IRRd - Internet Routing Registry Daemon
User/Configuration Guide

Merit Network, Inc.

IRRd - Internet Routing Registry Daemon User/Configuration Guide
by Merit Network, Inc.
IRRd User/Configuration Guide Edition
Published September 2001

Revision History

Revision 2.1 2001-09-17 Revised by: ljb
Initial release of SGML version of document.
Revision 2.1.2 2001-09-24 Revised by: ljb
fixes/additions for documentation on irr_rpsl_submit.

Table of Contents

Chapter 1. Introduction

IRRd is a stand-alone Internet Routing Registry database server. IRRd can store information and answer queries about local network, campus and ISP backbone topology, address allocation and routing policies. IRRd can be used as an independent local database server, or as part of the global Internet Routing Registry (IRR). The Internet Routing Registry is the union of a growing number of world-wide routing policy databases, including servers operated by Cable & Wireless, APNIC, Merit, Bell Canada, RIPE, Verio, and many other organizations. See http://www.irr.net/docs/list.html for an up-to-date list of registries.

IRRd supports both RIPE-181 and Routing Registry Specification Language (RPSL) routing registry syntaxes. The IRRd distribution includes all needed IRR support services, including: automated real-time mirroring of other IRR databases, update syntax checking, update security checking, and update notification. The current version of IRRd also supports most RIPEdb whois flags.

When used in conjunction with policy tools such as RtConfig, Roe, and Aoe, the IRRd server allows:

- Automated generation of router configuration files and access-lists
- Internet topology visualization
- Network trouble-shooting and debugging

In addition to the usual RIPE/RPSL whois queries, the IRRd Server also provides a protocol for getting information from RIPE/RPSL-style database files that is not easily (or rapidly) obtainable using the standard whois queries. These IRRd queries can be submitted one-at-a-time via whois, or by establishing a connection to the IRRd Server, issuing multiple queries, and then closing the connection. The IRRd distribution also includes the irr_rpsl_submit e-mail/TCP database update tool.

Document Conventions

The following document conventions are used in the IRRd User/Configuration Guide:

- Commands and keywords are in **boldface**.
- User-supplied variables are enclosed in <angle brackets>.
- Optional elements are shown in [square brackets].
- Alternative but required keywords are grouped in {braces} and separated by a vertical bar.

Getting Help

Merit hosts a public mailing list for IRRd questions, advice, and feature suggestions at irrd-discuss@merit.edu. List administravia requests should be directed to irrd-discuss-request@merit.edu.

Credits

The IRRd development team includes: Larry Blunk, Dale Fay, and Christerfer Frazier at Merit. Project alumni include Gerald Winters, Susan Harris, Craig Labovitz, Jon Poland and Matt Lewinsky.

IRRd RPSL support was developed with funding from the National Science Foundation (NCR-9321060). Commercial funding from Merit Network, Inc. supported all other IRRd development.

Portions of the IRRd code are based on software libraries from the MRT toolkit. MRT was originally developed by Merit Network, Inc., under National Science Foundation grant NCR-9318902, "Experimentation with Routing Technology to be Used for Inter-Domain Routing in the Internet." Additional MRT research was supported by the National Science Foundation (NCR-9710176) and gifts from Microsoft and the Intel Corporation. The design and ideas behind many of the MRT libraries draw heavily on the architecture pioneered in the GateD routing daemon.

A large number of bug reports and fixes were provided by IRRd beta testers, including: Kevin Oberman (ESNet), Mark Prior (ConnectNet), John Heasly (Verio), and George Matey (Bay Networks).

Chapter 2. Obtaining IRRd

The IRRd source code may be freely modified and redistributed so long as the University of Michigan copyright notice is included with the redistribution. The latest IRRd source distribution can be found at the following website:

http://www.irrd.net

System Requirements

IRRd will run on most Unix operating systems. But for best performance, we recommend IRRd be used on a threads-capable operating systems, such as Linux or Solairs. For optimal performance, we also recommend IRRd be used in memory-only mode (-m on the command line). This assumes one has sufficient system memory to hold the database indexes.

IRRd is generally I/O bound and does not require significant CPU cycles (with the exception of the period during infrequent database reloads).

IRRd has been compiled and tested on various Unix platforms. The target platforms for development are Linux, Solaris, and FreeBSD.

Building and Installation Procedure

1. Obtain the source archive and then unzip and untar it.

```
% cd /tmp
% ftp ftp.merit.edu
ftp> cd radb/irrd/source
ftp> get irrd.tar.gz
ftp> quit
% cd /usr/local/src
% gzip -cd < /tmp/irrd.tar.gz | tar xvf -
```

2. Change into the irrd directory and run **./configure**.

```
% cd /tmp/irrd-<version>/src
% ./configure
```

Several options can be specified as parameters to the configure command:

```
% ./configure -disable-ipv6      Disable IPv6 connection support
% ./configure -disable-threads   Disable thread support
% ./configure -disable-wall      Disable -Wall gcc option
% ./configure -with-gdbm         Support GNU DBM database files
% ./configure -with-pgpv5       Prefer PGP V5 over default of GnuPG.
```

3. Now run make. After successful compilation, you will have binaries in each of the program directories..

```
% make
```

4. Run make install. By default, binaries will be installed in /usr/local/bin. Use the –prefix option with configure to change the default install directory.

```
% su
# make install
```

5. See Section 5 for information on configuring email and TCP object submission.

Chapter 3. Using IRRd

Getting Started

Before using IRRd, you will need to obtain an initial copy of the IRR databases. The IRRd distribution includes a tool, IRRdCacher, which can be used to fetch and maintain copies of the IRRd databases. The tool and instructions are available as part of the IRRd distribution.

See Appendix A for more information about IRRdCacher.

By default, IRRd expects to find copies of the databases (ans.db, cw.db, radb.db, etc.) in /var/spool/databases. This default IRR directory may be overridden with the -d command line flag, or the irr_directory config file entry.

For real-time mirroring, you will need to contact the database administrators to obtain the appropriate IP address and port number used for mirroring service.

By default, IRRd listens for queries on the standard whois TCP port 43. Alternatively, the whois port number may be specified by the irr_port configuration command.

In addition, IRRd listens for user configuration/management telnet connections by default on TCP port 5673. You can optionally specify the port on which the server listens for telnet connections by adding the following line to /etc/services. Feel free to choose your own port numbers.

```
irrd     5674/tcp    # IRRd routing registry server
```

The daemon may be configured by editing a configuration file, or by invoking the configuration utility from the interactive user telnet interface. The interactive interface features a Cisco System. Below is an example of telneting to the user interactive interface (UII) port on a machine running IRRd.

```
>telnet 127.0.0.1 irrd
IRRd version 2.1.0 [09/13/2001]
User Access Verification
foo password: *****
foo IRRd#
```

If a password is specified in the configuration file, it must be supplied at the password prompt. Initially, IRRd defaults to no password access control and restricts user interactive telnet to the loopback address or the interface address of the local machine.

IRRd Synopsis

irrd [-a] [-d *database dir*] [-f *conf file*] [-g *group name*] [-l *user name*] [-m] [-n] [-r] [-s *password*] [-u] [-v] [-w *irr port*] [-x]

Options

```
-a                  Enable atomic transactions for database updates
-d <path>           Set database directory
-f <conf file>      Specify the configuration file to use (de-
                    fault: /etc/irrd.conf)

-g <group name>     Drop priveleges to given group name
-l <user name>      Drop priveleges to given user name
-m                  Memory only mode
-n                  Do not daemonize
-r                  Force re-indexing of databases
-s <password>       Set the UII password
-u                  Don't allow privileged commands
-v                  Verbose logging, debug mode
```

Description

IRRd is a complete Internet Routing Registry Server supporting indexing, mirroring, whois queries, and email/TCP updates. Interactive telnet connections are on port "irrd" in /etc/services.

Interactive Interface

IRRd provides an interactive user interface that can be used to control various and operational aspects of IRRd and show the current status of the daemon.

The port number can be specified in the configuration file. The default is TCP port 5673, or the number associated with "irrd" in /etc/services. If a password is specified in the configuration file, it must be supplied on login.

Unix shell-like redirection (or filename) is available for output. To edit a line, emacs-like line editing including ^a, ^b, ^e, ^f, ^d, ^k, ^u and ^c is available. To reuse a previous line, tcsh-line history function is available by typing ^p and ^n.

The IRRd command language shares many similarities with the language used on Cisco Systems routers. Commands include:

- **show config** – view the configuration file
- **show version** – show the current version
- **show threads** – show the status of application threads
- **show connections** – show current TCP tool queries
- **reboot** – restart the daemon
- **help** – shows all commands available
- **exit** – leave the UII interface

- **mirror** – synchronize database with remote server
- **reload** – reload an IRR database file
- **show database** – show database status
- **dbclean** – synchronize IRR diskfiles with memory

Below is an example of a user interactive telnet command to the IRRd daemon:

```
[47] IRRd> show databases
 Database    Size (kb)    Rt Obj   AutNum Obj   Serial #
 -----    ------    ----   -------   -----
  cw           6722.3      40076      435        0
  radb        10257.5      42913     1083        19889
  ans         58654.5       9067       24        6498
  ripe         3823.7      16854     1461        1312991
  canet        1027.3       9073       58        0

cw mirroring whois.radb.net
radb mirroring whois.radb.net
ans mirroring whois.radb.net
ripe mirroring whois.radb.net
canet mirroring whois.radb.net
```

Getting Status Information from IRRd

show mirror-status – shows the status of mirroring a remote repository.

This command makes it possible to determine whether your repository and the remote repository are synchronized. Use of the show mirror-status command requires that both repositories support the "!j" functionality."

```
For example:
show mirror-status telstra
telstra (Mirror)

Local Information:
  Oldest journal serial number: 31
  Current serial number: 1810
Remote Information:
  Mirror host: 203.50.0.201:43
  Mirrorable.
  Oldest journal serial number: 1.
  Current serial number: 1810.
  Last exported at serial number: 1810.
```

The local information shows the oldest serial number in your local journal (for providing third-party mirroring of the remote repository) and the current serial number in your local repository. The remote information shows the status of the repository, where it is mirrored, its oldest journal number for mirroring, and the database's current serial number. In the above example, the local current serial number and the remote current serial number are the same, and thus the databases are in sync.

When a repository that doesn't support !j functionality, such as the RIPE server, is queried, output such as the following is returned:

```
ripe (Mirror)
Local Information:
  Oldest journal serial number: 13037243
```

```
  Current serial number: 13051817
Remote Information:
Mirror host: 193.0.0.200:43
  Remote status query unsupported.
```

show statusfile – shows location of the IRRD_STATUS file.

IRRd version 2.0 makes it possible to store additional state information for remote repositories. This data is used for responses to the show mirror-status command and other queries. By default, the `IRRD_STATUS` file is stored in your IRRd configuration directory and is called `IRRD_STATUS`. You can use the set statusfile command to set a different location.

set statusfile – sets location of the `IRRD_STATUS` file.

Configuration Commands

When IRRd is started for the first time and no configuration file exists on disk, the programs will create a default configuration in volatile memory. This configuration may be modified in memory by issuing the "config" command from the UII telnet interface prompt. Modifications to volatile memory may be saved to disk using the "write" command. Modifications not saved to disk will be lost if the daemon terminates or is rebooted.

Upon startup, IRRd will search for the default configuration file for the daemon (usually /etc/<irrd.conf>). The user may also override the default configuration file by providing a "-f <filename>" flag on the command line of the daemon.

IRRd supports the following configuration commands:

password <**string>** <**access-list number>**

Sets a password <string> for the telnet interface. Note that if a password is not set, access verification will not be performed and interactive user telnet connections will only be allowed from the localhost. If <access-list number> is specified, telnet connections will be restricted to IP addresses allowed by the access list. See the access-list description below for more information.

uii port <**number>**

Changes the port number with <number> for the telnet interface. The default is the value specified in /etc/services for the daemon name "irrd". If an /etc/services entry does not exist, the port number defaults to 5673.

debug <**server**|**submission> file-name** <**filename>**

debug <**server**|**submission> file-max-size** <**bytes>**

debug <**server**|**submission> syslog**

debug <**server**|**submission> verbose**

Turns on logging for the IRRd server or object submission by the email/tcp irr_rpsl_submit process. file-name specifies the disk file, or "stdout." file-max-size bytes automatically truncates the log file at <size> byes. Configuring syslog sends logging information to syslog on the local machine. Verbose enables verbose logging.

access-list <**number> {permit**|**deny}** <**prefix> [refine**|**exact]**

Defines an access list <number>, which permits or denies access if the condition is matched. all can be specified as <prefix>. exact will be assumed if neither refine or exact is specified. exact matches only the prefix, while refine matches more specific prefixes, excluding the prefix itself.

Matches are performed in the order in which they appear. At the end of a list with the same number, permit all is assumed.

<number> must be an integer between 1 and 1000.

For example:

```
! Access only from Merit Nets
access-list 1 permit 198.108.60.0/24 refine
access-list 1 permit 198.108.0.0/24 refine
access-list 99 deny all
```

! – comment and separator

Comments can appear at the beginning of a line, or any other place in the line. A comment at the beginning of a line is treated as a separator, which ends a command clause followed by its sub-commands.

redirect <directory>

Allows redirection to files in this directory. Unrestricted redirection was deemed a security problem.

For example, in a configuration file:

```
redirect /tmp
```

At the UII prompt)

```
> show database > /tmp/ouput
```

irr_directory <path>

Specify the path for the cache directory and database files.

ftp directory <path>

Specify the directory in which to copy files for ftp access. Also see the irr_database export command below.

irr_database <name> [mirror_host <hostname> [port <port number>]]
irr_database <name> [authoritative]
irr_database <name> [access <num>]
irr_database <name> [write-access <num>]
irr_database <name> [mirror-access <num>]
irr_database <name> [filter [non-critical | routing-registry-objects | <object name>]
| [~(non-critical | routing-registry-objects | <object name>)]
irr_database <name> [export <export interval>]

Include a database named <name>.db in the IRR directory in the list of databases provided by the server. If available, enable automatic mirroring to hostname on the selected port (default is 43).

If the authoritative keyword is used, updates will be allowed for this database.

Access restricts read, write and mirror access to IP addresses permitted by access-list <num>.

Write access refines access and limit updates to IP addresses permitted by access-list <num>. By default, write access is restricted to the loopback address of the local machine. Mirror-access refines access and limits database mirroring to IP addresses permitted by access-list <num>.

Some databases (like RIPE) contain a significant volume of non-routing related information like person objects and role objects. To reduce the size of the database, you can use the filter command to specify the objects you want to include (or not include) in your database.

The export option will atomically copy the database into the ftp_dir directory for exporting.

irr_mirror_interval <seconds>

The interval for obtaining mirror updates. The default is 10 minutes.

irr_port <port>

The port to listen on for "RAWhoisd" style machine TCP connections

irr_max_connections <number>

Limit the number of simultaneous queries. The default is 25 connections.

dbclean [interval <number of seconds>]

Sychronize the disk database files with IRRd memory. During normal operation, IRRd marks updated or deleted objects with a special flag. By default, IRRD rebuilds the database.db (without these deleted objects) once every 24 hours.

no dbclean

Disable database cleaning.

Configuration Commands for irr_rpsl_submit

The following configuration commands control the behavior of the IRRd submission module:

irr_server <host>

The IRRd IP network address of the remote IRR server. Defaults to "localhost".

irr_port <port>

The IRRd command/query port. The default is 43.

override_cryptpw <password>

The encrypted password used for overriding normal authentication checks. The default is "piSFDzJu5e1wY" (i.e., foo).

pgp_dir <path>

The directory path of the PGP ring files. The default is ~/.pgp

db_admin <email address>

The email address of the DB administrators. The default is db-admin@localhost.

Database submissions will be sent to 'irr_port' and 'irr_server'. The defaults are host 'localhost' and port 43. The 'override_cryptpw' command sets the system password and is typically used by an administrator to enter new maintainer objects into the system. Two log files are created: a submission log 'trans.log' and an acknowledgement log 'ack.log'. The log files will be created by default in the directory specified by 'irr_directory' (i.e., the IRRd cache directory). The default can be overridden with the 'submission_log_dir' configuration command.

The 'pgp_dir' configuration command specifies the directory location of the public and secret PGP ring files. The default is the normal PGP default, ~/.pgp from the UID of the invoking process. The 'pgp_dir' command is useful, for example, when sendmail invokes the DB submission process from /etc/aliases running under user

'daemon'. Note that the PGP directory will need to be permitted properly to allow access from the irr_rpsl_submit process. Alternatively, one may want to install the irr_rpsl_submit binary as set-uid to a particular ID which has access to the PGP directory.

Example 3-1. Sample Configuration File

After editing the configuration file, the user may return to the top-level of the interactive telnet interface by typing a ^Z or entering exit. Here is an example of an IRRd confguration file:

```
!
! Test config file
!
password xxxxxx
uii_port 5673
!
! The cache directory
irr_directory /var/irr/databases/
debug server file-name /var/spool/log/irrd.log
debug submission file-name /var/spool/log/irr-email.log
!
! The port of whois and RAToolset connections
irr_port 43
!
! Make sure we don't get overwhelmed
irr_max_connections 25
!
irr_mirror_interval 1800
irr_database radb mirror_host whois.radb.net
irr_database ripe mirror_host whois.radb.net
irr_database canet
irr_database ans mirror_host whois.radb.net
irr_database cw
irr_database localdb authoritative
irr_database localdb access 1
!
db_admin db-admin@merit.edu
override_cryptpw EhjhsdhEhjhsd
pgp_dir /irr/etc/.pgp
irr_server whois.radb.net
!
! Access only from Merit Nets
access-list 1 permit 198.108.60.0/24 refine
access-list 1 permit 198.108.0.0/24 refine
access-list 99 deny all
```

Chapter 4. Querying IRRd

RADB-style machine telnet queries are available on the port specified in the configuration file. Although IRRd was designed for use by tools such as RtConfig, peval, and PRtraceroute, it is also useful for compute-intensive queries generated by individuals.

IRRd supports two modes: single command mode and multiple command mode.

Single command mode – the query server processes one command, returns the results to the server, and closes the connection. This is the default mode for IRRd, and the normal operation of a whois server.

Multiple command mode – the query server continues to accept and service query requests on the connection until the remote user issues a quit command. See Appendix B for more information about IRRd's multiple-command mode.

Example 4-1. Whois Queries

IRRd also supports standard RPSL whois queries. For example:

whois -h whois.radb.net 128.223.0.0/16

The output is an IRR route object:

```
descr:          UONet
                University of Oregon
                Computing Center
                Eugene, OR 97403-1212
                USA
origin:         AS3582
mnt-by:         MAINT-AS3582
changed:        meyer@ns.uoregon.edu 19960222
changed:        auto-dbm@ISI.EDU 19990729
source:         RADB
```

Chapter 5. Updating IRRd with irr_rpsl_submit

By default, IRRd receives updates through automated real-time mirroring of other IRR database servers. If you want to do more than run local server caching, i.e., if you want to run your own component of the IRR, you will need to set up e-mail and TCP submission for IRRd.

The irr_rpsl_submit module is configured by command line flag values, by setting configuration commands in the IRRd configuration file, or by a combination of both. Command line options override options set in the IRRd configuration file.

irr_rpsl_submit Synopsis

irr_rpsl_submit [-c *encrypted system password*] [-D] [-E *DB admin address*] [-f *IRRd conf file*] [-F *response footer string*] [-h *irrd host*] [-l *log directory*] [-p *irrd port*] [-r *pgp directory*] [-R] [-s *authoritative DB source*] [-v] [-x] [filename]

Options

-c <password>	Encrypted password that overrides normal authentication checks. The encrypted password is ciphertext generated using the crypt(3) utility. We also helpfully provide the crypt_gen.c file in the programs/irr_util directory.
-D	Read input from STDIN for direct/TCP (non-email) submissions
-E <DB email>	Email address for DB admin mail
-f <config file>	IRRd configuration file location
-F <footer string>	enclosed response footer string to add to messages
-h <server>	IRRd host/server
-l <log dir>	Log directory location
-p <port>	IRRd port number
-r <pgp dir>	PGP ring files location
-R	RPS Dist mode
-s <db name>	Specify authoritative database source
-v	Turn on verbose debugging/logging.
-x	Suppress notifications. The database will be updated but notifications will not be sent.
filename	Input filename

Description

irr_rpsl_submit accepts e-mail updates and controls the process of entering and modifying database data. irr_rpsl_submit can perform PGP authentication, the standard authentication mechanisms of encrypted password and mail-from, syntax checking, and standard RIPE/RPSL notifications.

The '-f' and '-p' options specify the IRRd daemon location. The defaults are localhost and port 43. The '-c' option specifies the system encrypted password used to override normal authentication checking. The default is "piSFDzJu5e1wY" (i.e., foo). '-D' causes irr_rpsl_submit to read from STDIN and disables mail feedback (for use in direct/TCP-based submissions). '-f' specifies the IRRd configuration file location. /etc/irrd.conf is the default. '-l' specifies the location for the acknowledgement and transaction logs. The default is the 'irr_directory' value from /etc/irrd.conf. '-r' gives the PGP ring files location. The default is ~/.pgp in the user's home directory. '-s' specifies authoritative databases. irr_rpsl_submit will only allow updates to authoritative databases and will signal an error for all others. The '-s' option may appear multiple times as necessary. '-x' stops notifications from being sent. 'filename' is the name of the input file.

The irr_rpsl_submit flag options override options in the IRRd configuration file. These options enable irr_rpsl_submit to reside on a remote machine from IRRd and to operate without an IRRd configuration file.

Configuration Commands

When an IRRd submission instance is invoked, the default IRRd configuration file is scanned for configuration information (/etc/irrd.conf). The default configuration path can be overridden with the "-f <filename>" command line option.

For a listing of configuration commands that control the behavior of the IRRd submission module, see Section 3.

Invoking irr_rpsl_submit

Many users will find it convenient to register irr_rpsl_submit in their Unix /etc/aliases file to allow convenient remote mail access. Note that a link to irr_rpsl_submit in /etc/smrsh will be needed for those systems which employ the Sendmail restricted shell (i.e. RedHat Linux). Also note that if PGP support is enabled, the irr_rspl_submit process will need to be able to read/write the PGP directory configured by pgp_dir. This can be achieved by installing the irr_rpsl_submit process as set-uid to a particular ID and permit read/write access to the PGP directory from that ID. Below is an example /etc/aliases entry for use with sendmail:

auto-dbm: "|/usr/local/bin/irr_rpsl_submit -f /etc/conf/irrd.conf"

The '-f' option gives the location of the IRRd configuration file. Any of the other flag options listed above can also be used.

Some additional utilities are bundled with the IRRd distribution, including mailloopbreak.pl which can be used in from of irr_rpsl_submit in the /etc/alias file to detect, and prevent mail loops from the irr-submit auto-generated email messages.

System Requirements

The irr_rpsl_submit module is a non-threaded application and is part of the IRRd software distribution. GunPG or PGP 5.0 (or higher) must be installed if you want irr_rpsl_submit to support PGP authentication.

irr_rpsl_submit Quick Start

Here are step-by-step instructions for obtaining and installing irr_rpsl_submit to handle submissions:

1. Grab and build an IRRd source code distribution: See Section 2 for these steps.
2. Initialize the /etc/irrd.conf file by adding the following entries:

   override_cryptpw EWUZmlvOSvHmk ! sets system password to "foo"
   pgp_dir <your ~/.pgp path> ! example: /usr/users/joe/.pgp

3. Execute IRRd.

   (See Section 2 for instructions on building IRRd)
   % /usr/local/bin/irrd -m

4. To allow email submissions, add the following to /etc/aliases (also add a link to irr_rpsl_submit in /etc/smrsh if using the Sendmail restricted shell and verify that the pgp_dir is permitted to allow read/write access to the process):

   auto-dbm: "|/usr/local/bin/irr_rpsl_submit"

5. To allow TCP submissions, add the following to /etc/services:

   irr_rpsl_submit 8888/tcp

6. Add the following to your /etc/inetd.conf and restart inetd:

   irr_rpsl_submit stream tcp nowait daemon /usr/local/bin/irr_rpsl_submit irr_rpsl_submit -D

Chapter 6. Submitting Email Updates

This is a brief explanation of how to send email updates to an IRRd server (via irr_rpsl_submit). It is intended as a sort of 'quick start' document, detailing only the minimum steps necessary to register. It is assumed that you are somewhat familiar with RIPE/RPSL-style routing registries and RPSL routing policy syntax.

Updates and Changes

This document provides templates for registering three types of objects in the RADB:

Object	Contents
Maintainer object	Individuals who can update your database entries
AS object	Administration and routing policy of an AS
Route object	A single route to be added to the registry

Step One - Register One or More Maintainers

Maintainer objects specify which parties are allowed to perform updates to the RADB, and how these parties are authenticated. When a route or AS object is submitted for registration, a Maintainer object must be referenced; otherwise the submission will be rejected. Thus, the first step to registering information in the IRR is to register one or more Maintainer objects. To do this, first determine the names and email addresses of those who will be allowed to update AS and Route objects. Then copy the maintainer object template below into an email message, filling in the fields with the appropriate information, and send it to database administrators (usually db-admin@<domain>). A human will read this message and add the information to the registry.

Maintainer objects need mnt-by attributes just like any other object. You should make sure that the maintainer objects you register contain a mnt-by attribute and its value should be the value of the mntner attribute. This self-reference specifies that updates to this maintainer object are allowed only from those authorization mechanisms specified in the maintainer object. Failure to register a maintainer object in this way means that anyone could modify that maintainer and subsequently modify the objects it references.

Example 6-1. Maintainer Template

```
————————————- CUT HERE ————————————-
mntner:
descr:
admin-c:
tech-c:
upd-to:
mnt-nfy:
mnt-by:
auth:
changed:
source:
————————————- CUT HERE ————————————-
```

Example 6-2. Maintainer Example

```
mntner:      MAINT-AS237
descr:       Maintainer for AS 237
admin-c:     Andrew L. Adams
tech-c:      Andrew L. Adams
upd-to:      ala@merit.edu
mnt-nfy:     ala@merit.edu
mnt-by:      MAINT-AS237
auth:        MAIL-FROM ala@merit.edu
auth:        MAIL-FROM dsj@merit.edu
changed:     ala@merit.edu 941219
source:      RADB
```

NOTE: Send only Maintainer objects to db-admin@<domain>. AS, Route and other objects must be sent to auto-dbm@<domain>. Of all the objects, only Maintainer objects undergo a human check before being committed to the registry and therefore, as might be expected, registration of Maintainer objects takes longer than registration of AS and Route objects. Turnaround time on Maintainer objects is on the order of hours rather than seconds, as in the case of other objects.

Once the Maintainer object is created, modifications can be sent to auto-dbm@radb.net. The new object will automatically replace the old one.

Step Two - Register AS and Policy Information

After registering a Maintainer object, the next step is to register an AS object, thereby specifying an AS's routing policy. Because AS objects are referenced by Route objects, they must be registered before Route objects. To do this, first determine how to express the AS policy in RPSL syntax. Then copy the AS object template below into an email message, filling in the fields with the appropriate information, and send it to auto-dbm@<database>. This message will immediately be checked for proper syntax and some semantic checks will be performed. If errors are detected, it will be returned to you with annotations describing the errors. Otherwise, the AS object will be added to the registry.

Note that the mnt-by field should contain the string you submitted in the mntnr field of the Maintainer object.

Example 6-3. AS Template

```
--------------- CUT HERE -------------
aut-num:
descr:
import:
export:
default:
admin-c:
tech-c:
remarks:
remarks:
mnt-by:
changed:
source:      RADB
--------------- CUT HERE -------------
```

Example 6-4. AS Example

```
aut-num:        AS3582
as-name:        UONET
descr:          University of Oregon
import:         from AS689
                action pref=10;
                accept NOT ANY
import:         from AS1798
                action pref=10;
                accept AS1798 AND NOT {0.0.0.0/0}
import:         from AS2914
                action pref=10;
                accept <^AS-WNA*$> AND NOT {0.0.0.0/0}
import:         from AS3701
                action pref=10;
                accept ANY AND NOT {0.0.0.0/0}
import:         from AS3838
                action pref=10;
                accept AS-SNS AND NOT {0.0.0.0/0}
import:         from AS4222
                action pref=10;
                accept <^AS-LEN*$> AND NOT {0.0.0.0/0}
import:         from AS5650
                action pref=10;
                accept AS-ELICUST AND NOT {0.0.0.0/0}
import:         from AS6447
                action pref=10;
                accept <^AS-OREGON-IX*$> AND NOT {0.0.0.0/0}
import:         from AS10876
                action pref=10;
                accept <^AS-MAOZ*$> AND NOT {0.0.0.0/0}
export:         to AS689
                announce AS3582
export:         to AS1798
                announce AS3582
export:         to AS2914
                announce AS3582
export:         to AS3701
                announce AS3582
export:         to AS3838
                announce AS3582
export:         to AS4222
                announce AS3582
export:         to AS6447
                announce AS3582
export:         to AS5650
                announce AS3582
export:         to AS10876
                announce AS3582
admin-c:        DMM65
tech-c:         DMM65
notify:         nethelp@ns.uoregon.edu
mnt-by:         MAINT-AS3582
changed:        meyer@antc.uoregon.edu 19980128
source:         RADB
```

(This example is taken from RFC 2650, "Using RPSL in Practice.")

Step Three - Register Routes

After registering Maintainer and AS objects, the next step is to register Route objects. To register a Route object, copy the Route object template below into an email message, filling in the fields with the appropriate information, and send it to auto-dbm@<domain>. This message will immediately be checked for proper syntax, and some semantic checks will be performed. If errors are detected, it will be returned to you with annotations describing the errors. Otherwise, the Route object will be added to the registry.

```
--------------- CUT HERE -------------
route:
descr:
origin:
remarks:
notify:
mnt-by:
changed:
source:
--------------- CUT HERE -------------
```

Example 6-5. Route Example

```
route:       35.0.0.0/8
descr:       Merit Network, Inc.
descr:       University of Michigan
descr:       4251 Plymouth Road, Suite C
descr:       Ann Arbor
descr:       MI 48105-2785, USA
origin:      AS237
mnt-by:      MAINT-AS237
changed:     radb-admin@merit.edu 981113
source:      RADB
```

Deletions

ISPs and network operators will often need to delete Route objects, AS objects, or Maintainer objects from the IRR. You'll need to delete a Route object (and submit a new one), for example, if:

- You are changing ISPs and need to change the origin AS (Home AS) for your route
- You are no longer using a particular route prefix
- You are moving to a larger aggregate and want to delete a more specific prefix

If you want to change any of the following attributes, you'll first need to delete the object, and then re-submit the corrected one:

```
Object                  Field
---                   ---
```

```
Route                   route:
                        origin:
AS                      aut-num:
Maintainer              mntner:
```

Follow these instructions to delete an object from the IRR:

1. Use the whois tool to get a copy of the object exactly as it currently exists in the registry. Put the object in a temporary file. For example:

   whois -h whois.radb.net 10.1.2.0/24 > temp

2. Edit the 'temp' file you've created. If the file contains more than one object, remove the extra objects so that only the object or objects you want to delete remain.
3. Do not change any lines in the object(s) you want to delete (not even the 'changed:' line). Simply append a line such as the following to the objects:

   delete: user@your.net <reason for the deletion>

   To delete more than one object, append a 'delete:' line to each object and separate each object by a blank line. Your 'temp' file might then look like this:

```
route:       10.1.2/24
descr:       Example-NET
origin:      AS0
mnt-by:      AS0-MNT
changed:     pern@Example-NET 950525
source:      RADB
delete:      user@your.net prefix no longer used

route:       10.1.3/24
descr:       Example-NET
origin:      AS0
mnt-by:      AS0-MNT
changed:     pern@Example-NET 950525
source:      RADB
delete:      user@your.net prefix no longer used
```

4. Submit the object to the irr_rpsl_submit email address (e.g. auto-dbm@<domain name>). For example:

   mail auto-dbm@radb.net <temp

Overrides

The IRRd software supports a "back-door" mechanism for database administrators to update the database. It is also commonly used to enter Maintainer objects for new database users.

To use this mechanism, you must configure the override_cryptpw option in the IRRd configuration file. Adding a "override: <person> <password>" to the end of an object email submission will override the normal security checks. For example:

```
mntner:   MAINT-AS229
descr:    Maintainer for AS 229
admin-c:  Joe Smith
upd-to:   admin@blee.edu
mnt-nfy:  admin@blee.com
mnt-nfy:  noc@blee.com
auth:     MAIL-FROM user@blee.com
source:   RADB
override: administrator secretpassword!
```

Chapter 7. IRRd FAQ

Q: Can I use IRRd for both RIPE-181 and RPSL queries at the same time?

A: No. IRRd accepts both types of queries, but you can't mix the two databases; i.e., you can't combine different databases with the same daemon.

Q: Presently under RIPE-181, we send database for objects that are password-protected by making the first line of the message body say'password: <password>' followed by a blank line, followed by the updated objects. For example, a message might look like:

```
To: db-update@radb.net
Subject: ...
From: ops@bbnplanet.com

password: CENSORED

<updated records here>
```

Does this continue to work now that we've transitioned to RPSL?

A: Yes.

Q: The latest version of the RtConfig does not work with IRRD-any suggestions?

A: The latest version of the RAToolSet tools uses a different default query protocol. You will need to explicitly specify that RtConfig and other RAToolSet programs use the IRRd protocol from the command line. Invoke RtConfig with the arguments '-p rawhoisd'. Please check the RtConfig man page distributed with the RAToolSet software for more information on the flags.

Q: Why doesn't IRRd accept this policy line?

irr_database canet authoritative access 10 write-access 10
Which seems perfectly legal, according to the documentation.

A: Like many Cisco Systems commands, IRRd only allows one keyword entry per line. If you rewrite your syntax as three separate lines, IRRd will be much happier:

irr_database canet authoritative
irr_database canet access 10
irr_database canet write-access 10

Q: I just set up irrd, and I'm attempting to mirror. I'm getting this response from 98.108.0.11:43:

% ERROR: serials (1 - 70421) don't exist!
Any ideas? Thanks!

A: Your RADB.CURRENTSERIAL value has the default value of 0, which means your DB cache was not seeded properly. The mirroring request from your irrd to the server is '1 - LAST' (I know this from the information you provided above) and the reply is telling you that serials 1-70421 don't exist (since the server does not keep all serial updates forever and flushes them periodically).

The solution is to go to our anonymous ftp site (i.e., ftp.merit.edu), cd to 'radb/dbase' and get radb.db.gz and RADB.CURRENTSERIAL (i.e., reseed your DB cache). Unzip radb.db.gz and then send irrd a cache refresh command. You can do this in two ways.

Method A: from the UII (user interactive interface):
1. telnet to the uii as configured in your irrd.conf file (default is 3674)
2. type command 'reload radb'

Method B: from the query/command interface:
1. telnet to the irrd host and port (default is port 43)
2. type command '!Bradb'

The key point here is that when you are intially seeding your DB cache you must get the *.CURRENTSERIAL file along with your *.db for your mirrored DB's. You tell irrd which DB's are mirrored by specifying the information in the irrd.conf configuration file.

This should take care of your problem.

Q: Can you tell me where I can find the wget sources?

A: The wget sources are available at:

ftp://prep.ai.mit.edu/pub/gnu/wget/

Q: I'm trying to get the 'override_cryptpw' feature working. I've added a line in the config file for the override, but keep getting the error "#ERROR: Incorrect override password.". On page 26 of the 'User/Configuration Guide' (Version 2.0 Beta, 12/12/99), it mentions that an override_cryptpw must be added to the config file (which I've done), and to use this in email submissions to override normal security checks (e.g., when you want to add mntnr objects via email).

Here's the version we're using.

> IRRd> sh ver
> 1.6.1 Beta [11/12/1999 snapshot]
> Compiled on Dec 9 1999
> (SunOS 5.6 Generic_105181-16 sun4u)
> UP for 2.74 hours

What have I missed?

A: Let me run through a quick example that should answer your question. First, you'll need to use a program in ~src/programs/irr_util called crypt_gen.c. You should be able to compile it from the command line with gcc to generate an encrypted password, e.g.:

```
% gcc crypt_gen.c -o crypt_gen
% crypt_gen foo
encrypted passwd is "pfPPYJKvH.qso"
```

The above gives the encrypted password "pfPPYJKvH.qso" for the cleartext password "foo". Next, update your irrd.conf file. e.g:,

override_cryptpw pfPPYJKvH.qso

After adding this to your irrd.conf you should restart irrd. Now you should be able to use the DB admin password. Here is the syntax for this example.

```
override: gerald foo
i.e.,
override: <text string, one token> <cleartext password>
e.g.:
mntner:      MAINT-GERALD
descr:       conv test #182
admin-c:     RDM45-ARIN
tech-c:      Gerald Winters
upd-to:      gerald@merit.edu
mnt-nfy:     gerald@merit.edu
auth:        NONE
auth:        CRYPT-PW pfrutahVELjzI
auth:        PGP-FROM gerald@merit.edu
mnt-by:      MAINT-GERALD
changed:     gerald@merit.edu 19991215
source:      RADB
override:    gerald foo
```

This would cause irrd to omit normal auth checking and allow changes to "MAINT-GERALD". The syntax for the "override" is historic so I agree the syntax is strange!

Chapter 8. Related Documents

RFC-1786: RIPE-181

RIPE-181 (RIPE-81++) started it all. This document describes the original database formats used by the RIPE NCC for the storage of routing policy in its database.

RFC-2622: Routing Policy Specification Language

The current routing language used by IRRd.

RFC-2650: Using RPSL in Practice

A tutorial that gives many examples of common policies in RPSL.

RFC-2726: PGP Authentication for RIPE Database Updates

How to store PGP public keys within the RIPE database format, and by extension, the RPSL database

RFC-2725: Routing Policy System Security

The RPSL-Security specification provides a mechanism for delegating objects and providing a rooted (top-down) delegation and authentication model for objects such as AS numbers, address space and routes. Status: IRRd does not yet support this RFC.

RFC-2769: Routing Policy System Replication

This mechanism provides for a more robust and authenticated mechanism of distributing data from registry to registry. Status: IRRd does not yet support this RFC.

Karrenberg, D., and M. Terpstra. Authorisation and Notification of Changes in the RIPE Database (RIPE-120)

http://www.ripe.net/docs/ripe-120.html

Appendix A. IRRdCacher

IRRdCacher can be used to fetch and maintain copies of the IRR databases. The tool is available as part of the IRRd distribution.

Synopsis

```
irrdcacher [-h irrd host] [-p irrd port] [-s remote ftp server and path] [-
w search path component] [-c irrd cache path] [-S ] [-C ] files ...
```

Options

`-h <host>`	`Specify IRRd host name (default localhost)`
`-p <port>`	`Set IRRd port (default 43)`
`-s <ftp server>`	`Set ftp server and remote directory (default ftp://ftp.radb.net/routing.arbiter/radb)`
`-w <path>`	`Set additional search PATH component (user process PATH is the default)`
`-c <directory>`	`Specify cache directory path (default./)`
`-S`	`Supress the cache refresh signal to IRRd`
`-C`	`Perform RPSL conversion`

Description

IRRdCacher is an add-on utility that can be used to retrieve IRR databases that do not support mirroring, such as the Cable & Wireless (formerly operated by internetMCI) and Bell Canada (formerly CA*net) registries. IRRdCacher is not needed to obtain copies of the ANS, RIPE, and RADB registries, which support mirroring and are all mirrored in near-real time by Merit's production database server.

IRRdCacher differs from FTP in that it can:

Convert RIPE-181 databases to RSPL databases
Recognize the databases that make up the IRR and give them special treatment
Automatically unzip the IRR databases
Send a cache refresh signal to IRRd

If you specify an IRR database, IRRdCacher will look for the remote file as *.db.gz, unzip the file for you in the IRRd cache area, and send a cache refresh signal to your IRRd Server.

Specify the "-S" command flag to supress the cache refresh signal. If you specify a non-database file, IRRdCacher will retrieve the file with no special treatment (e.g., the *.CURRENTSERIAL files).

IRRdCacher requires two external applications to function:

wget - a program that performs the file transfer operation
ripe2rpsl - a perl program that converts RIPE-181 databases to RPSL.

The -w flag can be used to specify an additional search component to find the wget and ripe2rpsl utilities, should IRRdCacher not find them in your default path.

The distribution also comes with a sample cron job entry.

Examples

Two sample IRRdCacher sessions are shown below:

```
> irrdcacher -C -c /users/my_home_dir/irrd_cache
   -w /users/my_home_dir/bin -h irrd host@my_domain.net
   -p 5678 cw radb RADB.CURRENTSERIAL
```

This example illustrates an IRRdCacher invocation that retrieves the Cable & Wireless and RADB databases, performs a RIPE-181-to-RPSL conversion, stores the databases in the /users/my_home_dir/irrd_cache cache directory, looks for "IRRdCacher", "wget", and "ripe2rpsl" in /users/my_home_dir/bin if they are not found in the default search path, and sends a cache refresh signal to IRRd listening on port 5678 at host "irrd_host@my_domain.net".

IRRdCacher also retrieves the RADB.CURRENTSERIAL with no special treatment.

The next example shows how to use IRRdCacher without command line parameters by utilizing the defaults:

```
>  irrdcacher radb ripe ans canet cw RADB.CURRENTSERIAL
       ANS.CURRENTSERIAL RIPE.CURRENTSERIAL
```

In this example, IRRdCacher retrieves and unzips the enire IRR along with the current serial files for mirroring from the default FTP site, ftp://ftp.radb.net/routing.arbiter/radb, places the files in the current working directory, and sends a cache refresh signal to the local host at port 43.

IRRdCacher is provided as a convenience. It is hoped that some day all registries will support mirroring or some other mechanism for sharing Internet routing registry databases.

Acknowledgement

Thanks to David Kessens of ISI for writing the ripe2rpsl conversion utility.

Appendix B. RIPE/RPSL Tool Query Language

IRRd supports two query modes: single command mode and multiple command mode, which is useful for compute-intensive queries. In multiple-command mode, the query server continues to accept and service query requests on the connection until the remote user issues a quit command.

Multiple command mode is initiated with the !! command. Note that this must be the first command of the session (since, otherwise, the server would process the first command and close the connection).

The extended queries will return the following:

For successful queries returning data:

```
A<data length>\n
<<data length> bytes (including newlines) of data>
C<optional messages>\n
The "\n" denotes a newline (ASCII value 10.
```

For multi-source non-indexed queries, <optional messages> is of the form <difference count>,<difference sources>, where <difference count> is the number of databases (sources) with values different from that returned and <difference sources> is a list of space separated source names that have the different values. (The value returned was from the first source with a non-null value.)

For successful queries returning no data:

```
C\n
```

For unsuccessful queries:

```
D\n
   Key not found.

E\n
   There are multiple copies of the key in one database.

F<optional message>\n
   Some other error, see the <optional message> for details.
```

Summary of Commands

!g Get routes with specified origin.
e.g., !gas1234

!h Get routes with specified community. (RIPE-181 only).
e.g., !hCOMM_NSFNET

!j performs distributed checks on database synchronization.
This command makes it possible to view the mirror status
(oldest journal number, CURRENTSERIAL) for a database.
If a : is present after the range, the database was last
exported at that serial number. For example:

```
!jRADB,RIPE,<foo>,<bar>
!j-*                # Show all databases
Output:
A<n>
RADB:Y:1000-2000
VERIO:Y:3500-4500:4000
RIPE:N:0-666
FOO:X:<explanatory text - optional>
BAR:X:<explanatory text - optional>
C
```

Y means that the database is mirrorable.
N means that the database is not mirrorable, but the
local IRRd server is reporting the current serial number.
You can use this option to check for updates. The first
number will _always_ be zero. The second number may be
zero if the CURRENTSERIAL file doesn't exist.
X means that the database doesn't exist, or the local
server is denying information about an existing database
for administrative reasons.
Returned databases are canonicalized to upper case.

!i Return all lines of as-set. Recursive lookup available.
e.g., !iAS-ESNETEU # non-recursive, don't expand
any embedded as-set's
e.g., !iAS-ESNETEU,1 # expand any embedded as-set's

!m Match an object of the specified type with the specified key. Return
immediately after first match.
e.g., !maut-num,as701 #lookup aut-num object with key as701
e.g., !mmntner,maint-as237 #lookup mntner object with key maint-as237

!q Quit the IRRd session.
e.g., !q

!n Identify the tool for statistics/logging purposes.
e.g., !nRoe

!v Provide the IRRd version number.
e.g., !v

!r Perform route searches.
Default finds exact prefix/len match.
o - return origin of exact match(es)
l - one-level less specific
L - all less specific
M - all more specific
e.g., !r141.211.128/24,l

!s Set the sources to the specified list.
Default is all sources.
Default search order is the order in which sources are
configured in the irrd-conf file.
e.g., !sradb,ans
lc - show the currently selected sources
e.g., !s-lc

```
!u      Update the database
        !us <database>-start update
         <ADD | DEL> <OBJECT>
        !ue-end update
```

IRRd treats each database (i.e., the RADB, Cable & Wireless, and RIPE databases) as a separate object. IRRd will scan each database and return an answer from each in search order. However, the !m ... commands (i.e., 'match objects commands') and the !i... command use a slightly different algorithm.

The !m... commands return immediately after finding an object, even if an identical object exists in another registry. The !m command is used to find a match for an object and will return at most one object. For example, !maut-num,AS1234, a search for aut-num object AS1234, will return the first object it finds:

```
!maut-num,as7456
A412
aut-num:        AS7456
as-name:        UNSPECIFIED
descr:          Interhop
import:         from AS2493
                action pref=1;
                accept ANY
export:         to AS2493
                announce AS7456
admin-c:        Jordan Baker
tech-c:         Jordan Baker
remarks:        Interhop
                jbb@interhop.net
mnt-by:         MAINT-AS7456
changed:        jbb@interhop.net 19970109
source:         ANS
C
```

The !i command finds AS set objects and will recursively expand embedded AS sets when the proper option is specified. For example,

```
!iAS-ICINET
A24
AS6561 AS7252 AS-LTINET
C
!iAS-ICINET,1
A28
AS6561 AS7252 AS7790 AS7346
C
```

The !i command searches the databases in the order specified by the user and returns when it finds an object. When the ,1 option is specified to indicate embedded set expansion, the command will expand embedded sets using the database where the set was found. The other database sources, as specified by the user, will only be used when a match is not found in the source in which the set was found.

Therefore in the above sample command !iAS-ICINET,1, assuming the specified search order is RADB, RIPE, Cable & Wireless, ANS, CANET (Bell Canada), and the embedded set AS-LTINET is found in the ANS registry, AS-LTINET will be expanded first in the ANS registry, rather than in the RADB.

Below is an example of telneting to the IRRd command port and issuing a command to see all less specific routes:

```
radb3> telnet radb3 whois
Trying 198.108.0.8...
Connected to radb3.merit.edu.
Escape character is '^]'.
!r198.108.60.88/32,L
A519
route:          198.108.0.0/16
descr:          MERIT Network Inc.
                4251 Plymouth Rd
                Ann Arbor
                MI 48105-2785, USA
origin:         AS237
mnt-by:         MAINT-AS237
changed:        har@merit.edu 20001115
source:         CW

route:          198.108.0.0/14
descr:          MERIT Network Inc.
                4251 Plymouth Rd
                Ann Arbor
                MI 48105-2785, USA
origin:         AS237
mnt-by:         MAINT-AS237
changed:        jmd@merit.edu 20010313
source:         CW
C
```