API and CLI

API (REST)

RT uses the REST interface to give external script access to the RT database.

The base URL for all requests is: https://ithelp.uoregon.edu/REST/1.0/

For a quick test, this will show a summary of ticket #340 (replace 340 with a ticket you have permission to see)

https://ithelp.uoregon.edu/REST/1.0/ticket/340/show

Libraries

There are libraries which do much of the low-level work

Perl - RT::Client::REST
Ruby - rt-client

Retrieving Tickets:

The URL you will use will look something like:

https://ithelp.uoregon.edu/REST/1.0/ticket/$ticketnumber

PERL Example (show ticket #1):

```perl
#!/usr/bin/perl
use LWP::UserAgent;

$uri='https://ithelp-test.uoregon.edu/REST/1.0/';

$ticketNumber=1;

my $ua = LWP::UserAgent->new;
$ua->timeout(10);

my $response = $ua->post( $uri . "ticket/$ticketNumber",
[ 'user' => 'user',
  'pass' => 'pass',
  'Content_Type' => 'form-data'
];

if ($response->is_success) {
  print $response->decoded_content;
} else {
  print "ERROR code: ", $response->code, ", msg: ", $response->message , "\n";
}
```

More URL examples: This will search for outstanding IP requests:

https://ithelp.uoregon.edu/REST/1.0/search/ticket?query=%28%20Status%20%3D%20%27new%27%20%20OR%20%20Status%20%3D%20%27open%27%20%20AND%20%20Subject%20LIKE%20%27IP%20Req%27%20%29

Resolve ticket #5 (be careful): https://rt-beta/rt3/REST/1.0/ticket/5/edit?content=Status:%20resolved
Internal scripts via cron:

This is used by several scripts to modify tickets on the server. These scripts must be run on the server with privileges.

See: escalating tickets and mailing reminders for examples.

CLI (Command Line Interface)

The CLI allows you to manipulate tickets on a UNIX command line. Very useful for automation. Be sure to never do this on a multi-user machine (such as shell.uoregon.edu) as other users may have access to passwords/etc. The CLI will create a secure web connection to the server.

Setting up a working environment for RT CLI

Before you can start using the RT CLI you have to set up a working environment for your RT, here we have two possible ways:

Environment Variables

Setting environment variables is a good way to start using the CLI. The following variables could be defined: RTUSER
RTPASSWD
RTSERVER
RTDEBUG
RTQUERY
RTORDERBY
RTCONFIG
EDITOR or VISUAL

You can set the important variables inside a BASH in this way: export RTUSER=your.rtusername
export RTPASSWD=your rt password
export RTSERVER=https://ithelp.uoregon.edu

RT Config File

Create at the users Home Directory a .rtrc file with the following min. Options: server https://ithelp.uoregon.edu
user your.rtusername
passwd your rt password (It should be different from your DuckID password for obvious reasons).

Advanced settings can also be defined here: query your default query string ("Status = 'new' and Priority > 5")
orderby your default sort order

To get the binary download and untar the latest RT, then look for /bin/rt, and copy it to your path.

Working with the CLI

$ ./rt
rt>

This will bring you directly to the rt cli shell. Now lets try us to getting help:jfe@crash ~ $ ./rt

rt> help
** THIS IS AN UNSUPPORTED PREVIEW RELEASE **
** PLEASE REPORT BUGS TO rt-bugs@bestpractical.com **

This is a command-line interface to RT 3.0 or newer

It allows you to interact with an RT server over HTTP, and offers an interface to RT's functionality that is better-suited to automation and integration with other tools.

In general, each invocation of this program should specify an action to perform on one or more objects, and any other arguments required to complete the desired action.

For more information:
- rt help usage (syntax information)
- rt help objects (how to specify objects)
The normal usage looks like this: rt <action> options arguments

The RT Actions:

You can currently perform the following actions on all objects:

- list (list objects matching some condition)
- show (display object details)
- edit (edit object details)
- create (create a new object)

The RT Objects:

Every object in RT has a type (e.g. "ticket", "queue") and a numeric ID. Some types of objects can also be identified by name (like users and queues). Furthermore, objects may have named attributes (such as "ticket/1/history").

An object specification is like a path in a virtual filesystem, with object types as top-level Directories, object IDs as subdirectories, and named attributes as further subdirectories.

A comma-separated list of names, numeric IDs, or numeric ranges can be used to specify more than one object of the same type. Note that the list must be a single argument (i.e., no spaces). For example, "user/root,1-3,5,7-10,ams" is a list of ten users; the same list can also be written as "user/ams,root,1,2,3,5,7,8-10".

The RT Types:

You can currently operate on the following types of objects:

- tickets
- users
- groups
- queues

Working with Links via the CLI

Syntax: rt link -d <id-A> <link> <id-B>

Creates (or, with -d, deletes) a link between the specified tickets. See 'rt help link' for more details.

Examples

To Make a new user: rt create -t users add Name=userName EmailAddress=email@uoregon.edu
To get a users information :rt show user/userName
To create a ticket in Queue Quux: rt create -t ticket set subject='New ticket' priority=10 add cc=foo@uoregon.edu queue=Quux
To show summery for a specific ticket: rt show ticket/3
To show extended information a specific ticket: rt show -l ticket/3
To show the history of a specific ticket: rt show ticket/3/history
To show specific fields for a ticket:rt show -t ticket 42 -f id,subject,status,queue,owner,priority
To make a report of tickets in a queue: not resolved, that are 30 days old: rt ls -o -t ticket "Queue = '4' AND Status != 'resolved' AND \
Created < '30 days ago'" -f id,subject,status,created,lastupdated,owner \

Creating tickets with multiple lines of text

To create a ticket with multiple lines of text in the body indent the lines with a space. Example: rt create -t ticket set queue=myqueue subject="MultLine Test" text = "Line 1 Line 2 Line 3"