating A Hierarchy Of Bisket Def 592
 Example Creating A Master Slave Biskit Relationship 597
 Example Creating A New Basic Bisket Def 587
 Example Creating A Set Of Bisket Def 588
 Example Creating An Inheriting Biskit Def (Booking) 602
 Formulae 564
 Formulaic Properties 564
 Integer Property Definitions 552

- Bakery 537
 - Integer Type 552
 - Java Enum 555
 - Java Enum Definitions 555
 - Java Enum Property Definitions 555
 - Mapped Integers 552
 - Mapped Strings 562
 - Master Slave 549
 - Property Definitions 544
 - Property Layout Editor 603
 - Property Storage Mechanisms 537
 - Run Data Definition Validation At Boot 525
 - Set And List Property Definitions 556
 - String Enum 563
 - String Enum Property Definitions 563
 - String Enumerations 564
 - String Property Definitions 558
 - Tag Properties 567
 - Tags 567
 - Updating The DB 568
 - User Roles 554
 - View Mode 569
- Banner, Global Preferences For 510
- Barcode 559
- Barcode Type 560
- BeanShell
 - Advanced Booking Rules 269
- Biskit
 - Biskit Property Definitions 549
 - Biskit Property Type 549, 557
 - Collection Editor, Properties Visible In 144, 547
 - Definitions 539
 - Detail, Properties Visible In 144, 547
 - Dynamic Biskits 542
 - Format 542
 - Inheritance 543
 - Integer Property Definitions 552
 - Integer Type 552
 - Java Enum 555
 - Java Enum Property Definitions 555
 - List, Properties Visible In 144, 547
 - Many To Many Properties 557
 - Master Slave 549
 - Properties Visible In Biskit Detail 144, 547
 - Properties Visible In Collection Of 144, 547
 - Properties Visible In List Of 144, 547
 - Run Data Definition Validation At Boot 525
 - Set And List Property Definitions 556
 - String Enum 563
 - String Enum Property Definitions 563
 - String Property Definitions 558
 - String Type 559
- Biskit Property Type 549, 557
- Bit Sets 554
- Booking Rules 65
 - Advanced Rule API 271
 - Advanced Rules 269
 - Applies To 238
 - Apply To Admin, Preference For 531
 - Applying To Bookings 238
 - Booking Time To Include 267
 - Bookings To Count 241
 - Bookings To Ignore 241
 - Bookings To Include In Count 266
 - Choosing Between Time Templates, Booking Rules And Permissions 223
 - Configuring 235
 - Does Not Apply To 241
 - Double Booking Rules 249
 - Duration Rule 256
 - Editor 236
 - Exclusive Booking Rules 249
 - Global Preferences For 531
 - Holiday Rule 257
 - How They Work 235
 - Interval Rule 258
 - Java Rules 269
 - Number Of Bookings Rule 259
 - Predefined Slots Rule 262
 - Pre-requisite Booking Rules 249
 - Search Results Rule 263
 - Simple 243
 - Total Time Booked Rule 264
 - Types 243
 - Validator 277
- Booking Sub Type
 - Example Creating An Inheriting Biskit Def 602
 - Setting Up 215
- Bookings 39
 - Administration 148
 - Approval Process 150
 - Approving 151
 - Automatic Booking Approval Process 221
 - Background Colours 515
 - Background Tooltips 515

- Bookings 39
 - Best Possible 54
 - Calendar 39
 - Calendar, Creating Bookings 53
 - Calendar, Display 51
 - Calendar, Editing and Cancelling Bookings 58
 - Calendar, Exploring 40
 - Canceled 60
 - Changing More Than The Status 153
 - Colours, User Settings For 35
 - Controlling Bookings 221
 - Creating Bookings In The Past 288
 - Creating Repeat Bookings 57
 - Day, Week, Month View 44
 - Default Status 515
 - Default Status For Bookings Created By Admin 515
 - Default Status For Bookings Created By Users 515
 - Deleting 60
 - Enabling Changes To Old Bookings 288
 - Enabling Pre-defined Time Slots 289
 - Example Creating An Inheriting Biskit Def 602
 - Filtering On The Calendar 43
 - Format 515
 - My Bookings 71
 - My Projects' Bookings 72
 - Owner 515
 - Permissions 65
 - Properties 149
 - Reminder For A Particular Booking 56
 - Reminder, Global Preferences For 515
 - Reminders, User Settings For 34
 - Repeat Flushing, Time Between 525
 - Requests Page 151
 - Resource 41
 - Resource Columns 515
 - Restrictions 62
 - Search 73
 - Search For Cancellations 70
 - Searching For 66
 - Status, Default of 54
 - Status, Filter For The Calendar 43
 - Sub Type 215
 - Template Selection 42
 - Time Display 515
 - Time Templates 62
 - Tooltips, User Settings For 35
 - Using Searches 66
 - Bookings To Count
 - Double Booking Rule Advanced Settings 251
 - Bookings To Ignore
 - Double Booking Rule Advanced Settings 251
 - Bookmark Manager
 - User Permissions 183
 - Bookmarks
 - Calendar 182
 - Editing 185
 - Manager 182
 - User Created 182
 - Boolean
 - Boolean Property Definitions 550
 - Buttons
 - Global Preferences For 521
- C -
- Calendar
 - Bookings 39
 - Bookings, Creating 53
 - Bookings, Display 51
 - Bookings, Editing and Cancelling 58
 - Bookings, Exploring 40
 - Exporting iCal 49
 - iCal Export 49
 - Printable View 50
 - Read Only 155
 - Resource Usage 88
 - Time Templates 232
- Calpendo Administration Guide 148
- Calpendo Configuration Guide 210
- Calpendo Quick Start Guide 206
- Calpendo User Guide 28
- Conditions
 - Advanced 107
 - Anatomy Of A Condition 107
 - Combination Options 115
 - Context 108
 - Date Properties And Accuracy 114
 - Nested Conditions 116
 - Panel 108
 - Pattern Matching Text 112
 - Property Path 111
 - Referenced By Property Path 117
 - Relation 111

Conditions

- Text Pattern Matching 112
- Type 109
- Value 112

Configuring

- Booking Rules 235
- Email Preferences 213
- Groups 280
- Initial 213
- Location 284
- New User Requests 213
- Project Properties 218
- Project Template 218
- Projects 217
- Questions 210
- Resources 213, 284
- Service Orders 291
- Service Providers 291
- Services 291
- Time Templates 226
- Types 280

Converting From

- Existing Booking System 217

Creating Users 176

CSS, Preference Setting For 510

- D -

Data Explorer 139

- Editing Multiple Items 140
- How To Edit A Single Item 142

Database

- Backing Up 628
- Mapping Your Own Tables 542

Database Dumps

- Minimum Time Between 525

Date And Time Format

- Global Preferences For 522

Double

- Double Property Definitions 551

Double Booking Rule

- Advanced Settings 251
- Bookings To Count 251
- Bookings To Ignore 251
- Handling Holidays 254

Dynamic

- Biskits 542

- E -

Editor

- Authentication Methods 505
- Bakery 568
- Booking Rules 236
- Bookmarks 182
- FAQ 506
- Import 197
- Location 284
- Mapped Int 572
- Mapped String 572
- Menu 489
- Permissions 322
- Property Layout 603
- Reports 133
- Resource Usage Outcome 283
- Resources 284
- Service Providers 291
- Services 291
- String Enum Def 572
- Time Templates 229
- Tools 573
- Types And Groups 280
- Units 573
- Workflow Editor 485

Electronic Log (See Resource Usage) 85

Email

- Apparent Sender 523
- Authentication 524
- Connection Security 523
- Copy All Outgoing Emails 524
- Enable SMTP Debugging 524
- Global Preferences For 523
- HTML Signature On Outgoing Email 524
- Initial Configuration 213
- Limiting Who Can Be Sent Emails 524
- Reply To 523
- Sending Enabled, Global Preference For 523
- Signature On Outgoing Email 524
- SMTP Port 523
- SMTP Server 523
- Trust Server 523
- URLs Referring To Calpendo, Global Preference For 524
- Using Calpendo To Send 186

Example Automatic Email

- Example Automatic Email
- Send An Email When A Booking Is Cancelled 383
 - Send Email To A User Whenever Thier New User Request Is Denied 382
 - Send Email To An Administrator When A Project Request Created 379
- Example Bakery
- Adding A New Yes-No Mapped Int Property 584
 - Adding Created/Updated/Version Properties 582
 - Adding File Attachment Properties 580
 - Adding Formulaic Properties 577
 - Adding Properties 574
 - Booking Cost Formulae 577
 - Creating A Hierarchy Of Bisket Def 592
 - Creating A Master Slave Biskit Relationship 597
 - Creating A New Basic Bisket Def 587
 - Creating A Set Of Bisket Def 588
 - Creating An Inheriting Biskit Def (Booking) 602
- Example Booking Rule
- Disallowing Bookings More Than 6 Weeks In Advance 244
 - Handling Holidays 254
 - Number of Bookings Rule Example 260
 - Physicist Projects Can Book 4 Weeks In Advance, Biologists 6 Weeks 245
 - Prevent Cancellations With Less Than 24 Hours To Go 248
 - Special Projects And Admins Can Book Any Time In Advance 247
 - Special Projects Can Book Any Time In Advance 246
 - Total Time Booked Rule Example 268
- Example Booking Rule (Advanced)
- Enforce 15 Minute Gaps Between Bookings 275
 - Enforcing Project Resource Settings 273
 - Reject Bookings More Than 6 Weeks Into The Future 272
- Example Permission
- Admins May Create Or Update Anything 328
 - Anbody Can Modify A Booking Request They Created If The Booking Is Still A Request 327
 - Anybody May Create A Booking Request 326
 - Approval Of A Booking Requires An Admin 331
 - Configuring Multi-Step Approval Process 220
 - Hiding Resources And Booking Properties 333
- No One Can Email Password Details 332
 - Only Allow Specified Users To See The Price Being Charged For A Resource 329
- Existing Booking System
- Converting From 217
- Explorer, Data 139
- Exprodo 537
- ## - F -
- FAQ 506
- Configuring FAQ Answers 507
 - Viewing 145
- Far Term
- Time Template 65
- File, Attachment
- Attaching Documrents to Biskits 216
 - Creating A Set Of Bisket Def 588
 - Example Adding Properties For 580
- Formulae
- Known Problems 566
- Frequently Asked Questions
- FAQ Editor 506
 - Viewing 145
- ## - G -
- General
- Global Preferences For 525
- Getting Started 10
- Global Preferences 509
- Appearance 510
 - Booking Calendar Background Template Colours 516
 - Booking Calendar Date Format 518
 - Booking Calendar Start and Finish Time 519
 - Booking Calendar Tooltip When Template Has No Message 517
 - Booking Reminders 515
 - Booking Resource Columns 515
 - Booking Rules 531
 - Booking Status, Default Admin 515
 - Booking Status, Default User 515
 - Bookings 515
 - Bookings Miscellaneous Settings 520
 - Buttons 521
 - Check Reserved Words 525
 - Copy All Outgoing Emails 524

- H -

- | -

Import	190
Attachments	200
Data Errors	202
Example Bookings File	195

Import 190
 Example Files 190
 Example Project File 193
 Example Resource File 192
 Example Time Template File 194
 Example User File 191
 File Attachments 200
 Handling Errors 200
 Header Errors 201
 Import Editor 197
 Options 199
 The Import Page 197
 Violation Errors 203

Initial Configuration
 Before Going Live 213
 Email Preferences 213
 New User Requests 213
 Resources 213

Integer Type
 Property 552

- J -

Java
 Advanced Booking Rules 269
 Enum Definition 555
 Enumerations 555
 Java Enum
 Property 555

- L -

Layout Editor
 CSS Rule Editors 620
 Edit Mode 610
 Information Pane 608
 Moving Properties 611
 Multiple columns 606
 Notes 617
 Organisation Pane 607
 Preview 606
 Property Group Header Information 608
 Property Table Types 612

Licence
 Global Preferences For 526

List Report 120
 Location

Configuring 284
 Location Editor 284

- M -

Mapped Integers 552
 Mapped Strings 562
 Menu
 Default Admin Menu 527
 Default Root Menu 527
 Default User Menu 527
 Global Preferences For 527
 Menu Editor 489
 Run User Workflow Event 494
 Running A Workflow From A Menu 361
 Menu Editor 489
 Biskit Tree Viewer 495
 Creating a New Menu 497
 Custom Search Descriptors 496
 Custom Search Pages 496

More Calpendo Configuration
 Booking Sub Types 215
 File Attachments 215
 Training Records 215

My Bookings 71
 My Projects 82
 My Projects' Bookings 72
 My Projects' Orders 103
 My Projects' Resource Usage 94

- N -

Near Term
 Time Template 65
 Network Metrics
 Enabled, Preference For 528
 Global Preferences For 528
 Minutes Between Sending, Preference For 528
 Network Calls Between Sending, Preference For 528

- P -

Page Banner, Global Preferences For 510
 Password
 Allowing Browser To Remember 532
 Change 31

Password	
Requirements, Preference Setting For	533
Permission Denied	
Choosing Between Time Templates, Rules And Permissions	224
Permission Denied Message	224
Permissions	
Actions	319
Applies To	325
Breaking An Activity Into Its Parts	315
Choosing Between Time Templates, Booking Rules And Permissions	223
Choosing Between Time Templates, Rules And Permissions	224
Conditions	317, 324
Details	323
Does Not Apply To	326
Examples	326
How Permissions Work	314
Layering	315
Overview	314
Permission Denied Message	224
Permissions Editor	322
Positive and Negative	317
Precedence Of	316
Targeting Users	318
Tree	323
Workflow	338
pGina	
Authentication	306
Authentication (Calpendo Server Side)	308
Authorisation	309
Computer Monitoring	304
Configuration	302
Exprodo Plugin Services	304
Exprodo Plugin Settings	303
Gateway	310
Initial Installation	297
Notification	312
Plugin Order	313
Unblocking the DLL files	300
Updating Plugin	299
Preferences	
Global	509
Processes	622
Creating the BiskitDef	622
Creating the Layouts	622
Creating the Process Definition	623
Creating the WorkFlows	626
Project Association	84
Projects	74
Administration	155
Approval Process	158
Approval, Multi-step Process	159
Approval, Single-step Process	158
Approving Or Denying	160
Changing More Than The Status	161
Changing Users Associated With	178
Configuring	217
Configuring Approval Process	220
Configuring Multi-Step Approval Process	220
Configuring Single-Step Approval Process	220
Creating	75
Default Name, Preference Setting For	529
Global Preferences For	529
Modifying	78
My Projects	82
Properties	156, 218
Rendering Type, Preference Setting For	529
Requests Page	159
Requests, Filtering By User Type, Preference Setting For	529
Restricted	158
Roles That Can Book Any	529
Search	83
Searching	78
Template Configuration	218
Template, Preference Setting For	529
User Association Request	84
Using Search	78
Property	
Barcode	559
Barcode Type	560
Biskit Property Definitions	549
BiskitPropertyType	549, 557
Boolean Property Definitions	550
Double Property Definitions	551
Integer Property Definitions	552
Integer Type	552
Java Enum	555
Java Enum Property Definitions	555
Layout Editor	603
Null Allowed	548
Project	548
Property Definitions	544
Required Fields	548

Property

- Set And List Property Definitions 556
- String Enum 563
- String Enum Property Definitions 563
- String Property Definitions 558
- String Property Type 558
- String Type 559
- Tab Layout 548
- Visibility When Biskit Detail Viewed 144, 547
- Visibility When Biskit List Viewed 144, 547

- R -

Read-only Mode

- Consequences Of Putting Into Read-Only Mode 631
- Creating A Read-only Copy 630
- Global Preference For 525
- Requirements For A Read-Only Database 630

References 106

Reminders

- For A Particular Booking 56
- Global Booking Reminder Preferences 515

Repeat

- Bookings, Creating 57
- Time Between Repeat Flushing 525

Reports

- Report Editor 133
- Scheduling 136

Request

- Project 159
- Project Association 84
- User 167

Resource

- Custom Booking Tooltip 290
- iCal Import 290
- Picture 48, 291

Resource Usage 85

- Calendar 88
- Calendar Future Background Colour, Preference Setting For 530
- During Usage Page 87
- Enabling For A Particular Resource 287
- Global Preferences For 530
- IP Address Restriction, Resource Setting For 287
- Large Font, Resource Setting For 287
- My Resource Usage 93

Outcome When All Went Well, Preference Setting For 530

Pre-Usage Page 86

Recorder, Session 85

Resource Selector Page 85

Resource User Selectable 287

Search 90, 95

Search Project Resource Usage 94

Session ID Template, Resource Setting For 287

Session Recorder 85

Using Search 90

Resource Usage Outcome

Editor 283

Resources

- Configuring 284
- Initial Configuration 213
- Resource Editor 284
- Selection 48
- Usage Recorder, IP Address Restriction 287
- Usage Recorder, Large Font Settings 287
- Usage Recorder, Session ID Template 287

Roles

- Admin 163
- Admin And Projects That Can Be Booked 157
- Description Of 163
- Receiving New User Request Emails 213
- Root 163
- User Role, Editing Role Definition 554

Root Role

Introduction To 163

- S -

Search

- Booking 73
- Booking Cancellations 70
- Complex Content Types 124
- Editing Search Information 129
- Exporting Information 132
- General Search 118
- Group Report 127
- List Report 120
- My Bookings 71
- My Orders 101
- My Projects 82
- My Projects' Bookings 72
- My Resource Usage 93

Search

- Order Search 98
- Project 83
- Project Orders 103
- Project Resource Usage 94
- Resource Usage 90, 95
- Resource Usage Search 90
- Saving And Reusing 131
- Service Orders 98
- Summary Report 122
- Using Bookings 66
- Using Project 78

Searching 118

Security

- Brutce Force Password Hacking, Preference Setting For 532
- Forgotten Password Support 532
- Global Preferences For 532
- IP Address Stability 532
- Password Requirements, Preference Setting For 533
- Password, Allowing Browser To Remember, Preference Setting For 532

Send Email 105

Service Orders

- My Orders 101
- Search 98
- Search Project Orders 103
- Using Search 98

Service Providers

- Configuring 291
- Service Provider Editor 291

Services 96

- Available Services 96
- Configuring 291
- Service Editor 291
- Service Orders 291

Setting Up Different Booking Sub Types For Resources 215

Single Sign On 499

Skins, Preference Setting For 510

Starting

- Quick Start Guide 206

Statistics

- Network Metrics, Enabled, Preference For 528
- Network Metrics, Minutes Between Sending, Preference For 528

Network Metrics, Network Calls Between Sending, Preference For 528

Usage Statistics Enabled, Preference For 534

Usage Statistics Time Between Flushes, Preference For 534

Usage Statistics, Time Per Statistic, Preference For 534

Status

Booking Filter For The Calendar 43

Storing Temporary Data

Workflow 339

String Enum

Property 563

String Enumerations 564

String Property Type 558

String Type 559

Summary Report 122

Comples Content Types 124

System Events 204

System Usage

Global Preferences For 534

- T -

Template

Project 218

Text

Pattern Matching in Conditions 112

Time Booked Rule

Limiting 264

Time Templates

Abdicate 63

Apply To Admin, Preference For 535

Bookings 62

Calendar 232

Choosing Between Time Templates, Booking Rules And Permissions 223

Configuring 226

Editing 235

Editor 229

Far Term 65

How Time Templates Work 226

Near Term 65

Preferences For 535

Putting On The Calendar 234

Time Template Preferences 535

Toolbar Button

Standard Definitions 104

- Tooltips
 - Booking, User Settings For 35
- Total Time Booked Rule
 - Booking Time To Include 267
 - Bookings To Include In Count 266
- Training
 - Checking For Before Booking 215
- Types
 - Booking Types 279
 - Project Types 279
 - Resource Types 279
 - User Types 279
- Types and Groups
 - Configuring 280
 - Description 279
 - Editor 280
- User Group Membership, Changing 179
- User Group
 - User Membership, Changing 179
- User Requests Page
 - Request Filtering By User Type 536
- User Settings
 - Appearance 37
 - Booking Reminders 34
 - Buttons 37
 - Calendar View 35
 - Calender Refresh 36
 - Calpendo 34
 - Date & Time 37
 - Editing For Another User 181
 - Editing Your Own Settings 34
 - Email 38
 - Menu 38
- User Type
 - Requirement For 536

- U -

- URL
 - Referring To Calpendo In Emails 524
 - Within Email Action 376
- Usage Statistics
 - Enabled, Preference For 534
 - Time Between Flushes, Preference For 534
 - Time Per Statistic, Preference For 534
- User
 - Administration 163
 - Approving New Users 167
 - Approving Or Denying 168
 - Changing Group Membership 171
 - Changing More Than The Status 169
 - Creating An Account 28
 - Creating Users 176
 - Global Preferences For 536
 - Initial Configuration Of New User Requests 213
 - Modifying Users 177
 - New User Default Roles 536
 - Password Reset 177
 - Project Associations, Changing 178
 - Properties Of A User 163
 - Requests Page 167
 - Requirement For User Type 536
 - Reset Password 177
 - Roles 554
 - Search 172
 - Settings 34, 181
 - Special 181

- V -

- Version History 12
- Visibility
 - Of Properties In A Biskit Collection Editor 144, 547
 - Of Properties In A Biskit List 144, 547
 - Of Properties In Biskit Detail 144, 547

- W -

- Web Browser Compatability 145
- Workflow 334
 - Actions 365
 - Anonymous HTTP Event 342
 - Biskit Create Action 370
 - Biskit Delete Action 370
 - Biskit Function Type 389
 - Biskit Update Action 371
 - Business Days 353
 - Calendar Function Type 402
 - Conversion Function Type 405
 - Create Variables Action 371
 - Custom Function 347
 - Date & Time Function Type 416
 - Delay Action 372
 - Details 487
 - Diff Action 372

- Workflow 334
 - Email Action 373
 - Email Action Examples 379
 - Evaluate Expression Action 384
 - Events 340, 347, 353
 - Execute System Command Action 385
 - File Function Type 432
 - Find Text Action 386
 - For Each Action 387
 - Function Action 388
 - How Workflows Work 334
 - List Extract Action 476
 - List/Set Function Type 435
 - Logical Function Type 446
 - Mathematical Function Type 448
 - Miscellaneous Function Type 466
 - Network Function Type 470
 - Network Message Action 476
 - Permissions 338
 - Relative Time 353
 - Running A Workflow From A Button 363
 - Running A Workflow From A Menu 361
 - Search Action 478
 - Simple Action 479
 - Sort Action 479
 - Storing Temporary Data 339
 - Templated Text Action 482
 - Text Function Type 474
 - Tree 486
 - Type Cast Action 484
 - Use Cases 488
 - Veto Action 485
 - Workflow Editor 485
- Workflow Actions
 - Biskit Create 370
 - Biskit Delete 370
 - Biskit Function Type 389
 - Biskit Update 371
 - Calendar Function Type 402
 - Conversion Function Type 405
 - Create Variables 371
 - Date & Time Function Type 416
 - Delay 372
 - Diff 372
 - Email 373
 - Email Examples 379
 - Evaluate Expression 384
 - Execute System Command 385
 - File Function Type 432
 - Find Text 386
 - For Each 387
 - Function 388
 - List Extract 476
 - List/Set Function Type 435
 - Logical Function Type 446
 - Mathematical Function Type 448
 - Miscellaneous Function Type 466
 - Network Function Type 470
 - Network Message 476
 - Search 478
 - Simple 479
 - Sort 479
 - Templated Text 482
 - Text Function Type 474
 - Type Cast Text 484
 - Veto 485
- Workflow Email Action
 - Attachments 377
 - Attachments Tab 378
 - Body Tab 376
 - Dealing With Sensitive Data 378
 - HTML Email Body Tab 377
 - Property Path 374
 - Recipients Tab 373
 - Reply To Tab 377
 - Subject Tab 375
 - URL 376
- Workflow Event
 - Anonymous HTTP 342
- Workflow Find Text Action
 - Capturing Groups 387
- Workflow Function
 - Barcodes 413
- Workflow Function Action
 - abs 449
 - acos 450
 - add 436, 450, 474
 - addBusinessDays 417
 - addDays 418
 - addHours 418
 - addMinutes 418
 - addToGroup 390
 - addUserLayout 466
 - addUserToProject 436
 - and 446
 - asin 450

Workflow Function Action

assertActiveUser 467
 atan 451
 authenticate 467
 biskitDef 390
 bitwiseAnd 451
 bookingsWeeklyView 402
 calculateWorkingHours 419
 ceil 452
 concatenate 474
 contains 437
 convertUnits 405
 copy 390, 437
 cos 452
 cosh 453
 createAttachment 432
 createCalendarInvite 403
 createDate 419
 createDateRange 420
 createTime 419
 createFile 432
 createList 437
 createNumericSequence 438
 createOSFile 433
 createTemporaryDirectory 433
 createTemporaryFile 433
 csvDecode 474
 csvEncode 475
 cubeRoot 453
 dayOfMonth 420
 dayOfWeek 420
 divide 453
 enumerateBiskitProperties 391
 exculsiveUnion 438
 exp 454
 expandRepeats 391
 expMinusOne 454
 find 391
 first 439
 floor 455
 format 406
 formEncode 392
 fromJson 410
 generateUserWorkflowEventURL 467
 get 439
 getAsBiskit 392
 getAsBiskitList 392
 getAsBoolean 393

getAsDate 393
 getAsDateList 394
 getAsDateTime 394
 getAsDouble 395
 getAsDoubleList 395
 getAsInt 396
 getAsIntList 396
 getAsLong 397
 getAsLongList 397
 getAsString 398
 getAsStringList 398
 getBookableProjects 404
 getBookingTemplates 404
 getCookie 472
 getOSFileInfo 434
 getUsableBookings 404
 getUserInfo 468
 hasProperty 398
 hour 421
 identity 399
 if 446
 intersect 421, 440
 IO24TCP_getStatus 470
 IO24TCP_identity 471
 IO24TCP_setPin 471
 IO24TCP_setPort 472
 isBusinessDay 421
 isNull 455
 isRangeOutsideWorkingHours 422
 last 440
 LDAPSearch 473
 listOSFiles 434
 loadOSFile 434
 log 456
 log10 456
 logSystemEvent 468
 mapBiskitProperties 399
 Mathematical Function 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 460, 461, 462, 463, 464, 465
 max 440, 457
 millisecond 422
 min 441, 457
 minute 422
 minuteOfDay 423
 month 423
 monthZeroBased 423
 multiply 457

Workflow Function Action

nand	447	tanh	464
nor	447	timeDiffBusinessDays	429
or	447	timeDiffDays	429
parseDate	424	timeDiffHours	430
parseDateTime	424	timeDiffMilliseconds	430
ping	473	timeDiffMinutes	430
pow	458	timeDiffSeconds	431
prettyPrintJson	410	toDate	411
printBiskit	400	toDateTime	411
propertyDef	400	toDegrees	464
random	460	toDouble	411
randomDistinct	468	toInt	412
randomListOrder	441	toJson	412
randomURL	469	toLong	412
remainder	460	toPDF	413
remove	442	toPDF_from_docx	414
removeDuplicates	442	toPDF_from_odt	414
removeFromGroup	400	toRadians	465
removeUserFromProject	442	toString	415
reverseListOrder	443	union	445
round	461	warn	469
second	424	weekOfYear	431
setDayOfMonth	425	year	431
setHour	425		
setMillisecond	425		
setMinute	426		
setMinuteOfDay	426		
setMonth	426		
setMonthZeroBased	427		
setProperty	401		
setSecond	427		
setYear	427		
sin	461		
sinh	462		
size	443		
sleep	469		
sqrt	462		
strlen	475		
subList	444		
substring	475		
subtract	444, 463		
subtractBusinessDays	428		
subtractDays	428		
subtractHours	428		
subtractMinutes	429		
sum	444		
tan	463		



Calpendo is produced by Exprodo Software
www.exprodo.com
Email: calpendo@exprodo.com
UK Telephone: 01235 813458
International Telephone: +44 1235 813458