

# Customising RT

Jeremy Rowntree  
IT Systems Manager  
Department of Biochemistry



# Introduction

- Who am I?
- Why am I here?
- What I won't be telling you (much)
- What problem were we trying to solve?
  - Paper-based queue, manually entered into Excel
  - Needed an easy-to-use Web-based input form
  - Needed an easy-to-use management tool
  - Needed to allow all users to see all fault logs



Customising RT – Jeremy Rowntree

2



I'm Jeremy Rowntree, the IT Systems Manager for the Department of Biochemistry. I've worked there since 1993, following on from undergrad and graduate study there since 1983. Yes, that's right, I used to be a biochemist.

<Click> I am here because Ashley Woltering reckoned that the work I did on Request Tracker (commonly referred to as RT) was worth presenting.

<Click> I should also say that I will touch only briefly on how to set up a generic RT installation. There is plenty of documentation on this, written far better than I could manage as our copy of RT was set up by my colleague John Elder. I've done precisely one test install ... which was for this talk.

One final disclaimer. We don't actually use RT for our IT ticket system ... yet. That's in the pipeline for later on this year.

<Click> So, why, you may ask, did I come to be setting up RT in the first place?

Well, some of you may know that the Biochemistry Department moved into a shiny new building last October. As with all new buildings, there were a few teething problems. Actually, they'd reached about 200 when they realised that their paper-based logging system, manually re-typed into Excel every few days wasn't really up to the job.

<Click> So the call came in – can we have a web-based fault log system, please?

<Click> One that's really easy for both the users and the building managers to use?

<Click> And we need everyone to be able to see what's already been logged, to reduce repeat entries

# The solution

- 1) Use RT as the ticket management system
- 2) Create a web front end for entering tickets  
– makes sure the correct details are entered every time
- 3) Create a web page that displays open tickets
- 4) Find a way to import the existing Excel data, and the new web form submissions, while keeping all the values in separate fields



Clearly a job for a ticket management system. We still use Req for our IT queue, but only because I hadn't got round to switching to everyone's favourite, RT.

<Click> Catch is, most RT installations use free text email as the primary route in for requesters. We wanted to capture specific details about the location of each fault, to be able to sort by location, and so on.

I wasn't keen on making the RT interface available directly to all users. That would have meant Webauth-enabling RT and training everyone how to use it.

They were already used to using our web form front-end to Req, which generates the request email from the form input, so I decided to go for the same technique.

<Click> As for letting all users see open tickets, I reckoned it ought to be possible either to export the data from RT, or maybe get at directly from the database.

<Click> I also had a few ideas about how I might import the Excel data. The tricky bit, though, was going to be maintaining the individual field values.

# What we will be covering today

- How to create custom fields
- How to populate them from the request email
- How to create that email via a web form
- How to import existing Excel data
- How to display open fault log tickets to all users
- How to export the custom field values



I'll basically be describing about a month's work, on and off, covering

<Click> Adding custom fields to a standard RT installation

<Click> The technique needed to extract the custom field values from the request email

<Click> How we created the front-end request interface

<Click> How we pulled that 200 record Excel sheet into RT, complete with all the custom fields

<Click> How we enabled users to see all Open tickets without having to log into anything and finally

<Click> The technique we chose to extract detailed ticket reports for fault management meetings

# Installing RT

- Debian installation guides

<http://wiki.bestpractical.com/view/DebianLennyInstallGuide>

<http://www.debianadmin.com/howto-setup-request-tracker-36-on-debian-etch.html>

- I had to do this by hand

```
/usr/sbin/rt-setup-database-3.6 --action init --dba root -prompt
```

- RT-Mailgate

[http://www.oreillynet.com/linux/blog/2007/08/rt\\_mailgate.html](http://www.oreillynet.com/linux/blog/2007/08/rt_mailgate.html)

I had to add the actual host IP (127.0.1.1) to the NOAUTH bit of

```
/etc/request-tracker3.6/apache2-modperl2.conf
```



A short digression here. In order to prepare some of the screenshots for this talk, I needed a spare RT system to work on. I couldn't really use the production one, and didn't want to trouble John with another setup. The production system was done using Debian packages, so I figured I'd do the same. Bear in mind that I maintain Windows servers, not Linux ones, so my stumbles would probably be hurdled easily by seasoned Linux sysadmins.

Essentially, I followed the instructions at these two links.

However, I found that the route I followed didn't populate the database with the initial values, so my first login attempt failed because there was no root user.

<Click> The solution was to run the setup database command manually.

I also found I couldn't get mails into RT. The error code was a 403 - permission denied.

<Click> Turned out that my VMPlayer Debian box was on an IP which wasn't in the IP restricted access list for bypassing the authentication layer. Adding the IP to this file did the trick.

That got me a working basic RT installation. Time to start customising it.

## Remove default queue scrips

- Why? – Because you never know when you might need to turn a scrip off or edit it for just one queue. And you've now got 20 queues ...
- Take a screenshot, delete, add to each queue
- Delete at `Configuration | Global | Scrips`
- Add them back at `Configuration | Queues | General | Scrips`



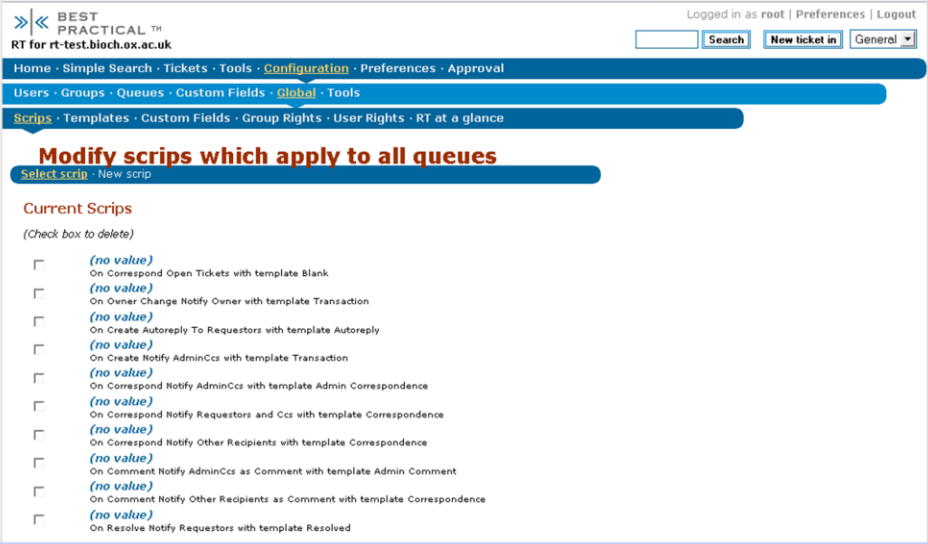
I got this tip from Ashley, who I believe got it from the rather excellent O'Reilly book, RT Essentials.

It may seem like a lot of extra work, but you'll be glad you did it when you find you need to run an import into your twentieth RT queue without sending an email to everyone, but you can't just turn off the Autoreply scrip because it's defined globally, and would affect all your queues. Or you're asked to tweak the template text for just the Web request queue. You get the idea.

<Click> Fortunately, the basic Scrip display screen contains all the information you need to recreate the 10 default scrips, so save a screenshot in a safe place before you delete the global definitions, then refer to it whenever you create a new queue.

Oh, in case you were wondering, a scrip is just RT's name for a script.

# The default scrips



The screenshot shows the configuration interface for BEST PRACTICAL RT. The main heading is "Modify scrips which apply to all queues". Below this, there is a section titled "Current Scrips" with a note "(Check box to delete)". A list of scripts follows, each with a checkbox and a description:

- (no value) On Correspond Open Tickets with template Blank
- (no value) On Owner Change Notify Owner with template Transaction
- (no value) On Create Autoreply To Requestors with template Autoreply
- (no value) On Create Notify AdminCs with template Transaction
- (no value) On Correspond Notify AdminCs with template Admin Correspondence
- (no value) On Correspond Notify Requestors and Cs with template Correspondence
- (no value) On Correspond Notify Other Recipients with template Correspondence
- (no value) On Comment Notify AdminCs as Comment with template Admin Comment
- (no value) On Comment Notify Other Recipients as Comment with template Correspondence
- (no value) On Resolve Notify Requestors with template Resolved

Because I'm feeling kind, here's a copy of that default list.

While you're busy writing them down, (only joking) I'll move on to the hunt for custom field management.

This was one of those Google moments. The type where you pick exactly the right search term first time round.

Suffice to say I knew that it was possible to add custom fields to RT. It's right there on the screen, before Global.

What I needed was a way to extract the custom field values from the request email. Fortunately, that was exactly what the author of the ExtractCustomFieldValues module decided to call his new creation.

# Install ExtractCustomFieldValues

- Download from <http://wiki.bestpractical.com/view/ExtractCustomFieldValues>
- Extract the file
- Run

```
perl Makefile.PL
(path to RT.pm is /usr/share/request-tracker3.6/lib)
make install
make initdb
```

(I had to set up a symlink from /usr/sbin/rt-setup-database to /usr/local/share/request-tracker3.6)



The installation is reasonably straightforward. It starts, of course, with a download

<Click> Then an extract, followed in this case by a Perl command to create the Make file. Debian puts RT.pm somewhere the script wasn't expecting, so I had to feed it the location by hand. Then comes the inevitable Make Install, then some database tweaks. Again Debian confused the script. Nothing a quick symlink couldn't solve though.



# Create your Custom Fields

The screenshot shows the 'Create a CustomField' form in the RT interface. The form has the following sections and fields:

- Name:** A text input field.
- Description:** A text input field.
- Type:** A dropdown menu with 'Enter one value' selected.
- Applies to:** A dropdown menu with 'Enter one value' selected.
- Validation:** A list of options including 'Enter multiple values', 'Fill in one wikitext area', 'Combobox: Select or enter one value', 'Upload one image', and 'Upload multiple images'.
- Link values to:** A list of options including 'Select one value', 'Select multiple values', 'Upload one file', and 'Upload multiple files'.
- Include page:** A list of options including 'Fill in one text area'.

At the bottom right of the form is a 'Submit' button. The footer of the page includes the I C T F logo, 'Customising RT – Jeremy Rowntree', the number '9', and the University of Oxford logo.

Custom fields in RT are very easy to create, and can be constrained to a fixed list of values, or simply declared to be mandatory.

Take care when you're choosing field names if you expect to have multiple RT queues as the fields are defined globally. So you can't have two different fixed value lists for the same field name in two different queues.

You can reuse common custom fields across Solutions queues though.

You can also have spaces in Custom Field names, which helps to make them readable

<Click> As you can see, there is a wide range of field types, including file and image uploads. So far, I've only needed "Enter one value" and "Select one value".

# Assign to one or more queues

The screenshot shows the RT web interface for 'RT for rt-test.bioch.ox.ac.uk'. The user is logged in as 'root'. The breadcrumb trail is: Home · Simple Search · Tickets · Tools · Configuration · Preferences · Approval · Users · Groups · Queues · Custom Fields · Global · Tools. The current page is 'Select custom field ... Room Number'. The main heading is 'Modify associated objects for Room Number'. Below this, there are tabs for 'Basics', 'Applies to', 'Group Rights', and 'User Rights'. Under 'Selected objects', the 'General' checkbox is checked. Under 'Unselected objects', it says '(None)'. At the bottom of the form area, there are 'Check All', 'Clear All', and 'Submit' buttons. The footer of the interface shows 'Time to display: 0.020115' and 'RT 3.6.7 Copyright 1996-2006 Best Practical Solutions, LLC.'

Once you've saved the basic custom field definition, you need to assign it to at least one queue. Just tick the required queues and Submit.

# Assign permissions

Best PRACTICAL™  
RT for rt-test.bioch.ox.ac.uk

Logged in as root | Preferences | Logout

Home · Simple Search · Tickets · Tools · Configuration · Preferences · Approval

Users · Groups · Queues · Custom Fields · Global · Tools

Select custom field · New custom field · Room Number

## Modify group rights for custom field Room Number

Basics · Applies to · Group Rights · User Rights

### System groups

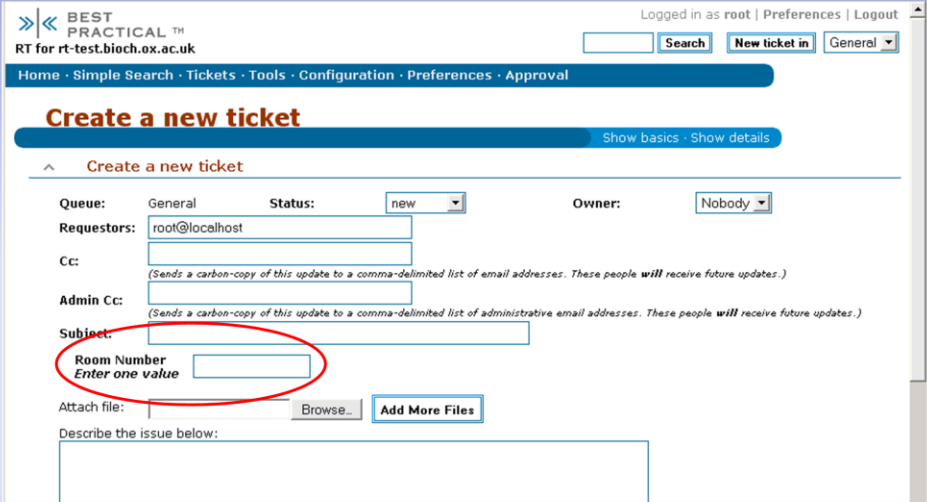
Everyone	<b>Current rights</b> (Check box to revoke right) <input type="checkbox"/> SeeCustomField	<b>New rights</b> AdminCustomField ModifyCustomField (no value)
Privileged	<b>Current rights</b> (Check box to revoke right) <input type="checkbox"/> ModifyCustomField <input type="checkbox"/> SeeCustomField	<b>New rights</b> AdminCustomField (no value)
Unprivileged	<b>Current rights</b> No rights granted.	<b>New rights</b> AdminCustomField ModifyCustomField SeeCustomField (no value)



You also need to assign see and modify permissions for your new custom field. You can either use the built in System groups, or define your own.

Creating User defined groups is covered in the standard RT documentation.

# What have we got so far?



The screenshot shows the 'Create a new ticket' interface in a web browser. The page title is 'Create a new ticket' and the URL is 'RT for rt-test.bioch.ex.ac.uk'. The interface includes a navigation bar with links for 'Home', 'Simple Search', 'Tickets', 'Tools', 'Configuration', 'Preferences', and 'Approval'. The main form has the following fields:

- Queue: General
- Status: new (dropdown)
- Owner: Nobody (dropdown)
- Requestors: root@localhost
- Cc: (text input)
- Admin Cc: (text input)
- Subject: (text input)
- Room Number: (text input) - This field is circled in red and has the text 'Enter one value' below it.
- Attach file: (text input) with a 'Browse...' button and an 'Add More Files' button.
- Describe the issue below: (text area)

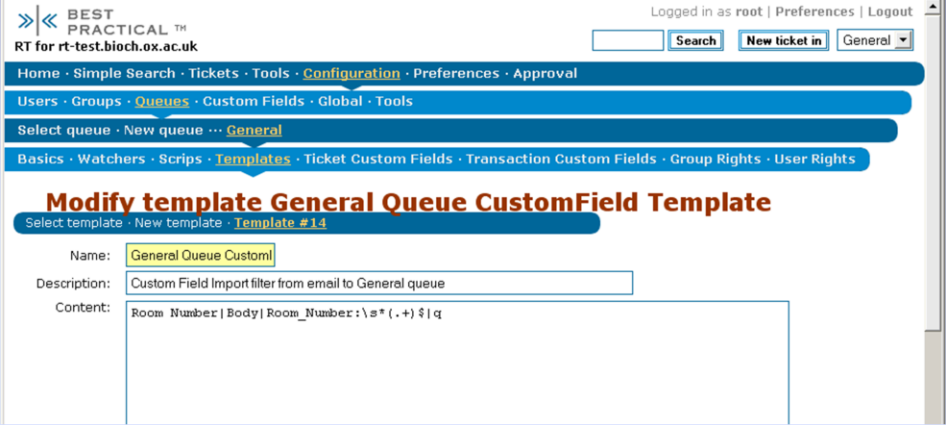
At the bottom of the slide, there are logos for 'ICTF 09', 'Customising RT – Jeremy Rowntree', '12', and 'UNIVERSITY OF OXFORD'.

The end result of the work so far is that the “Create a New Ticket” interface within RT has gained an extra entry, along with a bit of help text.

However, this only affects this internal interface, which might be used by help centre staff who transcribe incoming requests over the phone.

We still need to enable RT to automatically extract field values from incoming emails

# Add an import filter



The screenshot shows the RT web interface for 'BEST PRACTICAL™ RT for rt-test.bioch.ox.ac.uk'. The user is logged in as 'root'. The navigation menu includes: Home · Simple Search · Tickets · Tools · Configuration · Preferences · Approval · Users · Groups · Queues · Custom Fields · Global · Tools. The current page is 'Modify template General Queue CustomField Template'. The form fields are: Name: General Queue Customl, Description: Custom Field Import filter from email to General queue, and Content: Room Number| Body| Room\_Number:\s\*(.+)\s|q. The footer includes the I C T F logo, 'Customising RT – Jeremy Rowntree', the page number '13', and the University of Oxford logo.

To do this we need to create an import filter.

The ExtractCustomFieldValues module uses RT's template system to store the filter definition. As you can see from the name, you would usually create a separate template for each queue.

Templates are more commonly used to hold the boilerplate text for email auto-replies and so on, but since the filter definition is plain text, this technique works fine.

# Import filter syntax

```
Room Number|Body|Room_Number:\s*(.+) $|q
```

Field	Description
1	Custom field name in RT
2	Body or the name of the header field
3	Import filter
4	q = Quiet, as in don't record the import as a separate transaction

Regular Expression - `Room_Number:\s*(.+) $`

Captures the rest of a line that starts `Room_Number :`



The syntax provides a way of matching the name of the custom field within RT to a pattern within the email.

First you give the custom field name as held within RT. As you can see, spaces don't need to be quoted.

The default separator is the pipe character, but this can be redefined within the template text if necessary

<Click> Next you specify where to find the value, either within the body of the email, or by giving the name of a header.

This lets you extract common email header information, such as User-Agent or just the Subject. As we'll see later, it can also be used to hide additional values within the incoming email.

<Click> Then comes the crucial bit, a regular expression to match the required value within the email. For headers `.*` will usually suffice, but for body text you'll need to design a unique pattern. The recommended approach is the one seen here, the name of the field, in this case with an underscore, followed by a pattern to match to the end of the line.

<Click> Finally, there's an Options field, in this case just a `q` for Quiet. This prevents RT from logging the import of the field value as a transaction in its own right. The documentation mentions other possible values that I've not had need of.

You'll need to create one line in the template for each custom field. When all is done, just save the template. You can update it at any time.

The full import filter we used for our faultlog can be found at <https://wiki.oucs.ox.ac.uk/itss/CustomisingRequestTracker>

# Enable import filter

The screenshot shows the configuration interface for the BEST PRACTICAL RT system. The page title is "Create a scrip for queue General". The form is titled "Scrip Fields" and contains the following fields:

- Description: General CustomField E
- Condition: On Create
- Action: Extract Custom Field Values
- Template: General Queue CustomField Template
- Stage: TransactionCreate

At the bottom of the form, there are "Reset" and "Create" buttons. A message "Be sure to save your changes" is displayed next to the "Create" button. The page footer includes the I C T F logo, the text "Customising RT – Jeremy Rowntree", the number "15", and the University of Oxford logo.

Finally, we need to tell RT which filter template to apply to incoming tickets.

To do this we create a new scrip for the queue. The condition is, of course, "On Create".

The Action drop-down contains the only visible evidence that the ExtractCustomFieldValues module has been installed, the imaginatively named extract custom field values action.

The template is, as you might expect, the one we just created.

And finally Stage is set to TransactionCreate, in common with all the default scrips.

Click Create and we're ready to test our new custom field import filter.

# Send an email

```
jr@debian:/etc/request-tracker3.6$ mailx -s "Email with Room Number" rt

Room_Number: 10-026
Cc:
You have new mail in /var/mail/jr
jr@debian:/etc/request-tracker3.6$
```

The screenshot shows the RT web interface for 'BEST PRACTICAL™ RT for rt-test.bioch.ox.ac.uk'. The ticket title is '#12: Email with Room Number'. The 'Ticket metadata' section is expanded to show 'The Basics' and 'Reminders'. The 'Custom Fields' section is circled in red, showing 'Room Number: 10-026'. The 'Reminders' section has a 'New reminder' form with fields for 'Subject', 'Owner' (set to 'Nobody'), and 'Due (yyyy/mm/dd)'.



I used mailx to generate the test emails, largely because it makes short work of sending the same, or similar emails over and over again.

As you can see, aside from the subject, I've just included a single line of text which matches the regular expression we defined earlier.

<Click> The result is a new ticket, complete with the room number from the email stored in its own custom field.



## Controlling the input

- Users won't remember to type  
Room\_Number:
- They can't even get their email address right
- **Solution** – Webauth/OAK LDAP-enabled form
- Extract name and email from Webauth login
- Create form fields that match the custom ones



Clearly, we can't expect requesters to format their request email precisely according to the regular expressions of the import template.

The frequency of bounces from our current manual entry Web interface to Req confirms that many of them can't even spell their email addresses correctly.

So, we'll need a web form to guide the input process, and a script to format the value appropriately and generate an email for dispatch into the RT queue.

<Click> Having worked recently with OAK LDAP for another project, it struck me that this could provide an effective means of ensuring we capture the correct name and email address, at the same time as restricting access to the submission form to valid members of the department.

The rest of the form just needs to request the same fields as we've defined within RT.

# Setting up Webauth / OAK LDAP

- Well documented at

Webauth setup -

<https://wiki.oucs.ox.ac.uk/itss/AppWebAuth?action=show&redirect=WebAuth> & <http://www.oucs.ox.ac.uk/webauth/>

Certificate Service -

<https://wiki.oucs.ox.ac.uk/itss/CertificateService>

OAK LDAP - <http://www.oucs.ox.ac.uk/services/oak/sp/ldap/> & [http://www.oucs.ox.ac.uk/services/oak/sp/ldap/using\\_ldap\\_client\\_software\\_with\\_oak.xml.ID=mod\\_webauthldap](http://www.oucs.ox.ac.uk/services/oak/sp/ldap/using_ldap_client_software_with_oak.xml.ID=mod_webauthldap)



Setting up a Webauth enabled web server has been covered in other talks and is thoroughly documented already, as is the process of connecting your server to the OAK LDAP service.

There are a few choices to be made during the configuration, so I'll briefly run through them now.

The next few slides are mostly for the benefit of on-line readers, so don't worry if you don't follow all the details now.

# Our config files

## /etc/apache2/mods-available/webauth.conf

(as per <https://wiki.oucs.ox.ac.uk/its/AppWebAuth?action=show&redirect=WebAuth>)

```
# webauth.conf -- Basic configuration for the WebAuth module.
# webauth.conf,v 1.1 2004/06/23 06:22:10 eagle Exp

WebAuthKeyring /var/lib/webauth/keyring
WebAuthKeytab /etc/webauth/keytab
WebAuthServiceTokenCache /var/lib/webauth/service_token_cache
WebAuthSSLRedirect on

# Point to the Oxford Webauth service
WebAuthLoginURL https://webauth.ox.ac.uk/login
WebAuthWebKdcURL https://webauth.ox.ac.uk/webkdc-service/
WebAuthWebKdcPrincipal service/webkdc@OX.AC.UK

# If you're having trouble switch on debugging
#WebAuthDebug on
```



The basic webauth conf is standard to all installations, so I'll move straight on to ...

<Click>

# webauthldap.conf

See Section 4.2 of

[http://www.oucs.ox.ac.uk/services/oak/sp/ldap/using\\_ldap\\_client\\_software\\_with\\_oak.xml.ID=mod\\_webauthldap](http://www.oucs.ox.ac.uk/services/oak/sp/ldap/using_ldap_client_software_with_oak.xml.ID=mod_webauthldap)

```
# webauth.conf -- Basic configuration for the WebAuth LDAP module.  
# $Id: webauthldap.conf 2074 2004-06-23 06:22:11Z eagle $
```

```
WebAuthLdapKeytab /etc/webauth/keytab  
WebAuthLdapTktCache /var/lib/webauth/krb5cc_ldap  
WebAuthLdapHost ldap.oak.ox.ac.uk  
WebAuthLdapBase ou=people,dc=oak,dc=ox,dc=ac,dc=uk  
WebAuthLdapSSL on
```

**WebAuthLdapFilter**

```
(oakPrincipal=krbPrincipalName=USER@OX.AC.UK,cn=OX.AC.UK,cn=KerberosRealms,dc=oak,dc=ox,dc=ac,dc=uk)
```

```
WebAuthLdapAuthorizationAttribute eduPersonOrgUnitDN
```



... the configuration of the webauthldap module.

There are two options for the webauth LDAP filter as documented at this link.

We want “**Restricting access based on affiliation with a University unit**”. The alternative is “**Restricting access to members of a given unit with a given status**”

Note that we don’t define which unit we’re interested in at this stage.

# Apache config

## /etc/apache2/httpd.conf

```
<Location /rt>
  WebAuthExtraRedirect on
  AuthType WebAuth

# Limit access to bioch members
  require privgroup oakUnitCode=bioch,ou=units,dc=oak,dc=ox,dc=ac,dc=uk

# Retrieve full name for use in web form
  WebAuthLdapAttribute displayName

# Retrieve email address for use in web form
  WebAuthLdapAttribute mail

  ErrorDocument 401 /401.html
</Location>
```



That comes in the Location definition for the RT input form itself.

Require privgroup takes as a parameter the Distinguished Name of the unit we wish to enable access for

Then we simply list the two OAK LDAP attributes that we wish to retrieve for the newly logged on user.

The values are placed into server variables as we'll see in the next slide.

# The web form itself

- I used PHP here. The OAK LDAP docs cover Perl, Python and Java too.

```
<FORM ACTION="https://rt-test.bioch.ox.ac.uk/cgi-bin/rtmailer.pl" METHOD="post" name="faultlog">
<INPUT type="text" name="realname" value="<?php print $_SERVER['WEBAUTH_LDAP_DISPLAYNAME'] ?>"
SIZE="40" >
<input type="text" name="email" value="<?php print $_SERVER['WEBAUTH_LDAP_MAIL'] ?>" size="40" >
</FORM>
```



Here is a small extract from the form definition, written in PHP.

Bear in mind that I'm only displaying a couple of variables. The OAK LDAP language docs cover direct interaction with the LDAP service. The simplicity of using WebAuthLdapAttribute in the Apache config is that no further code is needed. I'll leave the creation of the rest of the form as an exercise for the reader. Or maybe I'll put it on the Wiki. Either way, there isn't space for it here.

Do note that you'll need to duplicate any Custom Field constraints that you've configured within RT. There's no point having provided a fixed value list for, say, "Operating System", if all the web form offers the user is a free text field.

You'll notice the form action script is a reference to an rtmailer.pl

The full form can be found at <https://wiki.oucs.ox.ac.uk/itss/CustomisingRequestTracker> as "input.php"

<Click>

# The “ACTION” script ...

- ... is just a copy of NMS Formmail from [http://nms-cgi.sourceforge.net/formmail\\_compat-3.14c1.tar.gz](http://nms-cgi.sourceforge.net/formmail_compat-3.14c1.tar.gz)
- Config section:

```
$emulate_matts_code= 0;  
$secure           = 1;  
$allow_empty_ref  = 0;  
$max_recipients   = 1;  
@recipients       = ();  
%recipient_alias  = (  
    'general' => 'rt@bioch.ox.ac.uk',  
$wrap_text        = 0;  
$send_confirmation_mail = 0;
```



Which is nothing more than a copy of NMS Formmail, a well secured Perl script designed to pass on values from a web form to an email.

I’ve not included the full config, only the values that I’ve changed from the defaults or are otherwise worth noting.

The key point of the design is that the target email address is buried in the script itself, and referred to via an alias. You can define several aliases if you wish, each pointing to a different email address, and perhaps, therefore, a different RT queue.

Note that we don’t send a confirmation email as RT itself takes care of that when it receives the ticket request email.

The full script can be found at <https://wiki.oucs.ox.ac.uk/itss/CustomisingRequestTracker> as faultlog.pl

## A few extra form variables

Formmail needs the following in the web form:

```
<input type="hidden" name="recipient"
value="general" />
<input type="hidden" name="return_link_url"
value="http://www.bioch.ox.ac.uk/" />
<input type="hidden" name="return_link_title"
value="Return to the Department of
Biochemistry Home Page" />
<input type="hidden" name="required"
value="realname, email, subject, Fault_Type,
Room_Number, Description" />
<input type="hidden" name="sort"
value="order:Fault_Type, Room_Number,
Description" />
```



Formmail is partially controlled from the web form via hidden variables.

You can see that we've used the general alias from the previous slide as a recipient.

We've also defined a return URL to be displayed after the form is submitted

Formmail allows some basic validation and will request a re-submission of the form if any of these required fields is missing.

Finally, we have the ability to define in which order the body text fields will appear. This will be important later on.



# The end result

**New Biochemistry Fault Log**

**Name:**

**EEmail:**

**Report Type :**  Fault Log  
 New Work Request

**Room number :**

**Location in room :**

**Fault Type :**

**Subject** (One line summary)

**Description** of fault or defect



No, I'm not entering this for any design awards.

Still, you can see how OAK LDAP has supplied my name and email address, and two examples of fixed value restrictions, a radio button and a drop-down list.

Failure to fill in any of the Red fields will trigger the validation check from the previous slide.

Provided everything is entered correctly, the result will be a correctly formatted email being sent to RT, which will extract the custom field values and send an autoreply.

## Dealing with the backlog

- We can now cope with new jobs, but what about the 368 existing ones, many with actions noted but not completed?
- Design a mail sender in Perl which converts Excel via CSV to email content.
- Use the original requester's email address as the From field, so they will receive comments



Building all this took a little while, during which time a further 168 jobs were entered on paper. Many of these were still current, but had actions recorded against them that would need to be displayed to the users.

We needed a script that could populate RT with these requests as if they'd been submitted by the users themselves.

I had a brief look at adding the jobs to the database directly, but each ticket generated far too many records created across a number of linked tables for this to be seriously considered.

<Click> Best bet seemed to be to do the same as the Formmail script, simply generate fake request emails, only using the Excel data as input.

<Click> By using the appropriate From address, we convince RT to correspond with the original requester regarding future updates to their tickets.

I looked briefly at modifying the Formmail script itself, but it turned out to be heavily targeted towards receiving web input, and not that easy to convert.

## Script workflow

- Email content resembles LDIF format
- I already had a CSV to LDIF converter script for Active Directory imports (See ITSS Wiki)
- Bolt that script onto a raw SMTP sender script from <http://www.dreamincode.net/forums/showtopic36108.htm>
- Add header for hidden fields (Action & Job #)
- Import with `Action|X-RT-Action|. *|q`



Instead I looked again at the email content, and recognised that it resembled LDIF format – one value per line, field name first, then it's value.

A while back I'd made use of a CSV to LDIF converter script while developing a link between our Active Directory and our personnel database.

So I dug that script out, and modified it to work with the custom RT fields.

<Click> I then went hunting for a mail sender script. The one I found had the ability to specify the From address separately for each mail. Perfect!

Linking the two scripts together was fairly straight forward.

You'll recall I mentioned the Excel had an Action field. This wasn't part of the request, so I didn't want it included in the email body. Same for the line number in the spreadsheet, which was being used a Job ID value. I didn't fancy trying to reset RT's job counter to zero for the import. RT's design doesn't include the ability to delete jobs, only to hide them. Therefore I'd have had to remake the database, then reapply much of the customisation.

<Click> Instead, I just used the Header import capability of the custom field extract module. You may remember I mentioned that `. *` was enough to import a header.

The full import script can be found at <https://wiki.oucs.ox.ac.uk/itss/CustomisingRequestTracker> as `mailer.php`, together with a sample CSV file snippet

## Running the import

- Turn off the “Autoreply to Requesters” scrip
- Run the “Resolved” import
- Turn the scrip back on again
- Run the “Open” + “New” import



I chose to disable the autoreply scrip when importing resolved jobs. I then warned users to expect email confirmations, then imported open jobs with the scrip turned on. This gave them an email which they could respond to if they needed to chase up the job.

## Letting users see all tickets

- Challenge – Let users see basic info about all open tickets without having to create logins
- RT interface doesn't support this, so take a look at the raw database format
- Ticket contents are spread across Tickets, Transactions and Attachments tables
- Custom fields are stored in another table as one record per field per ticket, identified by a CustomField ObjectID and the Ticket ID



As I mentioned earlier, RT is not well suited to providing information to non-logged in users. It does have a spreadsheet export, but that would need to be run manually then uploaded to the web. Furthermore the spreadsheet content is not customisable so I'd have needed some macro processing to tidy it up for public output.

Instead I spent a little time decoding the raw database structure.

# The SQL

```
SELECT
Tickets.ID AS Job,
Attachments.Content AS Description,
Users.RealName AS Name,
Transactions.Created AS Date,
ObjectCustomFieldValues.Content AS Action,

FROM (Users INNER JOIN
      (Tickets INNER JOIN
        (Transactions INNER JOIN Attachments
          ON Transactions.id = Attachments.TransactionId)
        ON Tickets.ID = Transactions.ObjectId)
      ON Users.ID = Transactions.Creator
    )
LEFT JOIN ObjectCustomFieldValues ON Tickets.ID = ObjectCustomFieldValues.ObjectId

WHERE (((ObjectCustomFieldValues.CustomField)=5)
AND ((Tickets.Status) = 'Open')
AND ((Tickets.Queue) = 3)
AND ((Transactions.ObjectType) = 'RT::Ticket')
AND ((Transactions.Type) = 'Create')
) ORDER BY Job DESC
```



The LEFT JOIN with ObjectCustomFieldValues means that we return one row for each Custom Field of each ticket. By selecting only those rows where ObjectCustomFieldValues.CustomField=5, we limit the data returned to just the rows containing the value of the Action custom field, which happens to have an ID of 5. We limit Status to 'Open' so give the queue manager the chance to vet new ticket content before it goes public. Otherwise, we'd select 'Open' or 'New'.

The full status page script can be found at <https://wiki.oucs.ox.ac.uk/itss/CustomisingRequestTracker> as status.php

# The result

New Biochemistry Fault Log List					
Job #	Name	Date	Description	Action	Status
405	See Mail	2009-07-13 23:29:28	Report_Type: Fault Room_Number: 10-042 Location_in_room: Corner of access corridor Fault_Type: Building Description: There is a leak in the access corridor outside 10-042. This happens every time it rains, and it seems to have to stop it up when it happens at night. Please fix this asap.	Refer to PSG	open



I've blurred the details as I can't imagine our Building Manager would want the whole of ITSS to see precisely what was wrong with his lovely new building. The single open ticket is genuine, though.

# Generating Reports

The screenshot shows a web interface for a ticketing system. At the top, it says "BEST PRACTICAL™ RT for rt-test.bioch.ox.ac.uk". The user is logged in as "root". There are navigation links: Home, Simple Search, Tickets, Tools, Configuration, Preferences, Approval. A search bar and a "New ticket in" button are visible. Below the navigation, it says "Found 2 tickets". There are links for "New Search", "Edit Search", "Advanced", "Show Results", and "Bulk Update". A table lists two tickets:

#	Subject Requestors	Status Created	Queue Told	Owner Last Updated	Priority Time Left
12	Email with Room Number jr@debian.localdomain	open 18 hours ago	General	root 7 sec ago	0 0
13	Test 13 jr@debian.localdomain	new 1 min ago	General	Nobody 1 min ago	0 0

Page 1 of 1

Don't refresh this page.

Update multiple tickets  
spreadsheet | Bookmarkable link | RSS | Work offline

chart grouped by Status style: bar

In addition to generating a public status display, I was asked to provide a reporting facility to make it easy to supply progress reports to site management meetings. The aforementioned spreadsheet export is perfect for this.

You simply run an appropriate search, then hit the "spreadsheet" link.





# Spreadsheet output

The screenshot shows a Microsoft Excel spreadsheet with the following data:

id	Queue	Subject	Status	Owner	Requestors	Created	Resolved	LastUpdated	CF-Room Number
12	General	Email with Room Number	open	root	jr@debian.localdomain	13/07/09 15:02		14/07/09 09:49	10-026
13	General	Test 13	new	Nobody	jr@debian.localdomain	14/07/09 09:47		14/07/09 09:47	50-023

No description!



The end result even includes the custom field content.

Unfortunately, it doesn't include the description as this is merely part of the email body text and doesn't make it into an exported field.

This is a problem as it contains much of the most useful information.

## Custom Field for Description

- `Description|Body|Description:\s*(?s)(.+)|q`
- This field has to come last as it captures the rest of the email, allowing multi-line descriptions to be included
- Note the use of `(?s)` to turn on “single line mode” half way through the expression.  
See <http://www.regular-expressions.info/modifiers.html>



The solution is, of course, to capture the description as a custom field.

This is a rather more complex job as the description is typically a multi-line text block.

Fortunately “single line mode” allows the Regular Expression to match beyond the end of line marker. You’ll need to add “Description: “ before the Description field in the email.

You also need to make sure the description appears at the end of the message as the multi-line matching regular expression will gobble up the rest of the email.

# Summary

1. We converted RT from an email to a web-based ticket handling system, from the requester's perspective
2. We used Webauth and OAK LDAP to ensure the requester's name and email are correct
3. We provided a reporting mechanism that includes all details about each job



# Thanks

- Ashley – for persuading me to write this talk
- Tony and the rest of the team – for putting together yet another excellent conference
- John Elder – for doing the RT installation
- Sysdev – for Webauth and OAK LDAP
- You lot – for listening



# Questions?

- If we have time ...
- Otherwise, these slides are on-line at [http://www.ictf.ox.ac.uk/conference/2009/presentations/wks18\\_rtcust.pdf](http://www.ictf.ox.ac.uk/conference/2009/presentations/wks18_rtcust.pdf) and the scripts can be found at <https://wiki.oucs.ox.ac.uk/itss/CustomisingRequestTracker>

Email: [jeremy.rowntree@bioch.ox.ac.uk](mailto:jeremy.rowntree@bioch.ox.ac.uk)



The questions mostly focussed on the ability of NMS FormMail to keep the destination email address hidden from spammers.

Any further questions about this presentation can be emailed to [jeremy.rowntree@bioch.ox.ac.uk](mailto:jeremy.rowntree@bioch.ox.ac.uk)