

Slipper Administrator's Guide

Iagu Networks
244 Pirie St
Adelaide SA 5000
Australia

Version 3.3 R 5
August 2008
© 2001-2008 Iagu Pty Ltd

Contents

1	Introduction	4
2	Installation	7
2.1	License	7
2.2	Requirements	7
2.3	Unpacking	8
2.4	slipper.cfg	8
2.5	slipper.rules	8
2.6	Databases	9
2.7	Automatic Start	10
2.8	DNS	11
3	Rulesets	12
3.1	Variables in Rulesets	17
3.2	Plug-in Modules	26
3.3	Ruleset Declarations	27
3.4	String Replacement	28
3.5	Ruleset lookup	29
3.6	Berkeley DB lookup	29
3.7	Plug-in Modules doing Replacement	29
3.8	File Inclusion	30
3.9	Embedding	30
3.10	Error Handling	30
3.11	Expansions Format and REGISTRATION	31
3.11.1	Redirects	33
3.12	Comments	33
3.13	Line Continuation	34
4	Security	35
5	Plug-in Modules	37
5.1	Pickup	37
5.2	MusicOnHold	38
5.3	OutboundRegister	39
5.4	SimpleVM	40
5.5	Sticky	40
5.6	RingToneSelection	41
5.7	MWlcache	42

5.8	ReferToFixup	43
5.9	HideAddress	44
5.10	Programming	45
	5.10.1 Packet Format	46
	5.10.2 Call Termination Modules	47
	5.10.3 Plugin Tips	48
	5.10.4 Example Uses	49
5.11	Programming Helper Applications	50
5.12	State-IVR	55
	5.12.1 Introduction	55
	5.12.2 Operation	55
	5.12.3 State File Format	56
6	Operational Information	61
6.1	Reconfiguration	61
6.2	Internal State Dump	61
6.3	Clustering and Replication	62
6.4	Single-server Clustering	63
6.5	Ping with SIP	64
6.6	Web Interface	64
6.7	call-accounting	65
6.8	db	67
6.9	check_ruleset	68
6.10	Network Address Translation	69
	6.10.1 Holding NAT entries open	69
	6.10.2 Media Proxy	69
7	Example Config	71
8	Third-Party Integration	86
8.1	Cisco IOS Routers	86
	8.1.1 Known Problems	92
8.2	Cisco SIP 79x0 IP Phones	94
	8.2.1 Known Problems	96
8.3	Cisco Analogue Telephone Adapters	97
8.4	Cisco 7905 and 7912 Phones	98
8.5	Ahead Software's SIPPS	98
8.6	Microsoft Windows XP Messenger	99
	8.6.1 Known Problems	99
8.7	X-Ten range of Soft Phones	100
8.8	Mitel 5055 SIP Phones	100
8.9	SIPphone	101
A	Release Notes	102
B	License	119

Chapter 1

Introduction

SIP is the Session Initiation Protocol, first defined by the IETF in RFC 2543 in March of 1999, and more recently updated in RFC 3261 in June of 2002. SIP is a signalling protocol designed on Internet Standards and to interoperate with other Internet services. It performs a comparable function to the International Telecommunication Union's H.323 family of protocols.

Slipper is a full-featured SIP proxy server. SIP proxy servers are comparable in function to H.323's gatekeepers. Its primary mode of operation is stateful, supporting a wide range of special features, and it also has a stateless mode of operation used in high-reliability clustering environments for no-interaction failover support. Features include:

- Rulesets.

Slipper has a generic ruleset mechanism that controls most of its behaviour. Slipper makes most decisions only after calling one of a user-definable set of rulesets, allowing easy configuration of its behaviour.

- Plug in Perl Modules.

Functionality can be extended by a variety of plug-in modules. The standard Call Pickup, Transfer Return, and Music on Hold features are implemented as modules, and this can be used as a template for extended features you may wish to implement. Plug in modules are called from the ruleset mechanism.

- Call Expansion.

Slipper has a generic methodology named "Expansion" for multiple potential end-points to a call. This allows multiple endpoints for a call to be specified, with various criteria as to when each endpoint starts ringing — time, failure of previous entries, etc. The first end-point to pick up gets the call. The current Expansion can be modified while the call is in progress using plug in modules — this is how Call Pickup is implemented, for example. Note that unlike many PABX systems, as many of these simultaneous-ring endpoints as desired can be external.

- Authentication.

Slipper has a flexible security model defined on its rulesets mechanism, to allow differing types of calls from differing locations. This is especially needed given SIP's ability to place calls over the Internet to email-style addresses, which does not require a previous relationship set up on either endpoint. Authentication by the end client can be requested based on target to be contacted, allowing random users on the Internet to directly dial certain targets, but require authentication for others, to allow external access to published numbers, but only authenticated user to place outgoing calls through gateways, for example.

- Codec reorganisation.

Based on the source and destination pairing, Slipper can vary the preference order of codecs, or remove codecs from the preference list altogether. This is useful both in controlling the codec used, maybe G.711 over a LAN and G.729 over a WAN, or removing codecs that do not meet the quality requirements of that call.

- Dial Plans.

Slipper has a canonification process, one step of which is to translate a presented telephone number into fully qualified (international) form. This step can vary its behaviour based on the source of the call, so it is possible to have a single Slipper server handling calls from a variety of locations with differing dial plans — be that different sections of a company with different lengths of extension number or overlapping extension numbers, or endpoints in differing local call zones who wish numbers to default to their local call zone.

- Instant Messaging.

Slipper will act as a rendezvous point for SIP based instant messaging, such as that used by Windows XP Messenger.

What Slipper does through external but tightly coupled helper applications:

- Music on Hold.

This is not a signalling function and should be provided by your voice gateway. However, as some gateways do not natively support Music on Hold, Iagu provides a plug-in module that interfaces with an external process (also provided) that streams music to a device when required.

- Voice Mail.

Slipper is designed to provide a signalling service and does not directly provide end-user services. Plug in modules, such as the “SimpleVM” that ships with Slipper, can provide these services, or Slipper will interact with a wide variety of external SIP-based voice mail services, particularly using its call expansion feature. There is an example in the Third-Party chapter using the VXML feature of Cisco Routers to run a voice mail service. Slipper implements such features as the “Diversion” header to provide hints to a voice mail system as to why it is receiving the call, such as “Alice didn't answer the phone” or “Bob is busy” or “Charlie has unconditional diversion turned on”.

The “SimpleVM” that ships with Slipper has some limitations - it does not support compression.

Chapter 2

Installation

Following is the quick startup guide. For more complete information, read the rest of the documentation. Slipper has a lot of flexibility that requires configuration of its rules files beyond the simple tweaks to the sample configuration listed below.

2.1 License

This copyright message is included at the beginning of the Slipper source code, and is replicated here for reference purposes. If you do not have permission, contact your local Slipper reseller or Iagu at `sip:reception@iagu.net` or `mailto:sales@iagu.net`.

Copyright © 2001-2005 Iagu Pty Ltd.
Use and / or duplication of software requires the express written permission of the copyright holder or a licensed partner.

The complete license can be seen in appendix B on page 119.

2.2 Requirements

The system Slipper is running on should be running perl 5.6 or later, as there are some serious inefficiencies in the IO::Select module in 5.5. The Digest::MD5 module must be installed, and the Net::DNS perl modules are optional but required for DNS SRV support and non-blocking DNS lookups. Time::HiRes gives the log files greater granularity, and some helper applications (Music on Hold, Voicemail) have other requirements. Perl 5.8 is required for the HelperApp framework and dependant features such as voicemail.

2.3 Unpacking

Slipper is usually delivered as a GNU compressed tape archive named `slipper-3.2z23.tgz`. The “3.1” indicates the primary version number. The letter indicates the type of release — “a” for alpha (new features being integrated), “b” for beta (no new major features, bug fixes and workarounds only), and “r” for public release. The last number is the patch level.

Unpack this to your preferred directory — some of this documentation and support files (notably the web interface) assumes `/usr/local/slipper`, but it can go wherever you want it to go. Slipper assumes all modules, config files, etc., are rooted in the same directory as itself.

If you want to make sure your installation has all modules required by the helper applications, get the file `slipper-assist.tgz`, unpack it, and run the command `slipper-assist install` from that directory.

If the installation will use Cisco phones, either download the required files from Cisco or provide Iagu Networks or your distributor with your contract number on the phones and you will get a file `cisco-tftpboot.tgz` with the current Cisco images in the right places, with config files changed as appropriate for Slipper.

2.4 `slipper.cfg`

Probably doesn’t need changing, but you may wish to change the PID declaration to `/var/run/slipper.pid`, but not all systems have a `/var/run` so it doesn’t default to that.

2.5 `slipper.rules`

Two paths — either install the `slipper-admin.cgi` script somewhere appropriate, go to it with a web browser, and click on “System Config” and fill in the blanks, or keep reading...

Edit the variable settings at the top. `Server_Name` is the most important, it can be an IP address, but if it’s a name one should have DNS SRV records as well as A records, since some SIP devices insist on SRV records. Change or add `Bind` lines so that your system’s local IP addresses are mentioned. If you need a wildcard, bind to `*`.

Aliases should also be set for any IP address or name that may be used to refer to the server running Slipper, but those IP’s mentioned in `Bind` commands are automatically added. `Default_Gateway` is set to the name or IP address of the router that provides PSTN access. `Local_Prefix` is set to the PSTN prefix this server is responsible for, so it knows to return a “604 number does not exist” message if a number is called that it can’t map to a name.

The example configuration assumes that local extensions start with a 2 and are three digits long — the 2 gets stripped off and is replaced with the `Local_Prefix` to get a canonical number. Note that in the example configuration shipped, it assumes that numbers should be treated as if dialled on the PSTN network in the 08 area code of Australia — this is reasonably easy to change. You’ll almost certainly want to change the Group ruleset to reflect your internal IP addresses.

2.6 Databases

Slipper's rulesets also refer to a number of databases that should be populated. Note that these can all be managed through the web interface if desired.

- **number-to-user.db**

Maps canonical addresses (usually numbers) to alphanumeric form. Should map all externally-visible numbers to something without a leading +, otherwise Slipper will believe that number is not set up for access and will be rejected. This behaviour can be modified.

```
Example: db number-to-user.db add +61884252255 reception
```

- **expansions.db**

Can be blank (just touch a zero length file), this controls targets that should ring in multiple places.

- **registrations.db**

Although you can manually modify this, usually it is left alone and is populated automatically by phones registering their presence at their current IP address. To match the above example, a phone might register as the user `reception`, and all calls to +61884252255 (or the equivalent) will be presented to that device.

- **user-to-number.db**

Any target numbers that divert off somewhere else should be here. For example, you might have:

```
db number-to-user.db add +61884252200 head-office-diversion
db user-to-number.db add head-office-diversion +61312345678
db expansion.db add reception reception,head-office-diversion=15
```

To allow head office to have a local number, and if reception doesn't answer their phone in 15 seconds (or immediately if the phone is busied out, eg, has Do-Not-Disturb or has too many existing calls for the device to handle), the call starts ringing in head office as well.

- **display-names.db**

Maps a canonical address to a text string to be presented in a calling line ID (CLID) on a SIP device. Usually used to associate a person's name with a number or address.

```
db display-names.db add +61884252255 "Iagu: Main Number"
```

- **gateway-display.db**

Modifies CLID information presented on calls going out the default gateway. Takes a Canonical name and returns what should be presented to the gateway. If the string `;privacy=full` is appended, the outbound call is marked with CLID suppressed, but is still set for billing or emergency services purposes. If the strings `;exposed=test` or `;alternate=` are appended — well, see the next database.

```
db gateway-display.db add reception +61884252255
```

```
db gateway-display.db add johndoe +61884252287;privacy=full
db gateway-display.db add janedoe +61884252288;exposed=test
db gateway-display.db add alice +61884252289;exposed=test;alternate=+61884252280
```

- **expose-clid-to-target.db**

If the `;exposed=test` tag is set as part of the above process, that CLID is only exposed if the call is going to a target listed in this database, otherwise the defaults are used. This allows one to set things up so that outbound calls from a desk phone give a generic number unless they are calling a company mobile or some trusted third party, in which case the actual internal extension is exposed. Only the presence of the entry counts, not the value it contains.

If you don't want it to use the defaults for outbound CLID, then follow up the `;exposed=test` entry in `gateway-display.db` with `;alternate=+61...`, then if there is no entry for the callee in `expose-clid-to-target`, the alternate number is used as the CLID.

```
db expose-clid-to-target.db add janedoe-mobile OK
db expose-clid-to-target.db add +61881242200 OK
```

- **transfer-rewrite.db**

This only exists for compatibility with gateways that only support digits in their dialling plans, to indicate to them where a call transfer is supposed to go to in terms of digits it can cope with. This is usually a mirror of the `number-to-user.db` database, and the web interface has a single click allowing you to sync this from the latter.

-

Note that all of the features described above only exist due to default rule-sets, in a carrier or other non-PABX environment all the above features can be disabled or completely changed to meet your requirements.

2.7 Automatic Start

The provided script `slipper.sh` can be called with "start" to start Slipper operating, and can be placed in the appropriate `rc.d` directory of most operating systems. Note you may want to add `su -c` commands to make it run as some other username.

2.8 DNS

An example minimal DNS configuration for SRV records might be:

```
$ORIGIN YOUR.DOMAIN.NAME.  
_smtp._tcp      IN  SRV  0  1   25  mail  
_www._tcp       IN  SRV  0  1   80  www  
_pop._tcp       IN  SRV  0  1  110  mail  
_pop3._tcp      IN  SRV  0  1  110  mail  
_sip._udp       IN  SRV  0  1 5060  sip  
*._tcp          IN  SRV  0  0    0  .  
*._udp          IN  SRV  0  0    0  .  
sip             IN  A                203.32.153.129
```

Chapter 3

Rulesets

The rulesets file (traditionally `slipper.rules`) defines rulesets and can also set variables, used either in the rulesets themselves or to modify Slipper's behaviour.

The three main rulesets are Canonify, to change a network-format SIP URI into internal normal form, Expand to provide user-level control of call features and presentation, and Route to convert internal normal form to external form and determine the final target.

The full list of rulesets Slipper itself calls are as follows:

- Authenticate

If the requester presents authentication credentials, this ruleset is passed the `presented_username` as input, and is expected to return either `password` or `password '!' group_modifier`. If the “password” returned is the same as the presented username, Slipper assumes this ruleset has not been set up and fails the authentication. Likewise, if the “password” returned is the null string, Slipper assumes the specified user does not exist and also fails the authentication.

If the password returned starts with `MD5:`, that prefix is stripped and is assumed to be an MD5 hash of `username ':' Realm_Name ':' password`. Note that although the password cannot be determined from this hash other than by brute force, this hash should still be protected as a clear-text password as it can be used to construct the correct authentication digest.

If a `group_modifier` is returned, it is prepended to the Group if it has a trailing hyphen, appended to the Group if it has a leading hyphen, ignored if it's the null string, or otherwise it replaces the Group name. This is traditionally used by the Authorise ruleset to determine if a roaming (but authenticated) user is allowed different levels of access to an unauthenticated external user by behaving differently for the group “External” that for the group “External-Auth” or “External-username”.

- Authorise

For any new call, the Authorise ruleset is passed `group_name '!' URI`, where the URI is what is returned from the Canonify ruleset. An error response is returned to the requester, anything else is ignored.

- Auth Digest

It is used if you wish to send a call to an external gateway that requires the provision of authentication details. If a request is sent and a response of 401 or 407 is returned with `WWW-Authenticate` or `Proxy-Authenticate` headers requesting Digest authentication, data is sent to this ruleset to request login details. If it returns data, it will resend the request with the authentication details.

This ruleset is called and passed `realm '!' Remote_Group '!' remote_ip ':' remote_port`. If the ruleset can find a valid username and password for authentication, it should return `remote '!' username '!' password`.

`remote` is either just an IP address or an IP address, colon, and then a port - the former affects all traffic to all ports on that IP address. The password may contain exclamation marks. Lack of at least two exclamation marks, or the same data returned from the ruleset as passed to it, is perceived to be a failure.

The password may be the special string `MD5:` followed by an MD5 hash of `username ':' realm ':' password` if you do not wish to store plaintext passwords.

Note that use of this ruleset technically violates a number of RFC's, but many consider it an absolute requirement. From a local perspective, Iagu believes the requirement that proxy servers not add authentication information to be burdensome given one then has to somehow distribute all authentication information to all SIP devices you may proxy a call from. Clearly, if someone random out on the internet is calling your SIP proxy and you wish to divert the call to a third party that requires authentication (for example, a SIP to mobile phone gateway), you do not wish to provide that unknown end user your authentication credentials for a service you may be charged for!

This is usually used with the `OutboundRegister` optional plug-in module - see section 5.3 on page 39 for more details.

- Canonify

Take an externally provided address and put it into normal form, whatever you consider normal form to be. In most cases this is to strip off local domains, fix up phone numbers according to a dial plan into "normal" form (normally international format) and map any aliases to their names, such as mapping phone numbers to alphanumeric SIP addresses.

For example, it might take an inbound call to `sip:0884252255@iagu.net:5060` and map it to `reception`, or maybe an outbound call to `sip:82335833@iagu.net` and map it to `+61882335833`.

- Codec

Re-prioritise the codecs in an application/sdp body — the standard Session Description Protocol message used to propose a set of media streams, specifying IP address and port of the endpoint as well as the offered codecs. The ruleset is passed `source_group '' destination_group`. If it returns `codec` or `codec[!],codec[!],...`, it will attempt to re-order the

codecs, moving codecs in the SDP description up in the order they are specified in the ruleset response, with any unlisted codecs coming last (in the order they were originally proposed), and any codecs in the response with a “!” prepended are removed.

- Expand

Expand a canonical address into a list of possible targets. See 3.11 for the format of the expected result.

- Force Socket

This ruleset is passed `source_group` “!” `destination_group`. If it returns a string starting with `SOCKET:` it then takes special action. If the remainder of the string is a valid socket created from the “Bind” list, named in the form `udp:ip_address:port_number`, then it forces the packet to be sent from that socket. The string after the `SOCKET:` can also be any of the following:

- Public

- Pick a socket from those bound with a public IP address.

- Private

- Pick a socket from those bound with a private IP address.

- NonLocal

- Pick any socket except one bound to a loopback IP.

- Group

The Group ruleset is passed an IP address, and it should return a group name. This is not used internally by Slipper, but is provided as a variable to other rulesets, and the source and destination groups are passed to the Codec ruleset. If the Group returned has the prefix “AUTH:”, it is stripped and authentication is forced. If the Group returned is the special string “JUNK”, the packet is discarded without further processing.

- Input Channel

For a received packet, after decoding has been performed, the Input Channel ruleset is called with the name of the Group. The packet is available through the normal ruleset mechanisms for modification.

This bears a lot in common with Input Filter. Input Filter is performed on the raw text prior to decoding, while Input Channel is allows operations or inspection to be performed on the packet after decoding. Except where required to fix up protocol violations that would prevent decoding from being successful, Input Channel is the preferred method.

- Input Filter

For a received packet, after the Group of the source has been determined but before actual decoding of the packet is performed, the Input Filter ruleset is called with the name of the Group. In the packet meta-data is a field called “Buffer”, which is a text string of the undecoded packet. The ruleset can then call modules that modify this particular string in

the packet meta-data. On the return of the ruleset, the packet is then processed as though the current contents of the Buffer string were the actual string received from the remote party.

This feature is intended for use to work around problems with certain devices that break the standard in some way and require fixup prior to becoming visible by the system at large. It can also be used for certain security requirements.

- On Register

After Registration has succeeded, this ruleset is passed `registration_target` ‘!’’ `registration_contact` ‘!’’ `source_group`, subject to any modifications the “Register” ruleset made.

This ruleset is intended to be used as a hook for plug-in modules interested in the presence of SIP devices, such as MWIcache.

- Output Channel

The reciprocal of the Input Channel, used to view or modify the internal representation of the output packet.

- Output Filter

The reciprocal of the Input Filter, used to modify the string representation of the output packet.

- Register

When a local registration request is received, this ruleset is passed `registration_target` ‘!’’ `registration_contact` ‘!’’ `source_group`. If the ruleset returns exactly what it was passed, or the string `REGISTER`, it does a normal internal registration. If it returns an `ERROR:`, that is returned. If the return is `REGISTER:new_target`, then `new_target` is registered with that packet’s contact details instead of the originally specified target. Any other return value is assumed to indicate a plug-in module performed some sort of registration, and a 200 OK is returned to the requester.

It is intended that this ruleset provide two features:

- Authentication support: It can return `ERROR:403:Forbidden` or `ERROR:407:Proxy Authentication Required` to refuse registration from that source.
- Allow plug-in modules to store registration information in some other database.

- Rewrite URI

Passed the URI in a “From”, “To”, or “Remote-Party-ID” header, as defined by the `Rewrite_ ... _Header` variable, by default only the “Remote-Party-ID” header. If this ruleset returns `URI:new_uri`, then the URI on that header is changed to `new_uri`. This is primarily used for two purposes:

- Fix up a dialled number such as 201 into the correct form, eg `andrewr`.
- Change the Caller ID to something acceptable to the PSTN network.

Rewrite URI is called on an outgoing packet, with `${Remote_Group}` set to the destination group. Note that if the returned URI has tags (such as `;tag=value`) then those tags are moved to the end of the complete header, and any existing tags of the same name are removed in favour of the returned one. This is usually used to override `;privacy=off` with `;privacy=full` to suppress Caller-ID, or `;screen=yes` to confirm this Caller-ID as authenticated.

- Rewrite Display

Similar to Rewrite URI, and called immediately after it. Passed the URI in the header. If this ruleset returns `DISPLAY:new_display_name`, then the Display Name on that header is changed to `new_display_name`. Commonly used to map the numbers of incoming calls from the PSTN network to the name of the owner of that number.

Like Rewrite URI, Rewrite Display is called on an outgoing packet, with `${Remote_Group}` set to the destination group.

- Rewrite “Header”

A number of headers (currently “Contact” and “Refer-To”) are Canonified as they come in, and then prior to being passed out a ruleset named “Rewrite” followed by a space, followed by the header name is called. This is after the outbound Group is determined, and is primarily intended for rewriting URI’s into forms palatable for the targeted device, such as rewriting alphanumeric user parts of the URI into a numerical equivalent for devices that only support numerical user parts. For cached authentication to work, the result of this ruleset should be something that the Canonify ruleset will turn back into the initial input to this ruleset.

If the ruleset returns the string it was passed, Slipper will pass it to the Route ruleset for final modification.

- Route

Take URL and determine where it should be routed. See also 3.11

Other rulesets may be defined and called by the major existing rulesets. The names of rulesets may contain spaces.

3.1 Variables in Rulesets

There are two types of variables, single variables and lists. Single variables are modified with the `Set` command, and lists have items placed in them with the `Add` command. Variable names should not have spaces. The format for `Set` and `Add` commands is:

```
[Set — Add] LWS+ Variable_Name LWS* [=>:] + LWS* Value
```

That means one or more, in any order, of “=”, “>”, or “:” must separate the variable name from the value. LWS refers to linear white-space, with “+” referring to one or more and “*” referring to zero or more. If the Value is set to “”, that is, two double quotes, it is instead treated as the empty string for “Set”, or removes all entries from the list for “Add”. Note quoting is not required.

The following variables are used internally by slipper:

- `Alias.Resolve`

If this variable is set to 1, then all names in the Alias list are resolved to IP addresses which are also added to the Alias list. This is intended to simplify configuration. If this variable is set to 0, then you must manually add IP addresses to the Alias list. This variable defaults to 1.

- `Allow_Unauthenticated_ACK`

If this variable is set to 0, Slipper behaves as you would probably expect with regards to Proxy Authentication. If it is set to a positive integer, then if an authenticated INVITE was received, the ACK following it does not need to be authenticated. Otherwise if someone could sniff the return 200 OK and then inject a fake unauthenticated ACK with different media credentials, they could hijack someone else’s outbound call. This is required for some broken implementations, and is such is currently set to 1 by default. Note that as more vendors fix their implementations we expect to change the default to 0, so if you know you require this behaviour we suggest you explicitly set it in the config file.

- `Authorise_All-Headers`

If this variable is set to 1, then all URL’s seen in Contact or Refer-To headers are automatically deemed authorised for this call, regardless of what the “Authorise” ruleset indicates. This should only be used where the network is trusted to the end point but users are not (eg, a private hard-phone network). This is a risk-management issue - do I want to spend the time getting the Authorise ruleset right in environments with complex rules as to who can call whom, or do I trust the users not to attempt to crack the SIP packets? Use at your own risk.

- `Call_Expiry`

Calls are removed from the call database after this many seconds. Defaults to 28800 seconds, or eight hours. The larger the call database, the slower packet processing, but if it removes a call that’s in progress, pre-authentication information is lost. This is an issue if A can call B, but B can’t call A, and A calls B but B wishes to send the BYE to terminate the call. This works if the call is still in the call database (as Slipper can verify that it’s part of an established transaction), but not if the entry has been retired from the database.

- **Call_Completed_Expiry**

Similar to `Call_Expiry`, but only removes calls for which all call legs are listed as either cancelled or finished (with a BYE request). Defaults to 300 seconds, or five minutes. If it is set to zero, this feature is turned off and only `Call_Expiry` is used.

- **Comma_Delay**

The number of seconds delay a comma introduces when returned by the `Expand` ruleset. Defaults to 5 seconds.

- **Debug_Level**

Specifies the level of debugging slipper reports.

- 0: No debugging.
- 1: Event summaries.
- 2: Packet dumps.
- 3: Descriptions of logic choices.
- 4: Detailed timing of IO operations.
- 5: Ruleset debugging. (This is usually set only by `check_ruleset`.)

Regardless of the level of debugging, warnings of unexpected activity will still occur.

- **Default_Register_Expires**

If a `Register` request is received without an `Expiry` time we can find (it may be embedded somewhere we don't expect), then treat the packet as if it had specified this number of seconds. Note this does not override an explicit setting of zero seconds, which is used to cancel a registration.

If you wish to cancel this behaviour, set it to 0 (zero), then any such registration will be considered a registration cancel.

- **Digest_Validity**

Specifies the maximum amount of time in seconds the same nonce can be used by the same client for MD5 Digest authentications. Defaults to four hours, and to avoid client confusion should be at least the registration interval.

- **Fixup_Refer_To**

If this is set to anything other than "0", then any `Refer-To` headers in a `REFER` packet are modified to have the URI part contain the `Contact` header from that call.

Consider the following situation: A calls B. B does an attended transfer to C, but C is not there. D does a `Call Pickup` to get the call from B. A conversation then ensues, and B commits the call transfer. If B sends C in the `Refer-To` header (rather than the contents of D's `Contact` header), the call goes nowhere. This fixup will overwrite C's info with that of D's `Contact` header, and the call should proceed as expected.

This is currently set to “0” (off) by default, as Cisco’s IOS gateways don’t handle alphanumeric URI’s according to the RFC’s and the transfer will fail. Note that one can use the `Slipper::ReferToFixup` plug in module to work around this problem, and doing so is advised for all new installations. It is intended that the default be changed to “1” at some point in the future.

- **Invite_Retransmit_Delay**
The number of seconds to wait before retransmitting an Invite Request. Defaults to 1 second.
- **HelperAppContact**
Used by the HelperApp system, a default of “helperapp” will be used if this variable is not set. It normally defines the user part of the URL used in the Contact header, but may contain a full URL.
- **Junk_For**
If the **Group** ruleset returns the special string “JUNK”, then the packet is discarded without further processing and without being put in the log. If the debug level is set to 3 or higher, a note will be placed in the log that a packet from that location was junked. Further packets up to the number of seconds specified in this variable (defaulting to 1800, or 30 minutes) will be ignored without even a junk message going into the log.
- **Keepalive_Period**
How often NAT keep-alive packets should be sent. Defaults to once every 30 seconds.
- **Log_File_Name**
Output is sent to this file instead of standard error - in fact, standard error is closed and reopened sending to this file. The filename can have %y, %m, %d, and %h present, which are replaced by the local year, month, day, and hour respectively. If you want to use UTC/GMT instead of the local time, use the letters uppercased instead.

This feature is not used if this variable is sent to the empty string, which is the default, although the example configuration file sets it. This is so notification of an error in the configuration file doesn’t fall into the trap of attempting to be written to a file that doesn’t exist.
- **MAIN_PROCESS** This variable is set only in the parent (signalling control) process to the value 1, and can be used by plug-in modules used in both signalling and media processes to modify behaviour as appropriate.
- **Max_Delay_Factor**
When Slipper retransmits packets, it does so on a random exponential basis. The first retransmit is exactly one multiple of the appropriate `*_Retransmit_Delay` later, the next is either one or two multiples, the third is either one, two, three, or four multiples later, and so on, the the maximum doubling each time. This variable defines the maximum possible multiple, and defaults to 16. As such, this will not come into effect unless `Max_Retransmits` is modified.

- **Max_Header_Length**
If any header is over this number of bytes in size, decoding of the packet is rejected. Note splitting on commas for headers such as Via is done first, so a long chain of proxies in a path will not cause hits on this test. Defaults to 1023.
- **Max_Retransmit**
The number of times a request should be retransmitted. As this is the number of times it is retransmitted, the maximum number of times the packet is sent is one more than this, counting the original transmission. If the maximum number is reached without response, then Cancels will be issued to cancel the request¹, and the system will remove the route (and fall over to backups) as if a 504 Server Timeout was received. Defaults to 3. See also Max_Delay_Factor for a discussion on the random exponential delay in retransmission.
- **NAT_Expiry_Period**
How long NAT keep-alives should be sent to a NAT device after the registration expires in seconds. Defaults to 600 (10 minutes).
- **NAT_Modify_Registration_Expiry**
If set to 1, then where non-application aware NAT is detected it will modify the expiry of any incoming registration request greater than NAT_Expiry_Period down to NAT_Expiry_Period. This should cause the requesting device to re-register more frequently, overcoming NAT timeouts without requiring keep-alives to be sent. Defaults to 1, should be set to 0 if Notify keep-alives are in use.
- **PID_File**
A file to write the process ID of this instantiation of Slipper to once the configuration has been read, checked as OK, and sockets have successfully been opened, just before entering the main loop. The PID is re-written during a reconfiguration. This may be set to "NONE" to prevent writing the PID to a file.
- **PID_File_Fail_OK**
If set to any value, do not count an inability to write to the PID file as an error that prevents the configuration from being used.
- **Random_Cluster**
Used to limit security checking in certain environments where multiple Slipper servers are clustered and for some reason messages associated with a call cannot be guaranteed to be tied to a single server for one reason or another. See 4.
- **Realm_Name**
The public domain name of the group of servers, used as the authentication realm in MD5 Digest challenges. Defaults to **Server_Name** if **Realm_Name**

¹Unless it was a Cancel that was being retransmitted, in which case the leg will just be ignored.

is not set. This is required to be set where there is a shared password database between servers, either with MD5: encrypted passwords (so they all need to use the same realm) or to limit the number of authentication requests the user gets by making sure all challenges are from the same realm.

If the `Realm_Name` is set, it is automatically added to the Aliases list.

Previously called `Auth_Realm`, it may be used in future releases for cluster identification, or other purposes related to (but not strictly limited to) authentication.

- `Registrations_File`

The file containing a Berkeley DB that registrations are to be stored in.

- `Registration_Grace_Time`

If a registration is received with greater than one second as the expiry time, a number of seconds equal to this value is added to the registration's expiry time. This is to allow for clients that don't re-register until the last possible second, and have some clock skew relative to Slipper such that there are a few seconds in which Slipper does not consider the client registered. Defaults to 5 seconds.

- `Registration_Min_Time`

Any non-zero expiry time in a registration request less than this number is increased to this value. Defaults to 30 seconds.

- `Replication_Key`

A key used for signing registration replication messages, must be the same on all systems in the cluster.

- `Replication_Tolerance` To prevent replay attacks, registration replication requests are not processed if the local clock differs by more than this many seconds from the time the request was sent. Defaults to 5 seconds.

- `Request_Retransmit_Delay`

The number of seconds to wait before retransmitting any request other than an Invite. Defaults to 3 seconds.

- `Remove_Excess_Codecs`

If set to 1, any codecs with a number below 100 beyond the first in a "200 OK" response are removed. Works around problems with some devices accepting more than one codec in the "200 OK" but then refusing to accept RTP for codecs beyond the first. Defaults to 0.

- `Rewrite_To_Header`

- `Rewrite_From_Header`

- `Rewrite_Remote_Party_ID_Header`

These three variables define if the rulesets "Rewrite URI" and/or "Rewrite Display" are called on their respective headers. If the value of the variable

contains a `U` then Rewrite URI is called, and if the value of the variable contains a `D` then Rewrite Display is called. By default, the variable for “To” is set to the empty string so no rewrites occur, the “From” is set to `D` so only the Display Name is modified, and for “Remote-Party-ID” it is set to `UD` so both are modified. Can also be set to `-` to do no rewriting.

The rationale for this choice is as follows:

Rewriting the URI in the From and To headers confuses a lot of clients that match calls based on these values instead of just the tags as the standard says, therefore this isn’t done. Although Remote-Party-ID should be used for Caller ID, many older clients don’t support it, so Display rewriting on the From field is used to duplicate it. Running rulesets unnecessarily can impact your performance, and since no clients known to us actively use the Display portion of a To header, this is not rewritten.

Modify only if you know not only what you are doing, but if you also know what most of your SIP devices also expect from their URI’s.

- `Server_Name`

The primary name the server should use to identify itself with. This is used in Via headers so must be unique to ensure the correct return path. In theory recipients may fix up the Via header with the IP address they receive the packet from, but this is not implemented widely enough to rely on it.

The `Server_Name` is automatically added to the Aliases list.

A cluster’s common name should be defined setting the `Realm_Name` to the name of the cluster.

- `Status_File`

If defined, Slipper will occasionally write to a file of this name the current state of all active calls in its database. This is then often used by web applications to give an operator a view of who is talking and who they are talking to. “Occasionally” is roughly defined as “as soon as possible after something changes, but waiting until a gap in packet processing so as not to slow things down.”

- `Status_File_Format`

This variable defines the format in which the status file is written out in. “2” is the default, “1” exists if backwards compatibility is required.

- 1:
 - * Call Source
 - * String: space “*iii*” space
 - * Call Destination
 - * String: space “(”
 - * Unix timestamp of last message in this call setup flow.
 - * String: space “/” space
 - * Code of last response, eg, “200” for “200 OK”
 - * String: “)”

– 2:

- * Call-ID
- * String: “;cseq=”
- * Last seen CSeq
- * String: space
- * Call Source
- * String: “;orig=”
- * Base64 encode of “From” header, including tag.
- * String: space “;” space
- * Call Destination
- * String: “;orig=”
- * Base64 encode of “To” header, including tag if available
- * String: space “(”
- * Unix timestamp of last message in this call setup flow.
- * String: space “/” space
- * Code of last response, eg, “200” for “200 OK”
- * String: “)”

- Stay_In_Path

If this variable is set to zero, Slipper proxies the packet but does not insert a Record-Route header, so the end devices communicate all signalling traffic after call setup directly. If it set to anything else (and also by default), then Slipper inserts a Record-Route packet and so requests to stay in the path. This does not affect Slipper’s handling of source routed packets, it handles them correctly regardless of this setting, so you can change the setting and it won’t affect any ongoing calls.

It can also be set to the special-case value 2543, in which case it inserts a Record-Route header requesting the older RFC 2543 strict routing, instead of the loose routing that replaced strict routing in RFC 3261. This is primarily intended as a backwards compatibility step for older clients.

Note that regardless of this setting, Slipper will honour requests for both loose and strict routing, and will also correctly handle mixed routes where part is strict specified and part is loose specified. It also tries to detect and fix up the case when Slipper has specified loose routing and the client chooses to implement strict routing incorrectly.

- Strip_Transport

If set, removes transport tags such as `;transport=udp` as some devices will drop packets bearing such information.

- Suppress_Empty_Packet_Reports

Some clients, such as X-Lite, send empty packets to their proxy server to hold dynamic NAT mappings only. Normally Slipper suppresses reporting null packets, however if this variable is set to anything other than “Yes” or “True” (case-insensitive) Slipper will report these events, sometimes useful for debugging. Default setting is “True”.

- `Type_of_Service_Signalling`

This is a comma separated list of strings defining the IP Type of Service set on all SIP signalling packets.

- `Digits 0-7`
This sets the IP Precedence to the indicated value.
- `Low-Delay`
- `Throughput`
- `Reliability`
- `Min-Cost`
The last four set the appropriate type of service flags.

All text strings are case-insensitive and the hyphens are optional. It defaults to “5,relia**ility**”.

- `Type_of_Service_Media`

This is used in a similar fashion to `Type_of_Service_Signalling`, but for media packets generated by the HelperApp system. The default is “5,lowde**lay**”. For music on hold, the default of “5,lowde**lay**” cannot be changed through the rulesets mechanism, but instead must be passed as an option.

And the following lists are used internally:

- `Aliases`

A list of names the server may be known as. Slipper uses this to determine if it the kernel has transparently intercepted packets and passed them to Slipper to detect. `Aliases` is usually also used in a variety of rulesets, especially `Canonify`, to determine if a SIP URI is local or not.

If a hostname appears in the `Aliases` list, at configuration time Slipper will call the system `gethostbyname()` function, and add any IP addresses found to the `Aliases` list as well.

Note that the `Server_Name` and `Realm_Name` variables are automatically added to the `Aliases` list.

- `Bind`

A list of IP addresses and/or ports to bind to and listen for packets. Valid formats include:

- `ip_address:port`
- `ip_address`
- `*:port`
- `*`

Where `ip_address` is of the form `1.2.3.4` and `port` is a number between 1 and 65535, inclusive. `*` indicates a non-IP-specific bind. If the port is not specified, it defaults to 5060.

If the first two formats are specified, they are added to the `Aliases` list. Note this means `1.2.3.4:5060` and `1.2.3.4` differ in that they bind the same, but a different string ends up in the `Aliases` database.

- Replication

A list of `host` and/or `host:port`, where “host” is either an IP address or a hostname. All registrations are replicated to this list of servers. Note `Replication_Key` must also be set. Any exact matches on `Bind` or `Server_Name` are stripped, allowing for ease of distributing identical configurations throughout the cluster. Host lookups are done through the local systems `gethostbyname()` call rather than DNS, and are done at time of configuration only, thus if the external name resolution changes a `SIGHUP` is required.

- `Replication_Aliases` Lists `host` and/or `host:port` that registration replication will be accepted from, where “host” is either an IP address or a hostname. Automatically includes the contents of the `Replication` list. If a hostname in the `Replication` list expands to multiple IP’s, only the first is used for replication, while all IP’s are used in this list. All the resolution caveats of the `Replication` variable also apply here.

- Tick

A list of functions or functions with parameters that will be called on a regular basis - per housekeeping tick, which is at most once per second, less under high load conditions. Current time and current config are passed at the beginning of the parameter list.

For example, if you had:

```
Add Tick: Slipper::MusicOnHold::tick
```

Then Slipper would arrange for the following to be called:

```
Slipper::MusicOnHold::tick ($time, $config);
```

If however you had:

```
Add Tick: Slipper::MusicOnHold::tick ("type3")
```

Then Slipper would arrange for the following to be called:

```
Slipper::MusicOnHold::tick ($time, $config, "type3");
```

3.2 Plug-in Modules

To declare a Perl package that will be used, the format is as follows:

```
Module LWS+ package_name
ModuleConfig LWS+ Perl_commands
```

A `ModuleConfig` command directive is interpreted directly as Perl code, and should return an array. The array returned consists of lines to be read into the configuration at this point, so if nothing needs to be added to the config and it is just to initialise data structures, a null list should be returned. The original intent of this directive was for data structure initialisation, but it can also be used to massage data (such as dial plan information) from some other format supplied by a third party into a format suitable for Slipper. Note that any valid config line can be returned, including other `Module` or `ModuleConfig` directives.

A `ModuleConfig` directive is re-evaluated at every reconfiguration, if live reconfigurations are done.

After a successful `Module` initialisation, Slipper attempts to run the subroutine “`slipper_autoconf`” in that module, with the same semantics as the `ModuleConfig` (ie, returning an array, inserted after the line rather than replacing the line.) Note that `ModuleConfig` may still be useful as variables can be set in between a module declaration and a `ModuleConfig` reference.

More information on modules can be found in Chapter 5 on Plug-in Modules.

3.3 Ruleset Declarations

Format of a ruleset declaration:

```
Ruleset LWS+ Name of Ruleset until end of line
```

Followed by one or more lines of the form:

```
LHS tab+ [operators] RHS
```

The left hand side and right hand side are pretty much Perl standard because it is actually implemented within Slipper as an `eval`. However, some pre-parsing is done to assist standard Slipper usage, so “+” is a literal “+” rather than meaning “one or more”, due to its prevalence in dial-plans — if you want the normal behaviour, use “{1,}”. “@” is also automatically escaped.

The following operators are defined:

- “” — no operator.

No special action

- “=” — final answer

This is the final answer for this ruleset and it will drop out and return the RHS value without processing further rules in the ruleset.

- “:=” — repetition

Keep repeating this line over and over until it fails to match. Normally each rule is only evaluated once, and regardless of success or failure on matching the LHS, it continues on to the next rule.

- “*=” — don’t strip

Normally, spaces are stripped from both sides, so the rulesets can be a little more readable. This operator inhibits white-space stripping from the RHS. It is expected this would only be used for intermediate steps inside the Rewrite Display ruleset, and any other ruleset it might call. Note that if you use this, and wish to match on the result later, you may need to wildcard it or use `\s`.

- “@=” — display final answer

Combines semantics of “=” and “*=”. Primarily for the Rewrite Display ruleset to return display names with spaces.

Note that if a ruleset is declared twice, rules from the second declaration are added to the end of the previous one. This can be important for `ModuleConfig` directives, who may wish to add rules to the end² of a variety of rulesets defined elsewhere.

“!” is traditionally used as a context separating placeholder. The only time it is used elsewhere is for negating codecs, due to the traditional interpretation of “!” for that purpose.

²Not always the end — it could be the beginning, or the middle, depending on the placement of the `ModuleConfig` directive.

3.4 String Replacement

String replacement: `${Variable_Name}` is replaced prior to the eval. In the LHS, it is replaced by a bracketed regexp portion that covers all members of that variable. This is can be important in lists, so that if `${Aliases}` matches on the LHS, it can be referred to by (say) `$2` in the RHS.

On the RHS, such an expression is replaced by the contents if it is a single variable — for lists, it could not determine which one to place, so it ignores it. As well as global variables defined in the rules file, one can also refer to variables of the packet itself. First global variables are checked for a match, then packet meta-data, then headers of the packet itself. If there is more than one matching header line, the first is returned. If there is no match, a null string is returned.

Packet meta-data includes:

- Method
The Method in a request, such as INVITE, ACK, BYE, or CANCEL.
- URI
The URI in the request, such as `reception@iagu.net`.
- Code
The code of the response, such as 100 for Trying, or 200 for OK.
- Reason
The text message associated with the Code of a response, such as “Trying”, “OK”, or “Need more digits”. Unfortunately error messages are not always presented to the user.
- Auth_User
If authentication has occurred, this is set to the username used in the successful authentication.
- Proxy_Auth_Require
The system believes that this packet requires authentication. Used internally — most rulesets won’t see this until authentication has succeeded.
- Proxy_Auth_Header
If this packet requires authentication, this is the suggested header to use, taking nonce history into account.
- Remote
Where the packet came from. This string is passed to the “Group” ruleset to determine the Remote_Group. On an incoming packet, this is set to `ip_address` “:” `port_number`.
- Remote_Group
The group the packet came from (for inbound packets) and the group it’s going to (for outbound packets). Not set in the “Group” ruleset, obviously! When Slipper calls the following rulesets they are considered incoming: Canonify, Authorise, Expand, and Route. Although Route

may be considered outgoing, the system doesn't know what the Group will be until the Route ruleset has told it where the packet is going. This also allows you to match on the incoming Group when determining where to route a call, for example when least-cost routing can't determine a destination, you may wish to send the call out as close as possible to its entry point to the network.

The Rewrite URI, Rewrite Display, and Codec rulesets are considered outgoing. This allows you to specify the way calls are presented to different target groupings, be they organisational, geographic, or otherwise.

It is important to note that there are actually two evals for each line, the first to check if a replacement is going to occur, and the second to actually do the replacement. This is primarily due to optimisation of embedded ruleset evaluations.

3.5 Ruleset lookup

Format: “%” Rule Set Name “:” input “%”

The “input” string is passed to “Rule Set Name”, and the resulting string is used to replace.

3.6 Berkeley DB lookup

Format: “%” file_name.db “:” [PREFIX “:.”] input [“ | ” alternate] “%”

Opens up file_name.db as a Berkeley DB file, and tries to find “input”, followed by a lower-cased version of “input” if it contains upper-case characters, followed by “*”.

If the DB lookup succeeds, it returns the value, with “PREFIX:”³ prepended if one was defined.

If the lookup fails, it returns the null string, or the alternate if one was defined. This means many lookups are of the following form, to preserve the original value if a match is not found.

```
%expansions.db:$1|$1%
```

3.7 Plug-in Modules doing Replacement

Format: “%” module>name [(“ flags “)] “:” [PREFIX “:.”] input [“ | ” alternate] “%”

A Perl module is called as:

```
module::name ( $input, $packet, ‘flags’ )
```

\$input is the “input” string, \$packet is the Slipper::SIP format packet that caused this lookup, and “flags” is the null string unless flags were set, in which case they are passed. If undef is returned, it is treated as the null string.

The prefix and alternate options have the same semantics as for the Berkeley DB lookup.

³Note two colons in the lookup format, but only one is returned.

3.8 File Inclusion

Format: “%<” file_name “%”

Inserts the contents of “file_name” at the current point in the configuration. There must not be anything else on the same line except white-space and comments.

3.9 Embedding

Note that various “% format” expansions can be embedded inside each other, eg:

```
(.*)          %some>function:SUCCESS::%Canonify:$1|$1%
```

Will call the Canonify ruleset on \$1, and pass the results to the Perl function some::function. If that returns a null string, the original \$1 replaces the current string, otherwise “SUCCESS:” is prepended to the returned string and that replaces the current string.

Also note that % format expansions can only exist on the RHS of a rule.

3.10 Error Handling

If any rule in a ruleset returns a string starting with **ERROR:**, it is treated as an immediate error and the ruleset drops out. The parsing of the string is as follows:

```
“ERROR:” Code “:” Reason [ “:” MIME-Type [ “:” Content ] ]
```

Content can include “:” and newlines. Note that the extended mechanism is often used by plugins to return, for example:

```
ERROR:200:OK:text/plain:Diversion Set and In-place
```

There are two special cases:

- If the code is 407 (Proxy Authentication Required), the generated bounce will automatically include an MD5 Digest challenge so the user can authenticate.
- If the returned string is **ERROR:IGNORE** then it is treated as if an error occurred, but no bounce is generated. Primarily useful for modules managing conversations themselves to cause ACK’s and similar to be swallowed. Also useful for ignoring packets from locations you wish to black hole for one reason or another.

3.11 Expansions Format and REGISTRATION

The Expand ruleset is expected to return a list of canonical addresses, separated by a comma (,), semi-colon (;), or tilde (~). The parser starts with a delay of zero, and inserts each item as it comes across it into a structure including when that target is to start ringing. A semi-colon does not increase the delay, a comma increases it by Comma_Delay (by default 5 seconds), and a tilde makes it a backup to the previous route — more on this later.

So if the expansion was “reception,sales,service;tech”, then the reception line starts ringing immediately, the sales line starts ringing five seconds after that, and another five seconds later both the service and tech lines start ringing. The first person to pick up their phone gets the call.

If all ringing lines are invalidated (eg, busied out by a Do Not Disturb feature), then the delays on all remaining calls are reduced by an amount to make at least one start ringing immediately.

The internal Perl function REGISTRATION takes a Berkeley DB file as “subroutine”, and is expansion aware. It looks up the expansions database, and for each entry in the current expansion list, leaves it alone if there is no matching entry in the registrations database. If there is an entry there, it gets replaced with zero or more semi-colon separated entries. It checks each matching entry in the registrations database, and includes it if it is current, and excludes it if it is not current — that is, the registration has expired.

In the example below, we assume the time is currently 225, so everything is still valid except for the tech line at 10.1.1.20 (which someone has unplugged to move from one end of the building to another), and that “callout” is handled in user-to-number.db (looked at by the Route ruleset) as an external route to a paging service.

```
Ruleset Expand
(.*                %expansions.db:$1|$1%
(.*                %REGISTRATION>registrations.db:$1%

server> db expansions.db | grep reception
reception          reception,sales,tech,callout+=10
server> db registrations.db | egrep 'reception|sales|tech'
reception          reception@10.1.1.25;expires=250
sales              sales@10.1.1.35;expires=350
tech               tech@10.1.1.30;expires=300,
                  tech@10.1.1.20;expires=200,
                  tech@10.1.1.40;expires=400
server> check_ruleset Expand reception
(Expand, reception) =>
    reception@10.1.1.25,sales@10.1.1.35,
    tech@10.1.1.30;tech@10.1.1.40,callout+=10
```

So the reception phone starts ringing immediately, then the sales phone after five seconds, then both tech lines simultaneously after another five seconds, and ten seconds after that the “callout” line goes. Note “callout” was not touched as it did not exist in the registrations database. If all registered phones then turn out to be busied out, the paging service will get the call with minimal delay.

The `db` utility can directly be used on the DB file to add or remove registrations. A registration without an `;expires=` tag will be assumed to have an infinite expiry. You can also set `"expires=never"` to achieve the same effect, or `"expires=neg"` which will (apparently) ignore all attempts to process a registration from that source, and will always treat that entry as expired. Actually, it puts the new expiry time after the `"neg"`, so if the negate is reversed the actual expiry time can be recovered. This is often useful when temporarily wishing to treat a phone as off. Also, if the entire value associated with the key is `"IGNORE"`, then all registrations are ignored, and when expanding it is treated as if there is no entry (that is, the key is left alone to fall through to some other code.)

As a modifier to the comma or semi-colon separators, an entry may have a trailing `=` or `+=` entry, which overrides the comma or semi-colon and makes the delay on that entry an absolute time (`=`) or a time relative to the current delay (`+=`) by a number of seconds. For example, if you wanted to make the delay 10 seconds on just one expansion you might enter `"reception,sales,service+=10;tech"`. Note that because the parser works left to right, one element at a time, `"reception,sales,service;tech=10"` probably won't do what you were expecting — the tech line will start ringing 10 seconds after the service line, whereas in the previous example service and tech started ringing at the same time, 10 seconds after the sales line.

Now, a entry separated with a tilde is a backup route. This is only used if the previous route is active and failed — it will not time over on no answer. This is used when simultaneous ringing is not desired, eg, when listing multiple gateways for reaching the same PSTN target. You might specify, for example:

```
+61884252255@closest.gateway.to.target~
+61884252255@closest.gateway.to.source
```

so that you try one gateway, and only try the second if the first returns a non-global error (4xx or 5xx, as 6xx is global). This allows you to fail over on conditions such as no free lines or timeouts due to network outage. You might also have:

```
test@domain.name~reception@domain.name,test@voicemail.domain.name=15
```

Which says to try the phone associated with test start ringing, and only if that phone is busy to start ringing reception — a "Divert on Busy" equivalent. Regardless of which phone is ringing, 15 seconds later it goes to voicemail — a "Divert on No Answer" equivalent. Unlike most PABX systems, this is done based on the original number dialed, allowing for greater flexibility.

The Expand ruleset is all about expanding multiple targets, and the Route ruleset is about routing that to a valid destination. Therefore it makes sense for a `~`-style expansion to also be available in the Route ruleset for purposes of actual routing as opposed to user chasing.

A route that expands to something containing commas is pushed back into the system as if an equivalent expansion had occurred. This means that if the Expand ruleset returns

```
user@domain.name,user-mobile@domain.name
```

and the Route ruleset for user-mobile@domain.name returns

```
0414000000@gw1.domain.name,gw2.domain.name
```

or

```
0414000000@gw1.domain.name,0414000000@gw2.domain.name
```

then Slipper treats it as if the original expansion was:

```
user@domain.name,  
0414000000@gw1.domain.name  
~0414000000@gw2.domain.name
```

Technical note: This isn't actually quite true. The first element of the list returned by Route is not evaluated a second time by the Route ruleset, so as to avoid loops. It is the user's responsibility to make sure this doesn't happen for subsequent elements of the list. Usually this isn't a problem, as in most ruleset styles only unqualified addresses (without @'s) would return such a list, but if you are using a different style you may have to be careful with this feature.

3.11.1 Redirects

If an expansion from a redirect points to a target that has already failed, it is not tried again. This way if two devices call forward to each other, Slipper gives up as both targets will be considered "failed". If there are no other possibilities in the expansion list, Slipper will return a 482 Loop Detected message.

Handling of redirects such as "302 Moved Temporarily" are a bit more complex. When a redirect is received, Slipper determines the expansion list based on what it would have done had the original call been to that target. It then considers the target it received the redirect from to have failed, and adds the new expansion list to the existing expansion list, offset by the delay so far.

For example, assume a call to A expands to "A1,A2", a call to B expands to "B1,B2", and a call to C expands to "C1,C2,A1,A2". A call to A is made. Assuming comma is still the default five second delay, the expansion list starts as "A1 at time 0, A2 at time 5". After one second, A1 returns a redirect to B. Slipper updates its idea of the expansion list to be "A1 has failed, A2 at time 5, B1 at time 1, B2 at time 6". Later when an invite is sent to A2, it responds immediately with a redirect to C. Slipper's expansion list is now "A1 and A2 have failed, B1 at time 1, C1 at time 5, B2 at time 6, and C2 at time 10" - neither A1 nor A2 is added back in.

3.12 Comments

Anywhere a "#" occurs in the slipper.rules file, it, anything after it until end of line, and any white-space preceding it will be considered to not be there. Blank lines (including lines that become blank after the above stripping) are ignored.

3.13 Line Continuation

Any line can use `\` on the end as a line continuation mechanism. Leading white-space on the following line is removed as well as the normal removals. Note comments are removed before checking for `\` as the last character on the line.

Chapter 4

Security

If you are connecting your voice system up to the internet, we strongly encourage you to block all SIP (UDP and TCP port 5060) messages to all devices except Slipper, to force call control to come through it. If your voice system is not connected to the Internet, and you have trustworthy users, then most of this section will not be relevant to you.

Beyond just the Authorisation ruleset, there is some extra stuff going on in the background. When a message is authorised, it is inspected for Contact headers, and those Contact headers are then automatically authorised for this Call-ID only. This allows a call diverted from an allowed number to a disallowed number (eg, reception@iagu.net diverting to a mobile number) to detect the Contact headers in a request and authorise ACK's sent to that URI later.

When sending a message out, the Via header has a branch code added to it. When a response comes back, this branch is tested to make sure it's right. If it's not right, the packet is discarded. The exception to this is if the configuration variable `Random_Cluster` is set to any value, in which case the system assumes it is part of a cluster hidden behind a single IP address with a clustering device that does not associate UDP transactions, or the device it is talking to does not send responses back to the address the request came from and relies on address resolution. In this case the system, if a branch does not match, Slipper falls back to stateless proxy code. It is suggested that forking of calls be avoided in these environments.

As part of its packet decode process, Slipper checks for the presence of nulls in headers and overlength headers. If either of these conditions exists, Slipper fails the packet decode process and drops the packet rather than issuing a reject, as it almost certainly is as the result of some cracking activity and responding is likely to either give the party extra information or, given the ease of forging UDP packets, be some form of smurf (reflection) attack upon some innocent party.

Headers that support being split by commas (such as "Via") are split before checking for over-length headers, so a packet that has been through a long chain of proxies is not adversely affected by this test. Control of the critical point for considering a packet over-length is controlled by the `Max_Header_Length` variable.

If you are not going to use the web interface, or intend to use it for database administration but not for server setup, we suggest removing or commenting

out the line near the top of `slipper.rules` that includes data from the file `slipper.webconfig`.

Note that when a response to a request is received, Slipper restores the original Via headers it received rather than trusting the remote end to correctly echo back the Via headers it received. This closes an obscure hole where a malicious entity can try and get a request sent to them (maybe by calling in an getting put on hold by an operator) and generate a response back with incorrect Via headers, getting the packet sent somewhere unexpected by the SIP proxy. Note that since it could be to any IP on any port, delicate services that don't handle data they don't understand could be attacked via this method.

Chapter 5

Plug-in Modules

5.1 Pickup

“Pickup” is a module designed to implement Call Pickups. It is designed to be the target of something in the Expand ruleset, eg:

```
Module                               Slipper::Pickup

Ruleset Expand
\*70(.*)                             %Slipper>Pickup>pickup(1G):$1%
```

The pickup module accepts an optional number and/or list of flags as an argument . If the number is present, it specifies the number of seconds between when it receives the request and when it should start the phone ringing. In principle it should be left blank, but some devices fail to process messages correctly if they arrive too quickly - notably the Cisco 7905 and 7912 phones.

Currently the only flag is “G”, which tells the Pickup module to restrict calls for which pickup is applicable to only those calls that have at least one device ringing in the same group as the pickup requestor. This is normally used to prevent cross-site or cross-tenant call pickups.

If it is passed input that is either blank or a * (ie, with the above configuration the user entered *70 only), it finds the call that has been ringing the longest and picks it up, subject to a couple of modifiers:

- Only local URI's are selected, so a call to user@remote is ignored.
- Calls for which the pickup requestor are listed in the expansion (but presumably have not started to ring yet) are given a high priority.
- Calls to URI's that after canonification only consist of plus sign numbers are given low preference.
- Calls where at least one device in the same Group as the device initiating the Call Pickup are given a small preference increase.

If the input passed is not either blank or an * then the input is canonified, and only calls to the specified target may be selected. If the input is comma separated, then only calls to any one of those targets are selected.

5.2 MusicOnHold

The music on hold support is implemented as an output filter that does not actually change the packet – it’s there simply to “sniff”, or monitor, the signalling information. When it sees a media pause, or hold, event sent by one device to another, and the second device acknowledge the media pause event but states it is still prepared to accept audio, this module arranges for an external process `rtp_send` to send audio to that device using the Real Time Protocol.

Before it does so, it calls the ruleset `Music on Hold`, passing it:

Group of Hold Initiator “!” Group of Party being placed on Hold “!” Codec in use

If the ruleset returns a string starting `MoH:` then it is assumed the rest of the returned string is the name of an audio file in `.au` format of the same codec type that should be played as music on hold. The example below shows that whenever a caller from the group “Gateway” is placed on hold, if they are using G.711 A-law encoding they will get music on hold obtained from the file `hold-music.au`

```
Module                               Slipper::MusicOnHold

Ruleset Output Filter
(.*)                                  = %Slipper>MusicOnHold>output_filter:$1%
```

```
Ruleset Music on Hold
.*!Gateway!PCMA/8000(|/1) = MoH: hold-music.au
```

If the filename has a trailing `:r` then rather than playing the music from the beginning of the file, it will start from a randomly selected position within the audio file.

If the filename has a trailing `:t=` followed by a type of service string as specified by the `Type_of.Service` family of variables, that IP Precedence / Type of Service value is used instead of the default “5,lowdelay”.

Multiple filenames can be separated with commas. If you have spaces in filenames, consider using the “@=” syntax in the rulesets to preserve spaces. In the example below, the module will randomly pick 10 audio files from the subdirectory “moh”, and play them in a loop, starting from a random position in the first. Note the number of files is optional as it defaults to 10.

```
Ruleset Music on Hold
.*!.*!PCMA/8000 @= %Slipper>MusicOnHold>random_files(10):moh% :r
```

Note that you can include other headers in the decision making process as normal in the RuleSets environment, so you could potentially customise the music on hold to a particular Caller-ID, or give different types of music on hold to users based on the department they dialed.

In an environment with multiple copies of Slipper running, it is suggested that music on hold be generated as close to the receiver as possible. As well as reducing network load, it also reduces the possibility of the codec being changed by an intervening SIP proxy (if multiple codecs are listed in the Hold event, which only a few devices do), corrupting the audio stream fed to the end user.

Iagu strongly advise you make sure of your legal right to play your chosen music. It is common to require some form of licensing fee if the music is not of a certain age, but you should check your local laws and regulations.

5.3 OutboundRegister

The Outbound Registration module looks for one or more items in the list “OutboundRegister”, consisting of a host@username portion of a URI to register, the local username that this registration should target, and an optional number of seconds for the lifetime of the registration, which will default to 3600 seconds - one hour.

```
Module                Slipper::OutboundRegister
Add OutboundRegister: 17473863272@proxy01.sipphone.com sue 600
Add OutboundRegister: andrewr@iagu.net andrewr
ModuleConfig          Slipper::OutboundRegister::config
```

It will actually re-issue registration requests when between 1% and 5% of the registration time (randomly determined) is remaining, to ensure overlap.

Note that as in most cases Authentication is required, you will also need to use the Ruleset “Auth Digest”, which is passed realm ‘!’ Remote_Group ‘!’ remote_ip ‘:’ remote_port - a typical example might be as follows:

```
Ruleset Auth Digest
.* iagu.net ! .* ! (.*)      = $1 ! andrewr ! somepassword
```

Or, if the realm name is not determinable, for example, if it keeps changing outside your control, then doing it by remote IP group may be practical:

```
Ruleset Auth Digest
.* ! .* ! (198.65.166\..*)   = $1 ! 17473863272 ! somepassword
```

Or, equivalently:

```
Ruleset Group
198.65.166\..*              = SIPphone
Ruleset Auth Digest
.* ! SIPphone ! (.*)       = $1 ! 17473863272 ! somepassword
```

5.4 SimpleVM

First one needs to understand this an extremely simple system intended for small-scale, unsophisticated use. Products like Cisco's Unity and Netcraft's Converse provide full-featured voicemail solutions. The most important restrictions are that it only supports G.711 A-law encoding, and that it sends the voicemails as an email to an address, but does not offer a retrieval service.

The following is usually near the top of the Route ruleset.

```
voicemail = internal:Slipper>SimpleVM>handle
```

Then one typically has an expansion pointing to the user and then to the voicemail system.

```
user1: user1,voicemail
```

The Voicemail system inspects the "Diversion" header to see who the call was originally placed to (after canonification), and the calls the ruleset `SimpleVM` `user` with that address. The ruleset should return the following:

Format: user "@" host "!" voicemail_greeting_message.au

Note that the part before the exclamation mark (the target email address) is often the same as the string originally passed in, where SIP addresses have been designed to map to email addresses.

5.5 Sticky

This module tries to make an external caller "Sticky" to a Slipper user, so that the caller is more likely to talk to the person they were talking to before. This feature is primarily required in call centre environments, but is often found to be useful outside those environments.

It has two components - an output filter that observes and records the last username (but not number) a call was connected to, and a rule intended to be placed in the "Expand" ruleset that will return `contact@host`, (note trailing comma) if appropriate.

It is intended that this be used something like the below:

```
Ruleset Expand
(reception) %Slipper>Sticky>prepend_sticky:$1%%expansions.db:$1|$1%
Ruleset Output Filter
(.* ) %Slipper>Sticky>output_filter:$1%
```

So that any calls to `reception` after canonification are checked to see if the Caller-ID has been talking to someone else recently, and if so they end up on the front of the expansion list.

A pairing seen remains valid for up to `Keep_Sticky` seconds, defaulting to 43200, or 12 hours.

If you wish to restrict Sticky to only prioritise calls and accelerate the selection of a target the caller has already spoken too, and not to add a new target into the list (as in the default operation), then add a flag of "accelerate:" followed by the name of a database to check. The input string is looked up in the nominated database (usually `expansions.db`), and the target in the sticky database is only added if the user part of the target appears somewhere in the lookup.

5.6 RingToneSelection

This module allows one to control the “Alert-Info” header, used in different ways by different phones. It is referenced at the beginning of the Codec ruleset, and so is passed `Source_Group` “!” `Destination_Group`. This input is passed through as-is to the “Ring Tone Selection” ruleset, which may return a string starting with `ALERT-INFO:`, in which case all existing Alert-Info headers (if any) are removed and a new one is inserted with the contents specified on the rest of the returned data. If the returned string instead starts with `NULL`, then any Alert-Info headers are stripped (good for cleaning up headers inserted by remote proxies). Any other response is ignored.

```
Module                               Slipper::RingToneSelection
Ruleset Codec                         (.*)                               %Slipper>RingToneSelection>selection:$1%
Ruleset Ring Tone Selection
Internal ! Internal                   = ALERT-INFO: <Bellcore-dr3>
.*                                    = NULL
```

Most phones support the five US Bellcore defined standards, named `Bellcore-dr x` , where x is a number from 1 to 5, with `Bellcore-dr1` being a default ring tone, and the other being variations on stutters in the ring tone.

Some phones support specification of a filename downloaded from some server - in some cases a URL is expected, in other cases a filename from their local TFTP server.

Other phones¹ ignore the contents of the header, but if the header is present behave differently, eg, a stutter in the ring tone, or playing an audio sample twice between pauses instead of only once.

¹For example, Cisco 7940 and 7960's, but not 7905's or 7912's.

5.7 MWIcache

This module intercepts Message Waiting Indicator messages and stores them in a cache file. It then resends the MWI events immediately, and resends on client registration so (a) a device rejoining the network gets the current status, and (b) a device that has had a network failure is eventually notified.

If the database is modified to have an entry of “never” against a user, then MWI status is not sent on registration. If an MWI NOTIFY is received by Slipper, regardless of the content an “no Message Waiting” indicator is sent.

```
Module                               Slipper::MWIcache
Ruleset Expand
(.*)                                 %Slipper>MWIcache>expand(mwi-status.db):$1%
Ruleset On Register
(.*)                                 %Slipper>MWIcache>on_register(mwi-status.db):$1%
```

Note also that if upgrading from an earlier version of Slipper, your Force Socket ruleset should look like this:

```
Ruleset Force Socket
Slipper ! .*                         = SOCKET:NonLocal
LocalHost ! LocalHost               = OK
LocalHost ! .*                      = SOCKET:NonLocal
.*                                   = OK
```

5.8 ReferToFixup

This module exists to work around Cisco's IOS SIP stack not honouring the host portion of the URI and not supporting alphabetic usernames. The output filter stores the original URI, and creates a new URI with itself as the host part and a unique string as the user part. When that string is received in an INVITE, the "canonify" function will convert back to the original URI.

As well as removing the need for the "transfer-rewrite" ruleset, it also solves the problem of IOS passing transferred call legs back out the PSTN in a hairpin toll call instead of using the VoIP network for toll-bypass.

```
Module                Slipper::ReferToFixup
Add Tick:             Slipper::ReferToFixup::tick
Ruleset Canonify
(D\d{10})              %Slipper>ReferToFixup>canonify:$1%
Ruleset Output Filter
(Gateway)             %Slipper>ReferToFixup>output_filter(D):$1%
```

This is usually combined with the Fixup_Refer_To variable being activated.

When clustering, it is suggested that each Slipper install have its own unique prefix, and the IOS devices have one dial-peer per cluster member. This avoids the temporary URI being presented to a cluster member other than the one that generated the URI.

Typically, the replication.rules file has the line:

```
Set Transfer_ID:      D1
```

And the Output Filter ruleset line becomes:

```
(Gateway)             %Slipper>ReferToFixup>output_filter(${Transfer_ID}):$1%
```

5.9 HideAddress

This module is used to hide some internal addressing information behind another address - this is sometimes required by a VoIP carrier not adhering to the SIP standard and caring about header contents instead of just the tags. It attempts to hide selected information as the address supplied with a view towards making these systems work.

As multiple internal addresses may be hidden behind one single address, it maintains a cache indexed by Call-ID. It is possible for new inward-initiated requests (without a previously known Call-ID) to be routed to the most recent activity instead of where it might otherwise be expected to go, although first preference goes to the Contact address supplied in registration requests.

```
Ruleset Input Channel
(SIPphone)      %Slipper>HideAddress>input_channel\
                 (17473863272@proxy01.sipphone.com):$1%
```

```
Ruleset Output Channel
(SIPphone)      %Slipper>HideAddress>output_channel\
                 (17473863272@proxy01.sipphone.com):$1%
```

Associated variables:

- **HideAddress_Grace_Time**
Records for calls are eventually expired after this many seconds. The default is 24 hours.
- **HideAddress_Host**
Local hostname used to overstamp the Contact header. Defaults to `Server_Name`, may need to be changed in some NAT or split-DNS environments.
- **HideAddress_Host_Group**
The group name supplied to the ruleset is appended to the string `HideAddress_Host_..`. If this variable exists, then for messages to that group only the contents of the variable will take precedence over `HideAddress_Host` and `Server_Name`.

This is known to be required with:

- The SER proxy server can be set to auto-generate the realm name to use in Digest challenges from the headers. In this case it prevents one from receiving a challenge for one's own realm name.
- Certain front-end border control software rejects calls where the user part of certain headers is non-numeric.
- A number of SIP carriers (eg, systems based on BroadWorks) are known to do username matching based off the contents of the From header instead of the username in the authentication header.

5.10 Programming

At its simplest, a plug-in module is passed some value and returns some value. The ruleset calling it passes some parameter, and optionally some ()-bounded flags. Slipper will also add in a reference to the internal data structure of the packet under consideration, so one can make decisions on headers, the message body under consideration, or packet meta-data. The return value replaces the call to the plugin, subject to the normal rules.

The simplest plug-in is the example plugin module, `Slipper/Example.pm`, that looks like this:

```
package Slipper::Example;

sub example {
    my ($input, $packet, $flags) = @_;
    return "ERROR:404:Not Found" if ($input eq '123');
    return $input;
}
1;
```

And is called from a Ruleset with a line like:

```
(.*)          %Slipper>Example>example(flags_passed_as_string):$1%
```

For progressively more complicated plugins, see `Slipper/ReturnTransfer.pm` and `Slipper/Pickup.pm`. The first looks up headers in the packet to see if it needs to modify its return value, and most of its complexity comes from security checks after the major decision has been made, trying to make sure people don't place calls to places they shouldn't by faking a return address on an otherwise valid call transfer. The second actually modifies Slipper's internal data structures to add a new element to an already expanded set of targets, effectively pretending it was there all along. It then leaves it up to the base Slipper system to handle the conversation from there.

5.10.1 Packet Format

It is a standard Perl hash with the following elements:

- **Header** points to a hash containing the names of all headers in the packet. Each one of these points to an array of strings, one per header line of that name.
- **Body** points to an array of strings that are the lines of the message body.
- **Method** — the Method name in a request packet.
- **URI** — the URI in a request packet's start line.
- **Code** — the numerical code in a response packet's start line.
- **Reason** — the reason string in a response packet's start line.
- **Version** — the version number in the start line.
- **Remote_Group** — the value returned by the "Group" ruleset for this packet. If the plugin is called from an "incoming" ruleset, this will be the Group the packet came from, while if it's an outgoing packet it will be the Group the packet is headed for.
- **Remote** — Where the packet came from, or is going to, as appropriate. Format is IP_address ":" Port
- **Alt_Remote_Group** — in an outgoing packet, this is copied from **Remote_Group** from the incoming packet.

5.10.2 Call Termination Modules

In some cases it is required to terminate a call inside Slipper rather than passing it through to some other end device, such as for voicemail. This can be achieved in the Route ruleset using the `internal:` prefix. Note in this case the module is not bound by `%`, which forces interpolation within the ruleset.

```
voicemail = internal:Slipper>HelperApp>handle(ha/leave-voicemail)
```

In this case, when normally a packet would be sent, instead the module is called with the arguments necessary for the module to access the `$calls` hierarchy. The actual route information is also included as a shortcut (many modules only need to modify their own information, and not other forks of the same call), and the current time is included as well to potentially reduce the number of system calls required.

```
sub handle {
    my ($call_id, $uri, $route, $route_info, $time, $module_args) = @_;
    ...
}
```

Note that in this case `$route_info = $calls -> {$call_id} -> {$uri} -> {$route}`.

When responding to packets, use `Slipper::Support::respond_with`, which will generate a response packet and re-inject it back into Slipper's internal processes.

```
Slipper::Support::respond_with ($route_info, $code, $reason, @extras);
```

`@extras` is of the form `('Key1', 'Value1', 'Key2', 'Value2', ... 'Body')`, with the trailing message body being optional.

When the "Group" ruleset is called, it will be with `SLIPPER_INTERNAL` rather than an IP address. The default `slipper.rules` will map this to a group name of "Slipper".

5.10.3 Plugin Tips

The “Codec” ruleset is the only one in the base Slipper system that modifies the contents of the message body, and then only if the Content-Type is `application/sdp`. This is also the only ruleset you can guarantee will get a go at every packet going through the system, so if you want to modify headers in every packet (and don’t want to use an input or output filter), you may have a plugin invoked from the Codec ruleset that just returns whatever it was passed after it has performed the required manipulation.

Note that in rulesets for a variety of security reasons variables are substituted before the Perl evaluation takes place, and references to plugins are evaluated after the evaluation. This is primarily to prevent those who know this is a Slipper server and wish to try and embed variable references in headers from gaining information about what those variables are set to.

If you wish Slipper to perform variable substitution for you in a module (often good for portability), you have to invoke it directly. Call `replace_in_action` as follows:

```
$new = Slipper::RuleSets::replace_in_action
      ($main::config -> {'Rules'}, $variable_name, $packet);
```

The first variable may be replaced if you’re running multiple ruleset databases. If you wish to have a separate ruleset database from the main one, you can do this by calling `Slipper::RuleSets::read_rules ($hash, $filename)`; which returns the number of errors encountered in parsing, hopefully zero. It is done this way so you can load multiple files into the one hash through separate invocations.

The “%” used in ruleset declarations to indicate ruleset calls, database lookups, and plugin references is actually converted to a null prior to processing. If you wish to arrange for further lookups, when returning a value one should return a null in the string rather than a “%”. Note that Slipper rejects inbound packets with headers containing nulls, so this protects against an externally-crafted packet attempting to invoke inappropriate plug-in modules.

If you wish to change Slipper’s internal database as the `Slipper/Pickup.pm` module does, you are advised to (a) look at the code in it for an example, and (b) use `SIGUSR1` to get dumps of Slipper’s internal data structures a lot when debugging, preferably even before starting coding, to familiarise yourself with how it works.

Note that Slipper does not cache the results returned by rulesets and plugins as it cannot know the validity period or even if the result has been obtained based on conditions it knows nothing about. Therefore, when appropriate, we strongly recommend that your plugins do their own caching.

5.10.4 Example Uses

Some things we've seen people use plugins for:

- Choosing different Caller-ID's based on database lookup as to the user's identity and mapping it back to the reception of their department.
- Changing the expansions on a rotation and ordering them so the target that has gone the longest time since getting off a call (by scanning Slipper's internal state database) is at the front of the expansion. For those still on calls, the one who has been on the current call longest gets presented with the call first.

5.11 Programming Helper Applications

Helper Applications are those applications external to the Slipper core that can provide useful services and functions to users, in much the same way as a web browser's helper applications (such as Adobe Acrobat) can provide assist in viewing some formats. In the case of VoIP, these can be such things as voicemail or IVR's (Interactive Voice Response) that have a menu the user accesses through DTMF tones through the phone keypad.

The Slipper distribution provides a programming framework for Perl programmers to be able to construct such systems.

- `new_stream (ip_details, packet [, dtmf_payload])`

Create a new stream object. Note that the first two arguments passed on the command line to a helper application are the IP and port of the remote end and a Base64 encoded copy of the INVITE packet, so calling it as below suffices for most users.

The optional `!tt dtmf_payload` parameter indicates the RTP payload type of RFC2833 telephone events. This is mainly provided for backwards compatibility or when knowledge of unusual systems is required - if not presented, the HelperApp framework will attempt to determine this from the session description in the INVITE packet.

An object is returned on success. On a failure a text string is returned describing the failure.

```
$stream = Slipper::HelperApp -> new_stream (shift, shift);
```

Note that if a valid executable is specified as the route target, but with trailing fake-filename, that extra information will be passed as the third command line argument, after the IP details and the Base64 representation of the packet.

- `report_port [(contact [, codec])]`

The port the helper application is going to use for audio is reported back to Slipper so it can complete the SIP dialogue. It sends `PORT:16384`, where 16384 is the port number used. Instead of returning this, an application may return an error of the form `400: Bad Request` or similar, which will be reported back in a SIP response as the Code and Reason for the call failing.

Separating this out from the `new_stream` subroutine allows an application to use the parsed data before deciding if it will accept the call, and a method of providing useful feedback if it doesn't.

Note that other messages received (eg, warnings or the program otherwise failing) will be recorded in the active Slipper log file and a 500 Internal Server Error returned to the requestor.

By default the system will send "helperapp" as part of the Contact header. An alternative default can be used by setting the `HelperAppContact` variable. If the optional `contact` setting is passed, that will be used instead.

If the string “AUTO” is passed, the HelperApp will try and set a contact header appropriate to the program being called.

If the optional `codec` is passed, it should be a number representing the codec to be used for this call, as per the codecs specified in the received INVITE. The default is 8, for G.711 A-law.

- `play_audio (audio_data, listen)`

`audio_data` may be actual audio data in audio/basic (.au) or audio/wav (.wav) formats, or may be a filename of such a file. It will play out this file to the requestor.

The `listen` parameter is optional. If not set, it is assumed to be true, and `play_audio` will terminate if a DTMF tone is received and the return value will be the DTMF character. If set to a false value, the audio ployout cannot be interrupted. If set to a string starting with a :, then only DTMF characters present in the remainder of the string will interrupt audio ployout.

If no interrupting DTMF is received, it will return the empty string, unless the other end hangs up in which case it will return the string BYE.

- `record_audio ([format [, timeout [, maximum_length]])`

Record audio from the other end, up to a maximum number of seconds as defined in `maximum_length`. If no audio packets are received due to silence suppression or similar factors for a number of seconds equal to `timeout` audio recording also ceases.

The string `format` defines the format of the resulting audio file. If it is “au” the file will be in audio/basic format, if it is “wav” it will be in audio/wav format. Normally the codec will be the same the audio was received in, but if the string passed is “wav/linear” then if possible it will be converted to a 16-bit linear format.

All the parameters are optional - it will assume audio/wav format, a 5 second timeout, and 60 seconds maximum length.

Returned is an array consisting of a DTMF value and the audio file. The DTMF value is a number, letter, the empty string, or the string BYE as per `play_audio`. If no audio is collected the audio file value will be undefined, otherwise a file will be returned.

- `send_dtmf ([tone [, length [, volume]])`

This will request via RFC 2833 that a DMTF tone of `tone` type be sent, of duration `length` seconds with a volume of `volume` (the volume is as specified in RFC 2833).

All the parameters are optional - it will assume a “7” tone with a length of .24 and a volume of 10.

This will return the string BYE if the call is hung up, otherwise it will return the empty string.

This function is normally only used to communicate with other automated systems as not all end devices will correctly play a DTMF tone as instructed by RFC 2833, even if they do correctly send them when a key is pressed.

- `collect_dtmf ([wait_for])`

Collect and return a DTMF digit. If one has not been received in `wait_for` seconds (defaults to 30) then return the empty string. If the call is hung up, return the string `BYE`.

- `find_vm_target`

Inspects the request packet for a Diversion header and from that tries to determine both the email the voicemail should be sent to and the audio file to use for the greeting. If successful, the destination email address will be set in the variable “VM To” of the stream, and the target audio file will be set in the “VM Greeting” variable.

It will look up the address specified in the Diversion header with the “Simple VM User” ruleset. If this returns a string with an “!” present, it will be deemed to be in the format “emailaddress!audio.file”, otherwise it will be assumed to be just the email address the voicemail should be sent to.

If the specified audio file does not exist, or one is not specified, it will try to find a suitable one in the subdirectory “vm/”. It tries to find a file with the extension “.wav” or “.au”, with the extension preceded by the full email address, the user portion of the email address, or the string “default”.

Returns the string “200: OK” if it found a voicemail target, otherwise it returns an error to be returned to the requestor.

- `send_vm (audio_data [, vm_to [, subject]])`

Sends an audio file as an email. `audio_data` is usually the data returned as the second list element of the `record_audio` function. `vm_to` defaults to the stream variable “VM To” normally set by `find_vm_target`. If neither `vm_to` nor the stream variable “VM To” are set, it will call `find_vm_target` to see if a result can be obtained.

`subject` defaults to “Voicemail for %t from %f (%r)”.

- %t is replaced by the address the voicemail is addressed to. This is especially useful if an IVR is sending it to an alias such as “sales”, so that the recipient can tell for which address the voicemail was intended.
- %f is replaced by the SIP address of the requestor, as determined by the function `vm_get_from`. This is effectively the Caller-ID.
- %r is replaced by reason for the diversion, as determined by the function `vm_get_reason`.

The function will return a string starting with “200:” on success, or return an error message.

- `transfer (target)`

`target` specifies an address this call should be transferred to. If no domain is present, the local “Server_Name” will be added.

Returns the code of the response to the REFER request to transfer the call, or 500 if an internal error occurred. Any number not between 200 and 299 inclusive should be considered a failure.

- **send_mwi** (*target*, *status*, *account*)

Sends a Message Waiting Indicator “NOTIFY” to the designated target. *status* should be “yes” or “no”. *account* is optional and specifies the URI the voicemail should be collected from - currently most devices ignore this.

- **media_inactive**

This returns true if `receive_end_of_media` has been called, usually because Slipper has signalled to the HelperApp that a BYE has been received - that is, the other end has hung up. Helper applications should check this during long operations. Note that if an application does not check this and tries to go into an infinite loop waiting for DTMF or some other event, the framework will detect when the string BYE has been returned around 156 times and abort the process as a defensive measure.

Following is a list of lower-level functions that may be utilised directly by programmers wanting to do something unusual.

- **read_audio_file** (*audio_data*)

Creates the audio data structure that is then passed to `send_rtp`. If *audio_data* is a file, it uses the `mmap()` function to map it into memory directly from the disk, rather than reading it in to memory. This is done to increase speed (no delays reading it in) and also to reduce the virtual memory impact of potentially having a busy IVR system loading the same audio files into multiple processes.

Once it has access to the audio file, it then inspects it for format (audio/basic or audio/wav) and determines the offset of the actual audio data, and parameters such as codec type, size of each RTP packet, etc, and then returns an audio structure containing all this information.

Normally `play_audio` calls `read_audio_file`, `send_rtp`, and `close_audio` in turn, but some users may wish to write their own equivalent to tweak the behaviour.

- **send_rtp** (*audio_structure*, *listen*)

audio_structure is an internal structure created by `read_audio_file`, and *listen* is set to true (to which it defaults) if you wish playing of audio files to be able to be interrupted by DTMF events. If *listen* is set to a string starting with a :, then only DTMF characters present in the remainder of the string will interrupt audio playout.

It returns the same values as `play_audio`, since that function is in fact just passing back whatever `send_rtp` returned to it.

- **close_audio**

If `read_audio_file` opened a file via direct virtual memory map, this function unbinds the mapping. It performs no function otherwise.

- `decode_dtmf (rtp_packet)`
Inspects the `rtp_packet` and sees if a valid DTMF event is found that hasn't already been reported (many clients send duplicate events for reliability purposes). Returns a string representing the DTMF event if one is found, or the undefined value otherwise.
- `vm_get_from`
Inspects the Diversion header of the packet and returns a sanitised version of the "From" header, or the string "unknown" if a valid "From" address cannot be found. Usually called by `send_vm`
- `vm_get_reason`
Inspects the Diversion header of the packet and returns the reason for diversion, or the string "divert" if a reason cannot be found. Usually called by `send_vm`
- `receive_end_of_media`
This function is called by the signal handler that receives the message from Slipper (a SIGUSR2) that the call has ended. People writing library functions for use by other may wish to call this when it is known the call is effectively over even if BYE's have not yet been sent.

A simple voicemail solution might be rendered as follows:

```
#!/usr/bin/perl -w
use strict;
use Slipper::HelperApp;
my $stream = Slipper::HelperApp -> new_stream (shift, shift);
if (! ref $stream) {
    print $stream . "\n";
    exit 0;
}
my $return = $stream -> find_vm_target;
if ($return !~ /^200/) {
    print $return;
    exit 0;
}
$stream -> report_port;
$stream -> play_audio ($stream -> {'VM Greeting'});
$stream -> play_audio ('vm/pling.au');
my ($dtmf, $message) = $stream -> record_audio;
exit 0 if (! defined $message);
$stream -> send_vm ($message);
exit 0;
```

5.12 State-IVR

5.12.1 Introduction

`state-ivr` is a program utilising Slipper's HelperApp framework to supply a higher-level interface to interactive voice response (IVR) systems.

5.12.2 Operation

Each instance of the IVR should have two files, the first is a symbolic link to the main `state-ivr` program, the second is a file of the same name with the string `.state` appended. This second file is an IVR state file, as described in the next chapter.

In a default Slipper configuration, this IVR is then referenced by the string `ha-name_of_ivr`, normally in either the `number-to-user` database of the `expansions` database.

Each time the IVR is started, it creates a log file in the directory `/usr/local/slipper/log`, with a filename composed of the following elements separated by dots:

- Name of the IVR.
- Timestamp of the start time, digits only, listing year, month, day of month, hour of day, and minute of hour.
- The process ID of the IVR.
- The Caller ID of the received call. If Caller ID is not available, it will record `"anonymous@identifier_of_gateway"`.

First in that file is a copy of the SIP INVITE packet that requested the call be set up, followed by logging information from each state as the IVR progresses.

5.12.3 State File Format

The state file consists of a number of lines starting with a keyword, followed by configuration specific to that keyword. Blank lines are allowed. Any text after a hash (#) on a line is treated as a comment and ignored.

Keywords may optionally be followed by colons (:).

- **State** *Name_of_State*

The name of each state may include alphanumeric, punctuation excluding commas, and spaces.

Each state has a group of actions defined by keywords associated with it. Progress through the IVR is measured by progressing from one state to another.

The first declared state in any configuration file is the initial state the IVR enters.

Please note that keywords within a state are processed in the order they appear in this keyword list, not necessarily in the order they appear in the configuration file.

- **Set** *Variable_Name Separator Modification*

Variables may be set through the course of the IVR. The **Set** keyword allows setting new variables and some basic manipulation of existing variables.

The Variable_Name may contain alphanumeric and underscores.

The Separator is one or more of the characters = > :, spaces, and/or tabs.

The Modification may be one of the following:

- ++

Add one to the current variable. Set to one if the variable does not already exist.

- --

Subtract one from the current variable. Set to -1 if the variable does not already exist.

- += *value*

Add the value to the variable. Set to the value if the variable does not already exist.

- -= *value*

Subtract the value from the variable. Set to zero minus the value if the variable does not already exist.

- *value*

Set the variable to the value.

A “value” may be either a numeric value or the name of another variable prefaced by a dollar sign (\$).

Set commands are evaluated as the IVR enters that state.

- *Audio Output*

A state may have any number of Audio actions associated with it to play audio to the caller. If more than one is present, they are played in the order they appear in the original configuration file for that state.

- *ReadCharacters: Value : Directory*

- *ReadDigits: Value : Directory*

- *ReadLetters: Value : Directory*

The above three are functionally identical, they iterate over each byte of the contents of the value (which may be a variable name prefixed by \$), and for each one play the file of that name appended with '.au' in the named directory.

For instance, if the variable "foo" contained the value "123", then

Audio: ReadDigits:\$foo:audio/numbers

would play the audio files "audio/numbers/1.au", "audio/numbers/2.au", "audio/numbers/3.au" in sequence.

- *ReadNumber: Value : Directory*

Similar to ReadDigits, except it understands that it is a number rather than a string of digits. The named directory should have ".au" files with the following file contents:

million.au thousand.au hundred.au 0.au 1.au 2.au 3.au 4.au 5.au 6.au
7.au 8.au 9.au 10.au 11.au 12.au 13.au 14.au 15.au 16.au 17.au 18.au
19.au 20.au 30.au 40.au 50.au 60.au 70.au 80.au 90.au

For instance, if the variable "foo" contained the value "123", then

Audio: ReadNumber:\$foo:audio/numbers

would play the audio files "audio/numbers/1.au", "audio/numbers/hundred.au", "audio/numbers/20.au", "audio/numbers/3.au" in sequence.

- *ReadAmount: Value : Directory*

- *ReadAmount.currency: Value : Directory*

- *ReadAmount.currency_subunits: Value : Directory*

Similar to ReadNumber, except that it explicitly deals with currencies. It may optionally specify the name of a currency and currency sub-units, which defaults to "dollars" and "cents".

The directory must also have the audio file "and.au", and files for any currencies used, eg, "dollars.au" and "cents.au".

For instance, if the variable "foo" contained the value "123.45", then

Audio: ReadAmount:\$foo:audio/numbers

would play the audio files "audio/numbers/1.au", "audio/numbers/hundred.au", "audio/numbers/20.au", "audio/numbers/3.au", "audio/numbers/dollars.au", "audio/numbers/and.au", "audio/numbers/40.au", "audio/numbers/5.au", "audio/numbers/cents.au" in sequence.

- Anything else.

Anything not matching one of the above patterns is assumed to be the name of an audio file to be played. An audio file may optionally have after it a semi-colon (;) followed by one of the following:

- * **nointerrupt**
Pressing a DTMF will not interrupt playing this audio file. Often used for user error messages.
- * **listen= *list***
The lists may consist of one or more digits, star (*) or hash (#). Only these characters may interrupt playing the audio file. Again, often used for user error messages, but in this case where you want them to press a button other than that they have been pressing to proceed to some other state.

Note that unless otherwise over-ridden, receipt of a valid DTMF digit interrupts playing audio files.

- **0-9, ,# *New_State***
After this keyword is the name of a state to proceed to if this character is received.
- **Immediate *New_State***
After the audio (if any) for this state has been played, proceed to the named new state, without waiting for any DTMF digits to be pressed. Note that if DTMF digits are received while audio is playing, and an action for that DTMF has been defined, then that will be the next state instead of the state named by the Immediate keyword.
- **Default *New_State***
If no DTMF is received after the audio (if any) for this state has been played, the IVR waits 10 seconds for number input before proceeding to the Default state. If no Default state is defined, the current state is repeated.
- **If *Value_1 Comparison Value_2 New_State***
A value must either be a number or the name of a variable containing a number. Move to the named state if the specified comparison evaluates to true,
The comparison may be any of:
 - =
- ==
Only if the two values match.
 - <>
Only if the two values do not match.
 - >
Only if the first value is greater than the second value.
 - <
Only if the first value is less than the second value.
 - =>
 - >=
Only if the first value is greater than or equal to the second value.

- <=
- =<

Only if the first value is less than or equal to the second value.

- **Collect:** *Digits Variable New_State*

“Collect” is not a keyword that belongs to a state, but is instead a special state. In this state, it collects the number of digits specified, stores it in the named variable, and then shifts to the next state specified.

If the Digits area includes a hash or a star, then receiving that character terminates collection of digits before the specified number of characters.

So to collect a credit card expiry date, one might specify:

Collect:4:Expiry:New State

But to collect an amount one might specify:

Collect:12#:Amount:New State

on the assumption the IVR won’t handle amounts containing more than 12 digits.

- **Transfer** *Target Fallback_State*

“Transfer” is not a keyword that belongs to a state, but is instead a special state. It attempts to transfer the call to the specified target, which may be a variable name prefixed with \$. If the call transfer fails, it proceeds to the designated state.

- **Program/Module/Function/Error_State**

“Program” is not a keyword that belongs to a state, but is instead a special state. It is used to call out to external Perl modules, for example to perform credit card processing or database verification of a customer ID.

- Module

The module name, for use in a Perl “use” command.

- Function

The name of the Perl function. It will be called with two parameters, the first is a hash of the current variables, which may be modified by the function. The second is a Slipper HelperApp “stream” object, in case the module wishes to perform some low-level audio functions, such as providing hold music while it performs a check that takes a long time.

The function should return two parameters, the first being the next state the IVR should proceed to, and the second being a text string that is added to the IVR’s log file.

- Error_State

Should the module fail, the IVR will proceed to the named state as a fallback.

- **MaxStateCount** *Number*

This is a safety mechanism against a user (for example) not hanging up a phone correctly, or trying to break the system. By default, if any state is entered more than 50 times, the IVR will hang up the call. This number can be changed from the default of 50 to some other value by this command. It is a global command that applies to all states.

- **Mask** *Variable_Name Mask*

Often contents of variables are written to log files. This global command that applies to all states can be used to mask some part of the variable, for example to obscure credit cards numbers for regulatory compliance.

Masks may be of the following forms:

- Digits,Substitution,Digits

Replace all except the first few and last few digits with the substitution character. For example, “2,x,2” applied to “12345678” produces “12xxxx78”.

- Digits,Substitution,-

Replace all except the first few digits with the substitution character. For example, “2,x,-” applied to “12345678” produces “12xxxxxx”.

- -,Substitution,Digits

Replace all except the last few digits with the substitution character. For example, “-,x,2” applied to “12345678” produces “xxxxxx78”.

As a safety measure, if a mask of an unknown or incorrect format is specified, the whole variable is masked with the character “x”.

Chapter 6

Operational Information

6.1 Reconfiguration

Sending a `SIGHUP` to Slipper will cause it to re-read its configuration. State is kept intact, but any new calls (or new legs of existing calls, such as transfers) will be processed under the new configuration. Bound ports can be changed: ports that are in the old and new config will be kept open, not closed and opened, so there is no window for packets being bounced. Note that if a socket is being actively used as part of a call leg still being processed, that socket is held open until that call leg is completed, at which point that socket is closed.

You can also call `slipper.sh config`, which will arrange to send a `SIGHUP` to the current running Slipper for you, assuming the PID file is in a location the `slipper.cfg` or `slipper.rules` and `slipper.sh` agree upon — this may be relevant if you have multiple copies running.

6.2 Internal State Dump

Sending a `SIGUSR1` causes Slipper to dump its internal state information in Perl-eval'able format (almost human readable) to its current log location (`stderr` if it hasn't been directed elsewhere). This is primarily to support development efforts by writers of plug-in Perl modules.

6.3 Clustering and Replication

A cluster of servers is a group of Slipper servers sharing the same `Realm_Name` and usually with DNS SRV records for that `Realm_Name` pointing to the group of Slipper servers. Generally the configuration and all databases are kept in line by some process, usually by designating one as the “master” (usually running `slipper-admin.cgi`), and the others pull updates periodically via `rsync` or some similar method. Since Registration information must be replicated immediately, use of the `Replication` and `Replication_Key` settings is suggested. A server receiving a successful registration will replicate it to the other servers in its `Replication` list, with the network packet having an MD5 signature based on the key, the packet contents, and a timestamp. All servers must have synchronised timestamps as packets received that were sent more than 5 seconds ago (or more than 5 seconds in the future) will be rejected to prevent replay attacks. If required, the acceptable time variance can be tuned with the `Replication_Tolerance` variable.

If packets may be received from an alternate IP address of a cluster member, that IP should be added to the `Replication_Aliases` list. Replication packets received that do not come from an IP address in either the `Replication` nor the `Replication_Aliases` lists will be discarded before any processing (including key checking) is performed.

Note that although these packets use the normal operating Slipper port, they are not in SIP format, but rather a Slipper-proprietary format both to reduce bandwidth consumption (for wide-area clusters) and to bypass normal SIP processing and thus reduce the CPU load that may otherwise become unacceptable in large clusters. Thus replication may be blocked by SIP-aware firewalls as an invalid packet.

When replicating databases, `rsync` is preferred over `scp` as the latter just overwrites the existing file, while the former “stages” the copy, transferring into a temporary file and then replacing the original when complete. This reduces the exposure to problems caused by missing entries, and also the possibility of the BerkeleyDB C code inside Perl of segmentation faulting my trying to read part of the memory-mapped file that doesn’t exist yet - leading to Slipper unexpectedly terminating with very little in the way of log files.

We also suggest that “slave” machines pull updates, rather than the “master” push updates. This way, if the master becomes unavailable for some reason and one of the slaves is designated the new master, any changes made on it are not obliterated by a push when the original master is restored. One may also want to protect against multiple `rsync`’s running concurrently in case some large files saturate a low speed link, for example updating music on hold files. Running the following script on the slaves from cron should be sufficient.

```
#!/bin/sh

MASTER="master.some.domain"

if {
    ps auxww | grep rsync | grep ${MASTER} \
        | grep -v grep > /dev/null
} ; then
```

```

        exit 0
fi

/usr/local/bin/rsync -aS --delete \
    --exclude=db/registrations.db --exclude=log --exclude=status-file \
    --exclude=etc/slipper.webconfig --exclude=etc/replication.rules \
    --exclude=vm \
    ${MASTER}:/usr/local/slipper /usr/local/
/usr/local/bin/rsync -aSu \
    ${MASTER}:/usr/local/slipper/vm /usr/local/slipper/
/usr/local/bin/rsync -aSu \
    /usr/local/slipper/vm ${MASTER}:/usr/local/slipper/

```

If you do not have many updates, designating a master may not be necessary — just `rsync` with the “-y” option (copy younger files over older files only) in both directions may be acceptable.

Since the `slipper.rules` file is usually replicated, but the `Replication` list should not include the server itself, it is advisable to include the replication settings in a separate file included from the main configuration. With this in mind, the default Slipper rules file includes the file `replication.rules` and the distribution includes this as a zero-length file, so this part of the configuration does not get replicated between servers.

6.4 Single-server Clustering

Slipper itself is single-threaded from a CPU perspective, although it is written to do as many things at once as possible without blocking. If you have a multi-CPU server, or have plug-in modules that block or have large amounts of disk wait, you may wish to run multiple copies of Slipper on the same server to take advantage of the extra CPU time available to you.

Set up whatever form of IP aliasing your system uses, and run multiple copies of `slipper` bound to different IP addresses. In this configuration, Slipper can share all config information (such as Berkeley DB files, including the registration database) without problem. If your configuration is multiple front-end servers running Slipper with a back-end file server to hold shared databases which require writing by all copies of Slipper (typically the Registration database only), file locking must be set up or the registration database may become corrupted. The standard Berkeley DB interface will sometimes cause the Perl interpreter to crash when accessing a corrupted database. We suggest avoiding this by having one registration database per server and using registration replication.

6.5 Ping with SIP

Slipper distributes with a small program called `pingsip` that sends an empty OPTIONS request to the designated SIP address. In an exit code sense, it returns true on receiving a 2xx response, and false on receiving any other response or a timeout.

This can be used for testing remote connectivity to other Slipper installations or to test device connectivity.

The standard usage message is below

```
Usage: pingsip [-options] target
-v|--verbose      Show all packet details
-q|--quiet        Return exit code only, no output
-t|--timeout #    Set timeout to this many seconds
-r|--requests #   Send at most this many requests
```

6.6 Web Interface

The file `slipper-admin.cgi` is a basic web interface to controlling the databases Slipper uses. It is primarily a tool for day-to-day management, and is not for modifying behaviour in `slipper.rules`.

It looks for files with the suffix `.db` in the directory `/usr/local/slipper` unless you modify it to do otherwise — the directory is set in a variable at the top of the program to aid in modification. It also has the ability to modify Cisco phone configurations for 7905, 7912, 7940, and 7960 models. For the 79x0 phones it looks for files with the prefix `SIP` and the suffix `.cnf` with the middle being an ethernet address, in the directory `/usr/local/tftpboot`. For the 7905 and 7912's it looks for the `ld-template.txt` template file and files with the prefix `ld` and suffix `.cnf`, and generates configuration files from those. It assumes the configuration tool supplied in Cisco's software image distribution ZIP file is present in `/usr/local/tftpboot/c7905`. The idea is to unpack the distributions in that directory and work from there.

The web interface can also make changes to some of the Slipper system-wide variables, primarily with a view to quick setup, by clicking on **System Config**. This saves variables in the file `slipper.webconfig`, which is included from the main `slipper.rules` file just after variable settings, so the settings from the web interface will override these variables.

It is expected that you will take whatever steps you deem appropriate to secure this web interface. The script assumes that whatever web server is invoking it has already made sufficient security checks (authentication, IP address, encryption, whatever you feel is necessary) to satisfy your requirements.

6.7 call-accounting

The command line utility `call-accounting` operates over Slipper log files of debug level 2 or above to generate tab separated value output (“TSV”) of call detail records. It can either accept log files on standard input or listed on the command line.

The output columns are as follows:

- **Group**
The Group the call came from. This is not included by default, to get it specify the `-g` option.
- **Auth**
The username used in successful authentication of the request. Shows as “-” if authentication was not required - for example if it came from an internal IP for which authentication isn’t required, a gateway, or was to a published target, eg, `reception@iagu.net`.
- **From**
Contact header in the original INVITE request, so you get the IP of the requestor as well as the name or Caller-ID number.
- **Dialed**
Shows the URI as received on the original INVITE.
- **Connected**
Shows the Contact header of the 200 OK response of the device that answered the call. This differs from the Dialed field because it shows the IP of the device so you can tell which gateway a call went out, and in the case of expansions, will show which user actually answered the call. Combining the two allows you to see (for instance) that “someone dialed Alice’s number, but Bob answered”.
Shows “Cancelled” if the caller cancelled the call (hung up) before it was connected.
Shows the error message if one was returned, eg, “604 Does Not Exist Anywhere” for a wrong number.
- **Initiated**
Time the call was first presented. The time format used is `YYYYM-MDD_HHMMSS.mmm`.
- **Answered**
Time call was answered or the request for a call was terminated (by cancellation or error). This time minus the initiated time is the time to answer the call.
- **Completed**
When the call completed. Will be a “-” if the call was never connected or has not yet completed.

- Duration

Completed minus Answered in seconds for ease of reference.

The web interfaces “Call History” option calls this program on the current date’s log file and then presents the output in an HTML table. By default, internal signalling events are not recorded (dialling access codes from a phone to cause internal events, like entering or leaving an agent queue or turning message waiting indicators on or off) - if you want these, specify the `-e` flag.

6.8 db

The command line utility `db` is used to manipulate Berkeley DB files in the BTree format (the default for Slipper). The usage is “`db db_name command list`”, where the list of commands can be any combination of the below:

- **add *key value***
Set the key *key* to the value *value*.
- **delete *key***
Removes the key *key* from the database.
- **show *key***
Prints the value for the key *key*, if present in the database.
- **list**
Prints all the keys and their associated values in the database.
- **zero**
Clears the contents of the database. To avoid race conditions, this actually sets a flag that is acted on after all other commands, so its exact location in the list of commands doesn't matter. After all other commands have taken effect, all keys will be removed from the database except those that appeared in “add” or “modify” commands during this run of the utility. This makes sure there is never a window when elements staying constant are not present.
- **load *filename***
Load a list of commands from a filename. The comment character is #, and lines can be continued with a trailing backslash character. Lines without a valid command but that have a string, followed by whitespace, followed by anything, will be considered to have an implicit “add” command. For the sake of brevity, + is a synonym for “add”, not requiring whitespace between it and the key, and likewise - is a synonym for “delete”.
- **modify *key value***
“Modify” is a synonym for “add”.
- **remove *key***
“Remove” is a synonym for “delete”.

An empty command list is assumed to include the command `list`.

6.9 check_ruleset

The command line utility `check_ruleset` takes as parameters a ruleset name and a string to pass to that ruleset, and then returns the results of the ruleset lookup and the time taken to complete. It has two primary uses - to assist the administrator in sorting out where in the rulesets some unexpected behaviour is coming from, and for developers to pin down which plugin modules are causing delays.

- `-r=<file>` or `-r <file>`
Read in an alternate rules file instead of the default `slipper.rules`.
- `<variable> = <text>`
Set the variable name. Processed after the rules file is read, so it can override variables set in the rules file.
- `-n` No intermediate output.
Only show the final result, don't slow down the process by looking up millisecond time and printing out progress as rulesets are called and return. Internally works by setting the debug level to 0.
- `-d` Debug progress.
Sets the debug level to 5, which produces the intermediate output. This is the default.
- `-d<n>` or `-d=<n>` Set debug level.
Sets the debug level to n, when n is a number. Usually only used by developers wishing to trigger specific debugs in modules.
- `--`
Do not look for arguments past this point. Useful if the input string you wish to process contains an equals sign.

Some example output:

```
devel90> ./check_ruleset Canonify 0884252255@iagu.net
2003/04/24 11:04:27.250: > Canonify (0884252255@iagu.net)
2003/04/24 11:04:27.255: > AU-SA Inbound (0884252255)
2003/04/24 11:04:27.257: < AU-SA Inbound (+61884252255)
2003/04/24 11:04:27.258: > number-to-user.db (+61884252255|+61884252255)
2003/04/24 11:04:27.259: < number-to-user.db (reception)
2003/04/24 11:04:27.260: < Canonify (reception)
(Canonify, 0884252255@iagu.net) => reception (took 0.010s)
```

Note that the system calls to get the time and print out the results do actually slow it down, if you set the `-n` parameter it completes quicker.

```
devel90> ./check_ruleset -n Canonify 0884252255@iagu.net
(Canonify, 0884252255@iagu.net) => reception (took 0.006s)
```

And a similar example with setting variables:

```

devel90> ./check_ruleset -n Remote_Group=Gateway URI=awatts-mobile@iagu.net \
'Rewrite URI' andrewr@iagu.net
(Rewrite URI, andrewr@iagu.net) => URI:0884252201@iagu.net;screen=yes
(took 0.023s)
devel90> ./check_ruleset -n Remote_Group=Gateway URI=random-number@iagu.net \
'Rewrite URI' andrewr@iagu.net
(Rewrite URI, andrewr@iagu.net) => URI:0884252255@iagu.net;screen=yes
(took 0.024s)

```

6.10 Network Address Translation

When a packet is received the top-most Via header is checked. If an IP address is not listed in the header (for example, if a DNS name is there), or if an IP address is listed, but it's not the IP address the packet came from, then a “;received=ip_address:port” tag is added to the header. When a response is received, the packet is sent back to the contents of the received tag.

Although this allows basic traversal of NAT gateways, it doesn't fix up the problem of transactions initiated from the other end (BYE's, INVITE's for holding) or registrations that use the Contact header. To work around this, if Slipper is modifying the top-most via header because of difference of IP (but not DNS name) it inspects to see if the IP claimed in the Via header is the same as the IP claimed in the Contact header. If so, it modifies the Contact header to have the IP address the packet was actually received from. Note this will not help if the request came from a device to a proxy server and then out through a NAT gateway, as the IP address in the Contact header will then not be the same as that in the top-most Via header.

Additionally, to work around NAT gateways that change the port number, if a port number is specified and is the same in both the Via and Contact headers, but the port it was received from is different, it re-writes the port number as well as the IP address in the Contact header.

6.10.1 Holding NAT entries open

NAT entries for UDP usually time out of firewalls much faster than registrations occur, in the order of one to five minutes. Once the entry has timed out, inbound calls will no longer work until the next outbound packet, usually a registration or some other outbound call.

If the following configuration is present in your `slipper.rules` file, then once every `KeepAlive_Period` seconds (default: 30 seconds) Slipper will send an empty UDP packet to every device currently in the `Registrations_File` database. Sending packets frequently like this holds open the NAT entries so calls to the device succeed at times other than immediately after registration.

```

Module                Slipper::NAT
Add Tick:              Slipper::NAT::send_keep_alives

```

6.10.2 Media Proxy

Sometimes it is required to funnel all RTP traffic through Slipper. Although this is generally not a good idea - one loses in-call redundancy, create load where

none need exist, etc, sometimes security policies or non-SIP aware NAT firewalls require the media to be proxied through a known IP address.

The media proxy is usually referenced from the Codec ruleset, and takes as an argument the IP address it is supposed to claim to that target as itself. For example, if Slipper was installed on a machine with an internal IP address of 10.0.0.1, and when presented to the outside world was 203.32.153.129 through a non-SIP aware NAT gateway, this would be an appropriate configuration.

```
Module                Slipper::NAT
Ruleset Codec
(.*)                  %Check for Proxy:$1%
Ruleset Check for Proxy
(Internal!External.*) = %Slipper>NAT>media_proxy(203.32.153.129):$1%
(External.*!Internal) = %Slipper>NAT>media_proxy(10.0.0.1):$1%
```

In this case we would also recommend having a split-DNS system, and setting the `Server_Name` variable to a DNS name that resolved externally to the 203.32.153.129 address and internally to the 10.0.0.1 address.

Chapter 7

Example Config

```
#
# Settings most installations should have.
#
Add Bind: 127.0.0.1
Set Debug_Level: 3
Set Stay_In_Path: 2543
Set Log_File_Name: log/slipper.%y%m%d
Set Status_File: status-file
Set Registrations_File: registrations.db
Set Fixup_Refer_To: 1 # Will become the default.

%<etc/slipper.webconfig%
%<etc/replication.rules%

#
# Include modules to be referenced later in the configuration.
# If you are not providing PABX services, comment these out and also the
# references to them later in the configuraton.
#
Module Slipper::E164
Module Slipper::HelperApp
Module Slipper::MWIcache
Module Slipper::NAT
Module Slipper::Pickup
Module Slipper::ReferToFixup
Module Slipper::ReturnTransfer
Module Slipper::RingToneSelection
Module Slipper::Sticky
Module Slipper::UserPreferences
Add Tick: Slipper::NAT::send_nat_notify_keep_alives

Ruleset Canonify
```

```

#
# Remove quoted text
#
(.*) \" [^\"]* \" (.*) := $1$2

#
# Focus, but not on empty <>.
# Handle special case where an anonymous URI is fed back in after already
# going through the canonification process.
#
.*( <>@<> ).* = $1
.*?<([^\<>]{1,})>.* $1

#
# Remove leading "sip:", if present.
#
sip: (.*) $1

#
# Remove trailing tags...
#
(.*)? ; .* $1

#
# Remove trailing default port
#
(.*) :5060 $1

#
# If there's no @ here, it's an anonymous host and should be left alone.
# Check first if it's actually us!
#
${Aliases} = <> @ <>
([^\@]*) = <> @ $1

#
# Strip off internally designated local URL's.
#
(.*)@<>$1

#
# Strip off local domains. Keep doing it in case there are multiple.
# Also strip of trailing ? in host portions to work around problems
# with Grandstream phones doing attended transfers.
#
(.*)@${Aliases}(|\?.*) := $1

#
# Everything else without a local domain should be left alone.
#

```

```

(*@.*) = $1

#
# Restore the original URI in a call transfer.
#
Conditional If (Transfer_ID)
(${Transfer_ID}\d{10}) %Slipper>ReferToFixup>canonify:$1%
Conditional End (Transfer_ID)

#
# User may have set some type of override code, which we need to record.
# Parsing of flags is delayed in call transfers until setup of the new call leg.
#
(*) ${Method} ! $1
REFER ! (*) $1
.* ! (*) %Canonify Parse Flags:$1%
Ruleset Canonify Parse Flags
\*30(*) %Slipper>UserPreferences>set_flag(UserOverride):$1%
Ruleset Canonify

#
# OK, it's a phone number, parse according to local conventions.
# Of course, we have figure out what local *is* first. :-)
#
(*) $1 ! ${Remote_Group}
(*) ! AU-NSW-* %AU-02 Inbound:$1%
(*) ! AU-ACT-* %AU-02 Inbound:$1%
(*) ! AU-VIC-* %AU-03 Inbound:$1%
(*) ! AU-TAS-* %AU-03 Inbound:$1%
(*) ! AU-QLD-* %AU-07 Inbound:$1%
(*) ! PNG-* %PNG Inbound:$1%
(*) ! .* %AU-08 Inbound:$1% # NT, SA, and WA are default

#
# Now see if it's one of our users or speed dials.
#
(*) %number-to-user.db:$1|$1%
(*) %speed-dials.db:$1|$1%

#
# Database lookups may have added a local host that needs to be stripped.
#
(*) @ ${Aliases} $1

Ruleset Expand
#
# If there's a @ sign here, it's probably an ACK or BYE sent to something
# listed in a Contact header, and we should leave it alone. It could also

```

```

# be something external, in which case we should also leave it alone.
#
(*@.*) = $1

#
# Is the user setting some expansion preferences?
#
\*31(\d*) %Slipper>UserPreferences>set_prefs:show-voicemail$1%
\*32(\d*) %Slipper>UserPreferences>set_prefs:hide-voicemail$1%
\*33(\d*) %Slipper>UserPreferences>set_prefs:show-mobile$1%
\*34(\d*) %Slipper>UserPreferences>set_prefs:hide-mobile$1%

#
# If it's a call pickup, we pass it to the pickup module.
# The flag "G" tells the pickup module to only do pickups from within
# the group of the requestor, ie, no cross-site call pickups.
# Note the flag "1" is to tell it to wait one second before sending the
# INVITE, as Cisco 7905's don't cope with the INVITE arriving while it's
# still doing call cleanup. If you don't have any 7905's, remove the "(1)"
# and call pickups will go faster for you.
#
\*70(*) %Slipper>Pickup>pickup(1G):$1%

#
# Let MWIcache intercept Message Waiting Indicators.
#
(*) %Slipper>MWIcache>expand(mwi-status.db):$1%
\*81 %Slipper>MWIcache>set_status(mwi-status.db):yes%
\*82 %Slipper>MWIcache>set_status(mwi-status.db):no%
\*83 %Slipper>MWIcache>set_status(mwi-status.db):never%

#
# Call Queueing
#
# Module Slipper::Queueing
# Set Queue_Port: 9123
# Add Open_Queue: main
#\*91 %Slipper>Queueing>register_queue_agent:main%
#\*92 %Slipper>Queueing>deregister_queue_agent:main%
#

#
# Otherwise, we look up the Expansions database.
#
(*) %expansions.db:$1|$1%
(*)@@([~,;~+=]*)(+\d*|+=\d*|)([~,;~].*|$) := $1 %Recursive Expand:$3!$2%$4
(*) %Slipper>UserPreferences>check_override:$1%

#

```

```

# Expansions may have give us a special command.
#
register-agent-(.*) %Slipper>Queueing>register_queue_agent:$1%
deregister-agent-(.*) %Slipper>Queueing>deregister_queue_agent:$1%

#
# Make callers "Sticky", that is, try and make the client talk to the same
# person they talked to last time.
#
STICKY:[;,](.*) %Slipper>Sticky>prepend_sticky:$1% $1

#
# If it's a call transfer, the originator might want the call back
# automatically if it doesn't get picked up.
#
(.*) %Slipper>ReturnTransfer>return_transfer(delay=15):$1%

#
# And we look up registrations (may expand to multiple targets).
#
(.*) %REGISTRATION>registrations.db:$1%

Ruleset Recursive Expand
(.*)!(.*) $1 ! %expansions.db:$2|$2%
(.*)!([^,;~]*)(.*) = $2$1$3

Ruleset Route
#
# Special handling of tags in expansions.
#
(.*) ! (.* ) := $1 ; $2

#
# Handle anonymous URI's first.
#
<> @ <>ERROR:400:Bad Request - \
Anonymous URI specifies this server
<> @ (.* ) = $1

#
# Recognise local addresses.
#
(.*) @ ${Aliases} $1
(.*) @ <>$1

#

```

```

# Now to route the call - first, user to number mappings.
#
(.* ) %user-to-number.db:$1|$1%

#
# If we have an @ sign, we have sufficient information to route.
# Handle Route Groups.
#
(.* )@(route group-.* ) = %Expand Route Group:$1|$2%
(.*@.* ) = $1

#
# Access to the HelperApp framework. Some shortcuts at the top. :-)
#
(get-|)voicemail ha-voicemail
(ha-|)queue-([^\@]*) ha-queue/$2
(ha-|)conference-([^\@]*) ha-conference/$2
helperapp = internal:Slipper>HelperApp>handle
ha-([^\@]*) = internal:Slipper>HelperApp>handle(ha/$1)

#
# Include public internet-only routes and local routing rules.
#
%<etc/public.rules%
%<etc/route.rules%
(.* )@(route group-.* ) = %Expand Route Group:$1|$2%
(.*@.* ) = $1

#
# If we get here we should have a number only. Work out where to route it.
#

#
# Reject the call if it's an internal number - if we got this far,
# it must not be registered.
#
${Local_Prefix}.* ERROR:604:Does Not Exist Anywhere

#
# OK, it's not internal, so pass it out the right gateway.
#
(+\d*) = %Slipper>E164>reverse(enum.iagu.net):$1%, \
  %AU Outbound:$1%@${Default_Gateway}

#
# Whoops! It wasn't a number.
#
(.* ) ERROR:604:Does Not Exist Anywhere

```

Ruleset Expand Route Group

```
(.*)!(.*) $1 ! VIA: %REGISTRATION>registrations.db:$2%
(.*);(.*):= $1 ! VIA: $2
([~!]*)!(.*)VIA:[~!]*@([~!]*)(.*) := $1 ! $2 $1@$3 $4
[~!]*!(.*) $1
(.*)!(.*) := $1,$2
```

Ruleset Rewrite URI

```
#
# Only called on Remote-Party-ID.
# We only do this going to our external gateway.
#
(.*) $1 ! ${Remote_Group}
(.*) ! (|.*-) Gateway $1
(.*) ! .* = $1

#
# Canonify what we've got and map up our internal alphanumeric name to
# an external numeric one, otherwise fall through to a generic number.
#
(.*) $1 ! ${Auth_User}
.* ! ([~!]{1,}) $1
(.*) ! %Rewrite Canonify:$1%
(.*) %gateway-display.db:URI::$1|$1%
URI: 2(\d\d)(|;.*) URI: ${Local_Prefix} $1 $2
URI: (.*)exposed=test(.*) \
%Check Exposed:$1$2
URI: ([~;]*)(|;.*) = URI: %AU Outbound:$1% @ ${Realm_Name}; $2; screen=yes
(.*)alternate=(+\d*)(.*) \
= URI: %AU Outbound:$2% @ ${Realm_Name}; screen=yes

#
# If this isn't internal, specify the bounce number.
# Let internal callers specify if they want Caller-ID inhibited.
#
.* ${Alt_Remote_Group}
(|.*-) Gateway = URI: %AU Outbound:${Local_Prefix} ${Bounce_Suffix}% \
@ ${Realm_Name}; screen=yes; privacy=off
External = URI: %AU Outbound:${Local_Prefix} ${Bounce_Suffix}% \
@ ${Realm_Name}; screen=yes; privacy=off
.* = URI: %AU Outbound:${Local_Prefix} ${Display_Suffix}% \
@ ${Realm_Name}; screen=yes

#
# Check to see if this target is allowed to see internal numbers.
```

```

#
Ruleset Check Exposed
(.* ) $1 ! %Rewrite Canonify:${URI}%
(.* ) ! .* @ .* = $1
(.* ) ! (.* ) $1 ! %expose-clid-to-target.db:MATCH::$2|$2%
(.* ) ! MATCH: .* = URI: $1

Ruleset Rewrite Display
#
# Canonify what we've got.
#
(.* ) %Rewrite Canonify:$1%

#
# Insert Diversion header
#
(.* ) $1 ! %Insert Divert:-%

#
# Map Anonymous
#
<>@<>!(.*) = DISPLAY:$1 Unknown Caller;privacy=off
<>@<>! *=Unknown Caller!$1

#
# Replace with the Display Name, if applicable.
#
(.* )!(.*) *=%display-names.db:DISPLAY::$1|NOMATCH:$1!$2

#
# And return what is appropriate
#
DISPLAY:(.* )!(.*) @=DISPLAY:$2$1
NOMATCH:(.* )!(.*) @=DISPLAY:$2$1

Ruleset Insert Divert
.* ${Diversion}
.*sip:([^\@]*).* = $1/
.* ${Referred-By}
.*sip:ha-queue-([^\@]*).* = $1/
.* =

Ruleset Rewrite Canonify

```

```

#
# After Canonification, if there's still a domain part, see if the domain
# part is an External group or not. If it is, we leave it alone, but if not
# we assume it's a phone or gateway with a local domain part and a user part
# that's hopefully unique to our system.
#
(.* ) %Canonify:$1%
(.* )@(.* ) $1 @ $2 ! %Group:$2%
(.* ) ! External = $1 # Leave alone if External.
(.* )@.* ! .* = %Canonify:$1@<>% # Strip domain part if not.

```

Ruleset Rewrite Refer-To

```

(.*@.* ) = $1
(.* ) = %AU Outbound:$1% @ ${Server_Name}

```

Ruleset Group

```

SLIPPER_INTERNAL = Slipper
127\.\0\.\0\.\1.* = LocalHost
${Default_Gateway}(|:.* ) = Gateway
${Local_IPs}.* = Internal
(.* ) = External

```

Ruleset Register

```

#
# Anything External that's trying to Register must authenticate itself
# first, and the target it's trying to register must be the same as the
# username used in Authentication.
# This ruleset is passed registration_target ! registration_contact ! group
#
.* ! .* ! External ERROR:407:Proxy Authentication Required
.* ! .* ! External-Full = REGISTER
(.* ) $1 ! ${Auth_User}
(.* ) ! .* ! External-.* ! \1 = REGISTER
.* ! .* ! External-.* ! .* ERROR:407:Proxy Authentication Required
.* = REGISTER

```

Ruleset On Register

```

(.* ) %Slipper>MWIcache>on_register(mwi-status.db):$1%

```

Ruleset Authenticate

```

#
# Return the password for that user, and optionally a modifier
# to the group name.
#
(.* ) %auth.db:AUTH::$1|$1%
AUTH:(.* ) = $1
(.* ) ERROR:407:Proxy Authentication Required

Ruleset Authorise
#
# Auth'd people are OK.
#
[^!]*-(Full|Normal)!.* = OK

#
# If we want to receive the call, then Canonify should have turned it into
# a local name. If it's still a number, then it's not a published address,
# and is probably someone trying to make free calls out our gateway.
#
External(|-.* )!(+|\*)\d* ERROR:407:Proxy Authentication Required

#
# If there's an @ sign and it's external, in all likelihood we don't want
# to know about it. If it's an ACK or similar (sent to internal addresses),
# the context-sensitive stuff will already have allowed it.
#
External(|-.* )!.*@.* ERROR:407:Proxy Authentication Required

#
# Pretty much everything else is OK.
#
(.* ) = OK

Ruleset Codec
#
# This won't change anything but may add headers to the packet.
#
(.* ) %Slipper>RingToneSelection>selection:$1%
#
# Slipper Internal stuff generally wants an uncompressed channel.
#
.*(Slipper).* = PCMA,PCMU
#
# Remove privilege markers before group comparison
#
(.* )-(Normal|Full|Gateway)(.* ) := $1 $3
#

```

```

# Consider the "Gateway" group to be "Internal" for codec-matching purposes.
#
(.*) Gateway (.*) := $1 Internal $2
#
# Check if we need a media proxy.
#
(.*) %Check for Media Proxy:$1%
#
# These are special cases where we should use low-rate codecs even if the
# Group name is the same.
#
.*(External|Remote).* = G729,iLBC,Speex,GSM,G723,PCMA,PCMU
#
# If both endpoints are in the same group, use the high-rate codecs.
#
([^!]*!) \1 = PCMA,G729,iLBC,Speex,GSM,G723,PCMU
#
# Otherwise they must be different areas, so use the low-rate codecs.
#
.* = G729,iLBC,Speex,GSM,G723,PCMA,PCMU

```

```

Ruleset Check for Media Proxy
%<etc/media-proxy.rules%

```

```

Ruleset Ring Tone Selection
(.*) ${URI} ! $1
sip:([^@!]*)@(.* ) %ring-tones.db:ALERT-INFO::$1|$1@$2%
(ALERT-INFO:.* ) = $1
[^!]*!(.* ) $1
External .* = NULL
(.*) ! \1 = ALERT-INFO: <Bellcore-dr3>
.* = NULL

```

```

Ruleset Force Socket
Slipper ! .* = SOCKET:NonLocal
LocalHost ! LocalHost = OK
LocalHost ! .* = SOCKET:NonLocal
.* = OK

```

```

Ruleset AU-02 Inbound
(.*\D.*) = $1
2 (..) = ${Local_Prefix} $1

```

```
0011 (\d{4,}) = + $1
0011 .* ERROR:484:Address Incomplete
000 = + 61 000
0 ([234578].{8}) = +61 $1
0 ([234578].{0,7}) ERROR:484:Address Incomplete
([1].*) = +61 $1
([2-9].{8}) = +61 $1 # Special for Telstra CLID
([2-9].{7}) = +61 2 $1
([2-9].{0,6}) ERROR:484:Address Incomplete
.* ERROR:404:Not Found
```

Ruleset AU-03 Inbound

```
(.*\D.*) = $1
2 (..) = ${Local_Prefix} $1
0011 (\d{4,}) = + $1
0011 .* ERROR:484:Address Incomplete
000 = + 61 000
0 ([234578].{8}) = +61 $1
0 ([234578].{0,7}) ERROR:484:Address Incomplete
([1].*) = +61 $1
([2-9].{8}) = +61 $1 # Special for Telstra CLID
([2-9].{7}) = +61 3 $1
([2-9].{0,6}) ERROR:484:Address Incomplete
.* ERROR:404:Not Found
```

Ruleset AU-07 Inbound

```
(.*\D.*) = $1
2 (..) = ${Local_Prefix} $1
0011 (\d{4,}) = + $1
0011 .* ERROR:484:Address Incomplete
000 = + 61 000
0 ([234578].{8}) = +61 $1
0 ([234578].{0,7}) ERROR:484:Address Incomplete
([1].*) = +61 $1
([2-9].{8}) = +61 $1 # Special for Telstra CLID
([2-9].{7}) = +61 7 $1
([2-9].{0,6}) ERROR:484:Address Incomplete
.* ERROR:404:Not Found
```

Ruleset AU-08 Inbound

```
(.*\D.*) = $1
2 (..) = ${Local_Prefix} $1
0011 (\d{4,}) = + $1
0011 .* ERROR:484:Address Incomplete
```

```
000 = + 61 000
0 ([234578].{8}) = +61 $1
0 ([234578].{0,7}) ERROR:484:Address Incomplete
([1].*) = +61 $1
([2-9].{8}) = +61 $1 # Special for Telstra CLID
([2-9].{7}) = +61 8 $1
([2-9].{0,6}) ERROR:484:Address Incomplete
.* ERROR:404:Not Found
```

Ruleset PNG Inbound

```
(.*\D.*) = $1
2 (..) = ${Local_Prefix} $1
05 (\d{4,}) = + $1
05 (.* ) ERROR:484:Address Incomplete
000 = + 675 000
(.{7}) = +675 $1
(.{0,6}) ERROR:484:Address Incomplete
.* ERROR:404:Not Found
```

Ruleset Hong-Kong Inbound

```
(.*\D.*) = $1
2 (..) = ${Local_Prefix} $1
05 (\d{4,}) = + $1
05 (.* ) ERROR:484:Address Incomplete
999 = + 852 999
(.{8}) = +852 $1
(.{0,7}) ERROR:484:Address Incomplete
.* ERROR:404:Not Found
```

Ruleset Hong-Kong Outbound

```
+852 (.* ) = $1
+ (.* ) = 001 $1
```

Ruleset AU Outbound

```
+61 ([234578].*) = 0 $1
+61 (.* ) = $1
+ (.* ) = 0011 $1
```

Ruleset PNG Outbound

```
+675 (.* ) = $1
```

```
+ (.* ) = 05 $1
```

```
#  
# Comment first line of the rule and uncomment second line if you want to use  
# the store system with email notifications rather than the default email.  
# Comment first line of the rule and uncomment third line if you want to use  
# the store system without email notifications as the default.  
#
```

```
Ruleset SimpleVM User
```

```
#  
(.*) %voicemail.db:MATCH::$1|NOMATCH:$1%  
#(.* ) %voicemail.db:MATCH::$1|NOMATCH:$1:store/$1%  
#(.* ) %voicemail.db:MATCH::$1|NOMATCH::store/$1%  
#  
NOMATCH:(.*:store/)(.*)@.* %voicemail.db:MATCH::$2|$1$2%  
NOMATCH:(.*)@.* %voicemail.db:MATCH::$1|$1%  
(|NO)MATCH:(.*) $2  
(.*@.*) = $1  
:(.*)!(.*) = : $1 ! $2  
(.*):(.*)(.*) = $1 @ ${Realm_Name} : $2 ! $3  
(.*)!(.*) = $1 @ ${Realm_Name} ! $2  
:(.*) = : $1  
(.*):(.* ) = $1 @ ${Realm_Name} : $2  
(.*) = $1 @ ${Realm_Name}
```

```
Ruleset SimpleVM Get
```

```
(.*) %get-voicemail.db:$1|$1%
```

```
Ruleset Conference Options
```

```
(.*) %conference-options.db:$1%
```

```
Ruleset Queue Expand by Queue
```

```
(.*) %queue-expand-by-queue.db:$1|$1%
```

```
Ruleset Queue Expand by Agent
```

```
(.*) %queue-expand-by-agent.db:$1|$1%
```

```
Ruleset Queue Redirect to Agent
```

```
(.*) %Slipper>Support>write_to_log(Queue-to-Agent):$1%
```

```
Ruleset Null
```

```
(.*) %Slipper>Support>write_to_log(Null-Log):$1%
```

```
Conditional If (Music_On_Hold)
```

```
Module Slipper::MusicOnHold
```

```
Ruleset Music on Hold
```

```
Conditional If (Local_Music_On_Hold)
```

```
%<etc/local-music-on-hold.rules%
```

```
.*(MoH:.* ) = $1
```

```
Conditional End (Local_Music_On_Hold)
```

```
.*!.*!PCMA/8000(|/\d*) @=MoH:${Music_On_Hold}
```

```
Conditional End (Music_On_Hold)
```

```
Ruleset Output Channel
```

```
Conditional If (Transfer_ID)
```

```
(.*Gateway.*) %Slipper>ReferToFixup>output_channel(${Transfer_ID}):$1%
```

```
Conditional End (Transfer_ID)
```

```
Conditional If (Sticky_Target)
```

```
(.*) %Slipper>Sticky>output_channel:$1%
```

```
Conditional End (Sticky_Target)
```

```
Ruleset Output Filter
```

```
Conditional If (Music_On_Hold)
```

```
(.*) %Slipper>MusicOnHold>output_filter:$1%
```

```
Conditional End (Music_On_Hold)
```

Chapter 8

Third-Party Integration

8.1 Cisco IOS Routers

First one needs to set up the SIP user agent.

- `sip-ua`

This enters the SIP User Agent configuration mode.

- `max-forwards 15`

For reasons unknown, Cisco IOS defaults to Max-Forwards set to one. The maximum (as of this writing, with IOS 12.2(13)T1) you can set it to is 15. RFC 3261 suggests 70 as a default.

- `sip-server ipv4:IP_ADDRESS:5060` or
`sip-server dns:DOMAIN.NAME`

Set a SIP server to use. The first form uses only a single target, while the second will use DNS SRV records, so it can specify a cluster of Slipper servers. Checking DNS SRV operation can be done with the `show hosts` command.

- `exit`

Exit from SIP User Agent Configuration.

- `dial-peer voice 1 voip`

Enter Dial Peer configuration mode.

- `session protocol sipv2`
`session target sip-server`

Configure that dial peer to use the configuration from the SIP User Agent configuration.

Here's some example config for a Cisco 5400 to use SIP:

```
!  
sip-ua  
max-forwards 15
```

```

sip-server dns:iagu.net
!
voice service voip
sip
  bind all source-interface Loopback0
  no rel1xx
!
voice class uri 200 sip
  host (iagu.net|sip.voip.iagu.net)
!
dial-peer voice 100 pots
  description --- Data - pass to resource pooling ---
  preference 1
  application data_dialpeer
  incoming called-number 08842522[78].
!
dial-peer voice 200 voip
  description --- Inbound Calls ---
  preference 2
  application session
  progress_ind setup enable 3
  voice-class codec 1
  session protocol sipv2
  session target sip-server
  destination uri 200
  dtmf-relay rtp-nte
  ip qos dscp cs5 media
  ip qos dscp cs3 signaling
!
dial-peer voice 250 voip
  description --- Inbound Calls ---
  preference 3
  application session
  destination-pattern 08842522..
  progress_ind setup enable 3
  voice-class codec 1
  session protocol sipv2
  session target sip-server
  dtmf-relay rtp-nte
  ip qos dscp cs5 media
  ip qos dscp cs3 signaling
!
dial-peer voice 251 voip
  description --- Call Transfer - ReferToFixup Module ---
  preference 2
  application session
  destination-pattern D.....
  progress_ind setup enable 3
  voice-class codec 1
  session protocol sipv2

```

```

    session target sip-server
    dtmf-relay rtp-nte
    ip qos dscp cs5 media
    ip qos dscp cs3 signaling
    !
dial-peer voice 300 pots
    description --- Just don't ask ---
    preference 7
    application session
    incoming called-number 08842522..
    no digit-strip
    direct-inward-dial
    port 7/0:D
    !
dial-peer voice 400 pots
    description --- Outbound calls ---
    preference 10
    application session
    destination-pattern ...
    no digit-strip
    port 7/0:D
    !
voice class codec 1
    codec preference 1 g711alaw bytes 240
    codec preference 2 g729r8 bytes 40
    codec preference 3 g729br8 bytes 40
    codec preference 4 g723r63 bytes 96
    codec preference 5 g726r16 bytes 80
    codec preference 8 gsmfr
    codec preference 14 g711ulaw bytes 240
    !
interface Serial7/0:15
    ip unnumbered FastEthernet0/0
    encapsulation ppp
    isdn switch-type primary-net5
    isdn incoming-voice modem
    isdn send-alerting
    peer default ip address pool default
    ppp authentication chap pap callin
    !
voice-port 7/0:D
    echo-cancel coverage 64
    cptone AU
    !

```

And if you want RADIUS accounting, add this — the `h323` keyword needs to be left as is, even through it's SIP, IOS makes some assumptions about all voice calls being via H.323. Note this example is for 12.2(11)T.

```
!
```

```

aaa accounting connection h323 start-stop group RADIUS
aaa nas port voip
radius-server vsa send accounting
!
gw-accounting aaa
method voip
acct-template callhistory-detail
!

```

The above configuration will produce two RADIUS stop records, one for the VoIP call leg and one for the PSTN call leg, an example of each (for the same call) listed below. Note the VoIP leg can be matched to Slipper records by the Call-ID, and the VoIP leg can be matched to the PSTN leg by the H323-Conf-ID. Note the Session-ID's do not match as they are reporting separate events.

```

Wed Mar 24 09:59:12 2004 203.32.153.130:1646 (4/62)
NAS-IP-Address = 203.32.153.130
Service-Type = Login
Called-Station-ID = "0884254970"
Calling-Station-ID = "0884252255"
Acct-Status-Type = Stop
Acct-Delay-Time = 0
Acct-Input-Octets = 305961
Acct-Output-Octets = 248661
Acct-Session-ID = "00000AC9"
Acct-Session-Time = 40
Acct-Input-Packets = 1953
Acct-Output-Packets = 1057
Cisco-AVpair = "call-id=0007ebf0-f3e80050-1e274a78-31c4f222@203.32.153.138"
Cisco-AVpair = "iphop=count:2"
Cisco-AVpair = "iphop=hop1:sip.voip.iagu.net"
Cisco-AVpair = "iphop=hop2:203.32.153.138"
Cisco-AVpair = "h323-incoming-conf-id=9AB554BB 7C5811D8 8614F89B 26523E77"
Cisco-AVpair = "subscriber=Unknown"
Cisco-AVpair = "session-protocol=sipv2"
Cisco-AVpair = "gw-rxd-cdn=ton:0,npi:0,#:0884254970"
Cisco-AVpair = "release-source=2"
Cisco-AVpair = "remote-media-address=203.32.153.138"
Cisco-AVpair = "gw-rxd-cgn=ton:0,npi:0,pi:0,si:1,#:0884252255"
Cisco-AVpair = "charged-units=0"
Cisco-AVpair = "disconnect-text=normal call clearing (16)"
Cisco-AVpair = "peer-address=0884252255"
Cisco-AVpair = "info-type=speech"
Cisco-AVpair = "peer-id=200"
Cisco-AVpair = "peer-if-index=95"
Cisco-AVpair = "logical-if-index=0"
Cisco-AVpair = "codec-bytes=240"
Cisco-AVpair = "coder-type-rate=g711alaw"
Cisco-AVpair = "ontime-rv-playout=20000"
Cisco-AVpair = "remote-udp-port=5060"
Cisco-AVpair = "remote-media-udp-port=23718"

```

Cisco-AVpair = "vad-enable=enable"
Cisco-AVpair = "receive-delay=65 ms"
Cisco-AVpair = "round-trip-delay=0 ms"
Cisco-AVpair = "hiwater-playout-delay=120 ms"
Cisco-AVpair = "lowater-playout-delay=60 ms"
Cisco-AVpair = "gapfill-with-interpolation=12025 ms"
Cisco-AVpair = "gapfill-with-prediction=0 ms"
Cisco-AVpair = "gapfill-with-redundancy=0 ms"
Cisco-AVpair = "gapfill-with-silence=0 ms"
Cisco-AVpair = "early-packets=2"
Cisco-AVpair = "late-packets=0"
Cisco-AVpair = "lost-packets=0"
Cisco-H323-Remote-Address = "h323-remote-address=203.32.153.129"
Cisco-H323-Conf-ID = "h323-conf-id=9AB554BB 7C5811D8 8614F89B 26523E77"
Cisco-H323-Setup-Time = "h323-setup-time=09:58:20.201 ACDT Wed Mar 24 2004"
Cisco-H323-Call-Origin = "h323-call-origin=answer"
Cisco-H323-Call-Type = "h323-call-type=VoIP"
Cisco-H323-Connect-Time = "h323-connect-time=09:58:32.237 ACDT Wed Mar 24 2004"
Cisco-H323-Disconnect-Time = "h323-disconnect-time=09:59:12.465 ACDT Wed Mar 24 2004"
Cisco-H323-Disconnect-Cause = "h323-disconnect-cause=10"
Cisco-H323-Voice-Quality = "h323-voice-quality=16"
Cisco-H323-GW-ID = "h323-gw-id=gw.voip.off.iagu.net"

Wed Mar 24 09:59:12 2004 203.32.153.130:1646 (4/63)

User-Name = "0884252255"
NAS-IP-Address = 203.32.153.130
NAS-Port = 10
Service-Type = Login
Called-Station-ID = "0884254970"
Calling-Station-ID = "0884252255"
Acct-Status-Type = Stop
Acct-Delay-Time = 0
Acct-Input-Octets = 248661
Acct-Output-Octets = 305961
Acct-Session-ID = "00000ACB"
Acct-Session-Time = 40
Acct-Input-Packets = 1057
Acct-Output-Packets = 1953
NAS-Port-Type = Async
Cisco-AVpair = "h323-incoming-conf-id=9AB554BB 7C5811D8 8614F89B 26523E77"
Cisco-AVpair = "subscriber=Unknown"
Cisco-AVpair = "out-intrfc-desc=— Optus %2b618842522.. Service —"
Cisco-AVpair = "gw-rxd-cdn=ton:0,npi:0,#:0884254970"
Cisco-AVpair = "h323-ivr-out=Tariff:Unknown"
Cisco-AVpair = "release-source=2"
Cisco-AVpair = "alert-timepoint=09:58:21.633 ACDT Wed Mar 24 2004"
Cisco-AVpair = "gw-rxd-cgn=ton:0,npi:0,pi:0,si:1,#:0884252255"
Cisco-AVpair = "gw-final-xlated-cdn=ton:0,npi:0,#:0884254970"
Cisco-AVpair = "gw-final-xlated-cgn=ton:0,npi:0,pi:0,si:1,#:0884252255"
Cisco-AVpair = "charged-units=0"

```

Cisco-AVpair = "disconnect-text=normal call clearing (16)"
Cisco-AVpair = "peer-address=0884254970"
Cisco-AVpair = "info-type=speech"
Cisco-AVpair = "peer-id=400"
Cisco-AVpair = "peer-if-index=96"
Cisco-AVpair = "logical-if-index=79"
Cisco-AVpair = "acom-level=90"
Cisco-AVpair = "coder-type-rate=g711alaw"
Cisco-AVpair = "noise-level=4294967240"
Cisco-AVpair = "voice-tx-duration=14000 ms"
Cisco-AVpair = "tx-duration=49770 ms"
Cisco-NAS-Port = "Serial3/0:10"
Cisco-H323-Conf-ID = "h323-conf-id=9AB554BB 7C5811D8 8614F89B 26523E77"
Cisco-H323-Setup-Time = "h323-setup-time=09:58:20.581 ACDT Wed Mar 24 2004"
Cisco-H323-Call-Origin = "h323-call-origin=originate"
Cisco-H323-Call-Type = "h323-call-type=Telephony"
Cisco-H323-Connect-Time = "h323-connect-time=09:58:32.233 ACDT Wed Mar 24 2004"
Cisco-H323-Disconnect-Time = "h323-disconnect-time=09:59:12.473 ACDT Wed Mar 24 2004"
Cisco-H323-Disconnect-Cause = "h323-disconnect-cause=10"
Cisco-H323-Voice-Quality = "h323-voice-quality=0"
Cisco-H323-GW-ID = "h323-gw-id=gw.voip.off.iagu.net"

```

To get all the above fields to come out, you will need to know what Vendor-Specific Attributes Cisco use. Radiance (another product from Iagu) has a reasonably standard dictionary format and uses the below, which may be able to be directly pasted into a similar format dictionary.

VENDOR	Cisco	9		
VENDORATTR	Cisco	AVpair	1	string
VENDORATTR	Cisco	NAS-Port	2	string
VENDORATTR	Cisco	Fax-Account-ID-Origin	3	string
VENDORATTR	Cisco	Fax-Msg-ID	4	string
VENDORATTR	Cisco	Fax-Pages	5	string
VENDORATTR	Cisco	Fax-Coverpage-Flag	6	string
VENDORATTR	Cisco	Fax-Modem-Time	7	string
VENDORATTR	Cisco	Fax-Connect-Speed	8	string
VENDORATTR	Cisco	Fax-Recipient-Count	9	string
VENDORATTR	Cisco	Fax-Process-Abort-Flag	10	string
VENDORATTR	Cisco	Fax-DSN-Address	11	string
VENDORATTR	Cisco	Fax-DSN-Flag	12	string
VENDORATTR	Cisco	Fax-MDN-Address	13	string
VENDORATTR	Cisco	Fax-MDN-Flag	14	string
VENDORATTR	Cisco	Fax-Auth-Status	15	string
VENDORATTR	Cisco	Email-Server-Address	16	string
VENDORATTR	Cisco	Email-Server-Ack-Flag	17	string
VENDORATTR	Cisco	Gateway-ID	18	string
VENDORATTR	Cisco	Call-Type	19	string
VENDORATTR	Cisco	Port-Used	20	string
VENDORATTR	Cisco	Abort-Cause	21	string

VENDORATTR	Cisco	H323-Remote-Address	23	string
VENDORATTR	Cisco	H323-Conf-ID	24	string
VENDORATTR	Cisco	H323-Setup-Time	25	string
VENDORATTR	Cisco	H323-Call-Origin	26	string
VENDORATTR	Cisco	H323-Call-Type	27	string
VENDORATTR	Cisco	H323-Connect-Time	28	string
VENDORATTR	Cisco	H323-Disconnect-Time	29	string
VENDORATTR	Cisco	H323-Disconnect-Cause	30	string
VENDORATTR	Cisco	H323-Voice-Quality	31	string
VENDORATTR	Cisco	H323-GW-ID	33	string
VENDORATTR	Cisco	H323-Credit-Amount	101	string
VENDORATTR	Cisco	H323-Credit-Time	102	string
VENDORATTR	Cisco	H323-Return-Code	103	string
VENDORATTR	Cisco	H323-Prompt-ID	104	string
VENDORATTR	Cisco	H323-Time-and-Day	105	string
VENDORATTR	Cisco	H323-Redirect-Number	106	string
VENDORATTR	Cisco	H323-Preferred-Lang	107	string
VENDORATTR	Cisco	H323-Redirect-IP-Address	108	string
VENDORATTR	Cisco	H323-Billing-Model	109	string
VENDORATTR	Cisco	H323-Currency	110	string
VENDORATTR	Cisco	Account-Info	250	string
VENDORATTR	Cisco	Service-Info	251	string
VENDORATTR	Cisco	Command-Code	252	string
VENDORATTR	Cisco	Control-Info	253	string

8.1.1 Known Problems

Cisco IOS support for SIP is pretty good from 12.3 upwards - we currently recommend 12.3T to most customers. If you have to use 12.2T for memory or other reasons, keep these issues in mind. Note that many overlap so you have to choose the set or problems you are comfortable with:

- Cisco IOS routers do not handle lower-case letters in URI's — their dial-peers can only specify upper-case letters, and will not match lower-case letters. In CSCdz12488, Cisco specify that it was an accident that upper-case letters are supported, and all support for letters outside A-D was removed in 12.2(11)T2.
- There are also some bugs in matching upper-case letters that should be stripped before presentation to a POTS service (eg, the classical trick in toll bypass of only sending things out with a leading "A" that is stripped off, so it has to go to Slipper to get the chance to re-route it somewhere else). CSCdy88670, fixed in 12.2(12.13)T
- There is a bug in sending VSA accounting information via RADIUS (CSCdz12234) that will cause the router to crash. Fixed in 12.2(11)T2.
- Introduced in 12.2(11)T1, AS-class routers don't produce ring-tone for inbound calls from the ISDN network when they have told the ISDN network they will generate it. Fixed in 12.2(13)T1.

- There are a couple of memory leak issues in 12.2T releases prior to 12.2(13)T1, the 12.2XB stream appears OK.

8.2 Cisco SIP 79x0 IP Phones

On boot, the Cisco SIP phones first attempt to establish if VLAN's are in use, by using the proprietary Cisco Discovery Protocol (CDP). If there is no CDP response, or the CDP response does not include a voice VLAN, then the phone assumes there is no VLAN and sends non-802.1q packets. If one is using a Cisco switch (powered or unpowered) and the setup requires normal data traffic in VLAN 1 and voice traffic in VLAN 2, the following is a configuration you might use.

```
interface FastEthernet0/1
  switchport trunk encapsulation dot1q
  switchport trunk allowed vlan 1,2,1002-1005
  switchport mode trunk
  switchport voice vlan 2
```

After this they use DHCP to obtain an IP address, and then attempt to get their configuration from a TFTP server specified in the DHCP lease. If the DHCP lease does not include a TFTP server, they use the last known TFTP server from NVRAM, which defaults to broadcast. In the examples below, the Slipper server has an IP of 203.32.153.129 and the router has an IP of 203.32.153.130. An example configuration for an ISC-derived DHCP server is:

```
subnet 203.32.153.128 netmask 255.255.255.240 {
  range 203.32.153.131 203.32.153.142;
  option routers 203.32.153.130;
  option domain-name "voip.iagu.net";
  option domain-name-servers 203.32.153.129, ns.iagu.net;
  option tftp-server-name "203.32.153.129";
  authoritative;
  ddns-updates off;
}
```

For a Cisco IOS-base DHCP server, the equivalent configuration is:

```
ip dhcp pool Voice
  network 203.32.153.128 255.255.255.240
  default-router 203.32.153.130
  domain-name voip.iagu.net
  dns-server 203.32.153.129 203.32.153.69
  option 66 ascii "203.32.153.129"
  option 42 ip 203.32.153.129
  lease 7 1
ip dhcp excluded-address 203.32.153.128 203.32.153.130
```

The most common mistake is thinking `tftp-server-name` is an IP address when it is a string. You can set it to be a domain name with more than one A record (IP address) associated with it, in which case it should try one then the other, useful for failover or load balancing.

The first file it looks for is `OS79XX.TXT` to inform it which code version it should be using. The contents of this for the 4.3 SIP image is `POS3-04-3-00`.

If your phone has a SIP or Skinny (SCCP) image with 2.2 or older, you may need to go via the 2.3 image with POS30203, the first image to support hyphens in the filename, otherwise the automatic firmware upgrade won't work.

Download the `SIPDefaultGeneric.cnf` file from the Cisco web site, and place a copy in your TFTP directory with the name `SIPDefault.cnf`. The lines you need to modify are summarised below:

```
image_version: POS3-04-3-00
proxy1_address: "DNS Name or IP address of your SIP server"
proxy_register: 1
preferred_codec: none
```

Other things you may want to change include:

```
timer_register_expires: 3600
dial_template: dialplan
sntp_server: "ntp.iagu.net"
sntp_mode: unicast
time_zone: CAST
dst_offset: 1
dst_start_month: 10
dst_start_day: ""
dst_start_day_of_week: 1
dst_start_week_of_month: 8
dst_start_time: 02
dst_stop_month: 3
dst_stop_day: ""
dst_stop_day_of_week: 1
dst_stop_week_of_month: 8
dst_stop_time: 2
dst_auto_adjust: 1
time_format_24hr: 1
proxy_backup: "DNS Name or IP address of your SIP server"
proxy_emergency: "DNS Name or IP address of your media gateway"
telnet_level: 2
services_url: "http://YOUR_WEBSERVER_HERE/services.xml"
directory_url: "http://YOUR_WEBSERVER_HERE/directories.xml"
logo_url: "http://www.iagu.net/slipper.bmp"
date_format: "D/M/Y"
```

If you have more than one line you wish to use, you'll need to add `proxy2_address`, etc.

The simplest dialplan in `dialplan.xml` in the same directory would be:

```
<DIALTEMPLATE>
  <TEMPLATE MATCH="000" Timeout="0" User="Phone" Route="Emergency"/>
  <TEMPLATE MATCH="*#" Timeout="0" User="Phone"/>
  <TEMPLATE MATCH="*" Timeout="15"/>
</DIALTEMPLATE>
```

That dialplan requires one to press the hash key (#) or the softkey "Dial" to place a call so the phone knows when the number has been completed.

After this, you need to create the phone-specific configurations. This is a file name with the ethernet hardware address embedded in it, such as SIP0007EFBEOA8C.cnf. An example is below:

```
line1_name: andrewr                ; On Screen
line1_authname: "andrewr"          ; Registration Username
line1_password: "ChangeMe"        ; Registration Password
line1_displayname: "andrewr@iagu.net" ; SIP Messaging Display Name

line2_name: tech                   ; On Screen
line2_authname: "andrewr"          ; Registration Username
line2_password: "ChangeMe"        ; Registration Password
line2_displayname: "tech@iagu.net" ; SIP Messaging Display Name

phone_label: "+61884252201 "       ; Has no effect on SIP messaging
phone_prompt: "AndrewR's Phone"    ; Password prompt, max 15 chars
phone_password: "ChangeMe"         ; Max 31 chars
```

If you have problems, try viewing the event log: press the “Settings” button on the phone, then “Status” (currently 5), then “Status Messages” (currently 1). If the phone thinks everything is OK it will respond with “I100 No errors”.

8.2.1 Known Problems

As of 4.1 of the phone image, there are a few bugs with the Record-Route features interaction with some other features, notably authentication. The phones also usually implement strict routing when loose routing is requested. If you require MD5 Digest authentication, set Stay_In_Path to 0 to turn Record-Route off, and the phone bugs will not be triggered. Note this means Slipper will not see call termination events, but you should probably be using RADIUS or similar for accounting anyway. These are fixed in 4.2.

Prior to 4.1.4 the SIP phones do not support daylight savings starting in a calendar month later in the year than when it starts — that is, they support the northern hemisphere but not the southern hemisphere. A temporary workaround is to specify that daylight savings ends just before the end of the calendar year, and then at the beginning of the year change the specification to start at the beginning of the year and finish at the actual finishing time.

In some cases, the SIPDefaultGeneric.cnf file supplied by Cisco will fit into the phone’s very small buffer, but once you have added the minimum to make it work, it won’t fit any more — it needs about an extra 200 bytes in 4.1 to not fit. The fix is to remove comments, blank lines, etc. until it works. Iagu uses a small script to strip a config with full comments down to one without comments for the phone to load, taking 5724 bytes down to 1676 which fits with lots of room to spare. This has been extended in 4.2 — the limit is now somewhere around the 8k mark.

In XML form submission, the phones do not currently escape spaces correctly in HTTP submission, so many web servers will reject the form or misunderstand it. Cisco are currently investigating in CSCea09043.

8.3 Cisco Analogue Telephone Adapters

The ATA will use DHCP to obtain an IP address and once it has an IP address it is possible to configure the device using a web browser. If you can not determine the IP address via other means you can use the IVR interface to query the device directly with a handset plugged into the first voice port press the button on the top of the ATA and when prompted enter 21# on the handset. The ATA will then read out the IP address assigned to it. Once you have the IP address (such as 203.32.153.135), use a web browser to request the configuration page <http://203.32.153.135/dev>.

Check that it is using the SIP (version 3) or H.323/SIP (version 2 peer to peer) image, not the SCCP/MGCP (client / server) image. If the latter image is being used (often the image that ships with the unit) the version number will be trailed by “ms”, while a H.323/SIP image has no trailing letters after the version number.

Parameters to change for use with Slipper:

- **UID0 & UID1**

These are the registration tags that the ATA will use for each line when registering with Slipper. These may be the telephone numbers of each line.

- **Proxy**

This is the IP address or DNS name of the server running Slipper. You can also use a name pointing to a DNS SRV record. Only in version 3, was called “GkOrProxy” in version 2 when there was a combined SIP/H.323 image.

- **GkOrProxy**

This is the IP address or DNS name of the server running Slipper. You can also use a name pointing to a DNS SRV record. Only in version 2, renamed “Proxy” in version 3.

- **UseLoginID**

Normally set to “0”.

- **LoginID0 & LoginID1**

If useLoginID is set to 1, then if prompted for authentication will use LoginID0 or LoginID1 (for lines 1 and 2 respectively) as the username, rather than the default of UID0 and UID1.

- **UseSIP**

Should be set to “1” to enable SIP. Only required in version 2 images.

- **SIPRegOn**

Causes Registration to occur if set to 1, which it should be set to in most operational environments.

- **OutBoundProxy**

Should be set to the IP address or DNS name of the server running slipper. It would appear that DNS SRV records are not usable in this field.

- Timezone, NTPIP and AltNTPIP

These can be set if desired. Note the ATA only supports integral hour timezones.

If you are connecting a fax machine to the ATA rather than a telephone handset then you may wish to alter the AudioMode field. This field is associated with both voice ports, the lower 16 bits correspond to the first port and the upper 16 bits correspond to the second port. A fax should have value "0012" while a handset should be set at "0015" and thus if you have a handset on the first port and a fax machine on the second port then the value would be "0x00120015".

All other fields can be left at their default value. We have yet to determine suitable values for the Tone fields to mimic non-North American country PSTN tones.

8.4 Cisco 7905 and 7912 Phones

These phones are based off the chipset and code used in the ATA's and are very similar in configuration, except for having one line instead of two. The easiest way to configure these is to move the file `ld-template.txt` to `/usr/local/tftpboot` or your local equivalent, modify the `slipper-admin.cgi` to reflect the best IP address for the phones to contact SLipper and TFTP on, and configure the phones through the web interface.

8.5 Ahead Software's SIPPS

Ahead Software¹ are running betas of a product called SIPPS, a full-featured softphone running on Win32, Linux, and FreeBSD. Iagu are actively working with Ahead Software to make sure that new features of either product will interoperate with the other.

Contact Iagu or Ahead Software if you wish to be involved in beta trials of their software interoperating with the Slipper platform.

¹Ahead Software are best known for their Nero CD burning software.

8.6 Microsoft Windows XP Messenger

To set it up, from the “Tools” menu select “Options”, when the window appears click on the “Accounts” tab, change the “Sign in with this account first” from “.NET” to “Communications Service”, un-check the “My contacts include users of .NET messenger service”, then under the “Communications Service Account Options” put in your SIP address (eg, `xp@iagu.net`) as the sign-in name, then click the “Advanced” button.

Change “Automatic Configuration” to “Configure Settings”, put in the DNS name or IP address of the Slipper server, and select “UDP” as the connection protocol. Click the OK button, and the next one back. Click the “Click here to sign in” hotlink under your SIP address. If Slipper is set up to request authentication, it will then ask for a username and password for authentication.

To place a call, click on the “Start a Voice Conversation”. Click the “Other” tab, and type in a SIP address, such as `reception@iagu.net`.

Instant Messaging through Slipper works fine.

8.6.1 Known Problems

Calls from Windows XP Messenger work, although calls to Windows XP Messenger don’t work if `Stay_In_Path` is set. This is not a fault with Slipper, it’s XP Messenger not supporting Record-Route correctly. You can work around this by setting `Stay_In_Path` to 2543 to tell Slipper to use the now-depreciated RFC 2543 method of Record-Route which Windows XP Messenger will handle.

Windows XP Messenger has no understanding of NAT gateways, so if there is NAT involved you must have a copy of Slipper running on the same side of the NAT as Windows XP Messenger.

8.7 X-Ten range of Soft Phones

Start up X-Lite or X-Pro, and click on the Menu button (to the right of the “Clear” button) if the menu does not automatically open. Click on “System Settings” and then “SIP Proxy”. Create or modify the default proxy entry. The setting inside it should be:

- **Enabled:** Yes
- **Display Name:** *Your Name*
- **User Name:** *User portion of target to register as*
- **Authorization User:** *Username to use when challenged for authentication*
Usually the same as the User Name.
- **Password:** *Password to use when challenged for authentication*
- **Domain/Realm:** *The default domain name*
This should match the Realm_Name used by Slipper for authentication and be the host portion of the target to register as.
- **SIP Proxy:** *DNS name or IP address of proxy*
- **Outbound Proxy:** *DNS name or IP address of proxy*
Optional, usually the same as above.
- **Proxy Mode:** Normal

All other options can be left alone.

8.8 Mitel 5055 SIP Phones

These have a pretty good web interface and the step-by-step setup guide is actually intuitive to follow. A few gotcha’s, though - you must go to the “Media Configuration” page to allow it to negotiate the full range of codec’s, by default it only allows G.711 u-law. Other items of note:

- With DHCP, it treats a netmask smaller than a /24 as a /24 (old-style Class C). This generally means you can get to everything on the internet except other things in the same /24, which may preclude the device from communicating with your local Slipper or some other critical service.
- It doesn’t use the NTP or TFTP servers supplied by DHCP.

8.9 SIPphone

The configuration fragments below are currently required to register with SIPphone. We understand the SIPphone requirement for the HideAddress module may be removed in the future. Replace NUMBER and PASSWORD with your SIPphone login information, and TARGET with the Canonified target you wish incoming calls to go to.

```
Module                Slipper::HideAddress
Module                Slipper::OutboundRegister
Add OutboundRegister: NUMBER@proxy01.sipphone.com TARGET
ModuleConfig          Slipper::OutboundRegister::config
```

```
Ruleset Group
198.65.166.*          = SIPphone
```

```
Ruleset Input Filter
(SIPphone)           %Slipper>HideAddress>input_filter\  
                    (NUMBER@proxy01.sipphone.com):$1%
```

```
Ruleset Output Filter
(SIPphone)           %Slipper>HideAddress>output_filter\  
                    (NUMBER@proxy01.sipphone.com):$1%
```

```
Ruleset Auth Digest
.* ! SIPphone ! (.*) = $1 ! NUMBER ! PASSWORD
```

And in route.rules then add the places you wish to reach through SIPphone:

```
+(1\d{10})           = $1 @ proxy01.sipphone.com
```

Appendix A

Release Notes

- 1.x — Initial beta tests.
- 2.0 — New proxy model, initial limited deployment release.
- 2.1 — New ruleset model.
- 2.2 — Following features moved from internal code to rulesets:
 - Display — Modify display names.
 - Group — determine group a particular client belongs to.
 - Authorise — Check if this message is allowed (Source Group, URI).
 - Codec — reordering of codec preferences (Source & Destination Group).
- 2.3 Stateful rewrite started.
 - Initial forking support.
 - Performance improvements.
 - Codec deletion added to codec reordering code.
- 3.0 Major rewrite, first commercial release.
 - Packet handling rewrite for improved forking & retransmission support.
 - Changed module hierarchy.
 - Ruleset support for $\${x}$.
 - Re-read configuration on SIGHUP.
 - Internal status dump on SIGUSR1.
 - Record-Route to include proxy in future traffic for a call.
 - “Rewrite URI” ruleset added.
 - “Display” ruleset renamed “Rewrite Display”.
 - “Authenticate” ruleset added, and MD5 Digest authentication process.

- 3.1a1
 - New versioning scheme introduced.
 - Started adding support for non-blocking DNS SRV.
 - “Register” ruleset added.
- 3.1a2
 - Allow_Unauthenticated_ACK added to work around proxy authentication problems in Microsoft’s Windows XP Messenger and Cisco’s IP phones.
- 3.1a3
 - Added “,” failover feature to the Route ruleset.
- 3.1a4
 - Optimisations to variable expansion in rulesets — caching regexp fragments for subsequent runs.
 - Added optional RFC 2543 Record-Route support for older clients as well as the existing RFC 3261.
 - “Auth_Realm” variable added for better authentication support in clusters. Some authentication optimisation done.
- 3.1a5
 - IP addresses after the first returned by a system “gethostbyname” call are now set up as backup routes.
 - Non-blocking DNS SRV support now rolled into main code stream.
 - Management CGI added to distribution.
 - `slipper.sh` startup script added to distribution.
 - `slipper.gif` and `slipper.bmp` added to distribution.
- 3.1a6
 - Significant bug fix — modify `branch=` parameter to handle multiple DNS SRV records pointing at different names that then resolve to the same IP address, such as for `sipcenter.com`.
- 3.1b1

Core code freeze, work now on documentation, utilities, and adding stable code from experimental branches.

 - Bug fix — infinite loop when a certain type of invalid packet received.
 - “Auth_Realm” renamed “Realm_Name”, and the latter is now used in some places where “Server_Name” was previously used.

- Added Rulesets “Rewrite Contact” and “Rewrite Refer-To” to change headers that may need to be represented differently to different targets, primarily to work around call transfer issues with Cisco IOS images that do not handle non-numeric characters in URI’s.
 - Bind targets can now be specified in the Rules file as a list as well as in the config file. The long-term target is to merge the functionality of the rules and config files.
- 3.1b2
 - The keyword `REGISTER` added to special handling in the Register ruleset.
 - `index.cgi` renamed `slipper-admin.cgi` and functionality added to manage Cisco 79x0 class phones.
 - The `Rewrite_ ... _Header` family of tuning variables added.
 - 3.1b3
 - Handling of more special cases in the “Remote-Party-ID” code as a result of field experience, and adding tag override abilities.
 - Example config changed to include the most common configurations in the field.
 - Fixed a bug causing the `Alt.Remote.Group` variable to not be set correctly.
 - 3.1b4
 - Fixed a bug in file inclusion.
 - Extended functionality of `slipper-admin.cgi`
 - Detection and automatic fix-up in some two-stage header modification code (eg, “Rewrite Contact”) of packets that have been duplicated from a packet in which both stages have already been completed and placed into an expansion where only one has been completed. Mainly for plugin modules such as Pickup.
 - Added `Max.Header.Length` variable.
 - Testing of null’s in headers to protect badly-implemented clients.
 - 3.1r1
 - No code changes from 3.1b4.
 - More documentation added.
 - 3.1r2
 - Documentation corrections and additions.
 - Bug fixed in DNS code dealing with A record lookups for hosts that have already been seen in the results of DNS SRV lookups.

- Minor cleanup issue — Display rewriting where `{}'`s haven't been supplied or a Display name without surrounding `"` has been supplied now works more like what one would expect.
 - Workaround of DB caching issues on some platforms.
 - Web interface CGI now works correctly when used as the index document for a directory by Apache.
 - Web interface now supports modification of routes.
 - Time code now detects and works around platforms that have a `gettimeofday` system call that return 0's instead of real values.
 - Ignore blank lines or lines composed only of nulls that `ModuleConfig` returns.
 - Bug fixed in digest challenges where a recently expired nonce is sometimes used as a challenge.
 - Added `Call_Expiry` and `Max_Delay_Factor` variables — previously hardcoded values.
 - Bug fix in handling of CGI parameters in Refer-To headers when part of the URI needs to be remapped — a problem for Cisco phone initiated attended transfers via Cisco routers that require use of the transfer-rewrite facility.
 - Fixed bug with multiple backups (`~`) in an expansion all pointing to registration entries rather than fixed entries.
- 3.1r3
 - Bug fix for backup routes specified using `~` pointing at the same IP address.
 - 3.1r4
 - Bug fix for multiple targets in a Route ruleset being pushed back into the original expansion.
 - 3.2a1
 - `Status_File` functionality added.
 - `Call_Completed_Expiry` functionality added.
 - 3.2a2
 - When a 200 to an INVITE is received after a CANCEL has been sent, a one-off ACK is now generated and a BYE that will be retransmitted. If the one-off ACK is lost, another round of this code is triggered when the retransmitted 200 is received.
 - ACK's not required by the standard no longer generated — Slipper was a bit over-chatty under circumstances, related to the previous handling of a 200/CANCEL packets passing over the network.
 - `Debug_Level 5` added

- `check_ruleset` now sets `Debug_Level` to 5 unless overridden from the command line, regardless of what `slipper.rules` specifies.
 - Added limited NAT fixup in inbound-packet parsing.
 - Included bug fix from 3.1r3.
- 3.2a3
 - Added rules to the example configuration showing how different dialplans from different locations work.
 - Registration rule added `REGISTER:new_target` feature.
 - Fixed issue with delayed backup routes not having their delay decreased if the parent route had been due to errors of those routes earlier in the expansion.
 - Some minor speed enhancements.
- 3.2a4
 - Diversion header added for better Voicemail support.
- 3.2a5
 - Bugfix from 3.1r4 incorporated.
 - `ERROR:IGNORE` added.
 - `Log_File_Name` added.
 - Some minor bug fixes to new features in core code.
- 3.2a6
 - Input and Output Filters added.
 - Bugfix in `Slipper::RuleSet`'s `REGISTRATION` function that would strip time modifiers of an entry not in the registration database.
 - Bugfix in `slipper-admin.cgi` adding a new 79x0 phone.
 - `call-accounting` program added to distribution.
- 3.2a7
 - Now handles loose source routes with `;lr=yes` or `;lr=on` - although this is not in the standard, some clients can't parse tags without an `=` present.
 - Bugfix where headers without a space after the initial colon and a colon elsewhere in the header (eg, port designators in a Via header) would be broken incorrectly.
 - Bugfix - now correctly handles codec negotiation with SDP bodies specifying an IP address between the `m=` media declaration line and the `a=` codec specifier lines.
 - Bugfix - Better handling of mixed loose source routing and strict source routing.

- 3.2a8
 - Slipper::MusicOnHold and rtp_send added to distribution.
 - Bugfix - Registrations handling for Expand Ruleset was not handling multiple backup route indicators when the registrations were present but expired.
- 3.2a9
 - Bugfix in generation of BYE messages when a CANCEL and 200 pass in transit.
 - Bugfix in Music on Hold not correctly shutting down RTP when a BYE is sent in the opposite direction that in which the hold was initiated.
 - Responses to registrations now return the expiry time for all currently registered contacts.
- 3.2a10
 - Diversion header creation changed so it is less likely to flag an a divert as unconditional for minor gateway-related changes, such as changing +6188252255 to 0884252255.
 - Now uses Time::HiRes for debug messages as well as `gettimeofday` and `time`, attempts to auto-detect best option.
 - `call-accounting` now inserts authentication username as well.
 - `call-accounting` has a bugfix. In a forked call where the 200 and the CANCEL swap in transit, Slipper generates a BYE to terminate the false leg. `call-accounting` no longer treats this BYE as the end of the session.
- 3.2a11
 - “Tick” functionality added.
 - Change so that when an empty expansion is returned, a 480 (Temporarily Unavailable) rather than a 404 (Not Found) is returned. This reduces the chance of a getting a “Check the number and try again” message generate by the carrier if a known user currently has no valid registrations.
 - In most cases error messages are no longer generated in response to ACK’s.
 - Bugfix - the BYE generation code introduced in 3.2a9 no longer produces BYE’s on a normal cancel when the 487 for the INVITE and the 200 for the CANCEL arrive out of expected order, but still in a valid order.
 - Bugfix - now correctly handles “?” modifiers to URI’s inside Refer-To and Contact headers bounded by a <> pair.
 - Time::HiRes scoping bugfix.

- 3.2a12
 - Transparent interception support removed from response handler - this should already have happened in the request, and so can really only happen if someone is up to no good or the configuration is incorrect.
 - Via headers are now restored from the outbound packet by matching the branch header.
 - Support for multiple Contact headers in a registration.
- 3.2a13
 - Auth Digest ruleset added.
 - Bugfix in Slipper::Pickup preventing Call Pickup under some circumstances.
 - Now copies the tags in the To header correctly when creating a BYE after a 200/CANCEL race condition.
 - Bugfix in Slipper::MusicOnHold where a call is rapidly placed on and off hold and multiple RTP senders exist, the last one might not be cleaned up correctly if a previous sender was cleaned up after the next one was started.
- 3.2a14
 - Internal modules now supported as SIP targets.
 - Slipper::SimpleVM module and `voicemail` application added to distribution.
 - RFC3581 “rport” support added.
 - Bugfix in Slipper::MusicOnHold. Two RTP streams would be generated where a retransmit of the INVITE from the party requesting the music on hold was received after Slipper sent the 200 OK.
 - Better handling and reporting of near-empty packets.
 - Now reports source IP address and port for packets that failed parse checks.
 - Handling the reduction of offering of PRACK’s in responses now mirrors that of requests.
 - `slipper-admin.cgi` has support for Cisco 7905/7912, and improved support for Cisco 79x0 phones.
- 3.2b1
 - Feature freeze in core Slipper code.
 - `voicemail` default file `vm/default.au` added to distribution
 - `slipper-admin.cgi` now adds `voicemail.db` on system configuration, `slipper.rules` modified to look up this database.
 - Bugfix in process cleanup for Slipper::SimpleVM.

- Where one element of an expansion has an explicit delay, and the other elements are registrations that are all currently invalid, Slipper used to wait for the explicit delay, but now it short-circuits the delay.
 - `ld-template.txt` configuration template file used by `slipper-admin.cgi` added to distribution.
- 3.2b2
 - `slipper-admin.cgi` modified to handle spaces in strings for Cisco 79x0 class phones.
 - Bugfix in `Slipper::MusicOnHold` preventing sessions from being cleaned up correctly. Only operational impact is excessive system calls to “waitpid”, so it may not be obvious to most users.
- 3.2b3
 - The Bugfix for `Slipper::MusicOnHold` in 3.2b2 unfortunately introduced a new bug not picked up in initial testing. This has been fixed.
 - Installation assistants `slipper-assist.tgz` and `cisco-tftpboot.tgz` referenced in the documentation, previously you had to know to ask us for them.
 - `slipper-admin.cgi` now lowercases the left-hand-side of any databases so the case-insensitive matching works (which lowercases the thing it’s looking up before looking for a match.)
- 3.2b4
 - Bugfix in `Slipper::Pickup` preventing Call Pickup where the first character of the Call-ID to be picked up is non-numeric. A related incorrect sorting problem was also fixed.
 - Common code in handling requests moved to `Slipper::Support` so that other modules (such as `Slipper::Pickup`) can also use it. This solves the problems some phones had in transferring calls that were picked up as Record-Route is handled better by such plug-in modules.
 - Hiding of the “User-Agent” and “Server” headers now turned off by default.
 - Minor cleanup of the `Status_File` code.
 - Bugfix - if the maximum number of hops were exceeded, Slipper would note that it was dropping the packet in the logs, generate a bounce, but still proceed to process the packet.
 - Slipper now does a better job of sending CANCEL’s to the same place the the original request went to, and sourcing it from the same IP address.
 - Bugfix - now uses the correct “;expires=” parameter when multiple Contacts are received in the one registration. Previously it used the first value for all entries. Also, this parameter is now preferred over the “Expires” header.

- `slipper.rules` change to include contents of the Diversion header (if present) with the display name portion of the Remote-Party-ID. Related bugfix in core Slipper to better focus on Display portions with multiple instances of `j;`'s.
- 3.2b5
 - Registration replication added for clustering - `Replication`, `ReplicationAliases`, `ReplicationKey`, and `ReplicationTolerance` all added.
 - Bugfix - `rewrite_uri_display` would crash if passed a string instead of the expected array of strings, which is officially allowed to make module writer's lives easier.
 - Bugfix - instant messaging no longer confuses the `Status.File` code.
 - `voicemail` helper application now presents the Diversion header differently if a leading `username /` is present, to better reflect the new information embedded as of 3.2b4.
 - `compile-c7905-tftp` added to distribution. This script automatically recompiles all phone-specific configurations for Cisco 7905G and 7912G phones when the `ld-template.txt` file or other dependency file is modified.
 - `slipper-admin.cgi` now removes unexpected characters such as spaces and quotes from the left-hand-side of database updates to reduce the potential impact of mistakes from untrained staff.
 - 3.2b6
 - `Slipper::OutboundRegister` added to the distribution.
 - The more generic “`Slipper::HelperApp`” replaces “`Slipper::SimpleVM`”.
 - “Force Socket” Ruleset added.
 - Changes to `call-accounting` to handle helper applications, and also to report failed calls as well as successful ones.
 - Special handling of “JUNK” when returned by the “Group” ruleset added.
 - Determination of whether to add the Divert header is now case-insensitive.
 - Fix problem where Contact header in 200 for REGISTER message incorrectly had a leading `j;`@ for anonymous URI's - the internal format indicating anonymous URI was incorrectly exposed. The Route ruleset is now called on the Contacts before returning them.
 - Added the “z9hG4bK” branch prefix as per RFC 3261.
 - `Replication` and `ReplicationAliases` now support hostnames (originally only IP's), and the smart-stripping feature added.
 - Record name of socket packet was received on if debug level is three or higher.
 - IP lookups now done for hostnames in the Aliases list.

- `slipper.cfg` file is now optional.
- `rtp_send` for Music on Hold now supports WAVE files as well (previously only AU format was supported.)
- Behaviour change - Variables used in the LHS of a ruleset match were automatically being treated as a backreference. Although this was the intended behaviour by the programmer, it was not documented and had caused confusion.
- “zero” and “load” commands added to the `db` utility.
- “Auth Digest” Ruleset had been added in 3.2a13 prematurely (before test cycle), now fully supported.
- Bugfix - restoration of original “From” headers would include the “;tag=” twice if the original header did not have `{}’s` and the response did. Also fixed a problem if multiple `{}’s` are present in the original header.
- Bugfix - `slipper-admin.cgi` had defaulted to 7905’s instead of the actual model number when editing the phone configuration for a 7912.
- Bugfix - `slipper-admin.cgi` would replace the RHS of a variable with a blank string when asked to edit it under some circumstances.
- Bugfix - fixed problem triggered by Windows Messenger sending a smaller Content-Length header than the actual size of the Content.
- Bugfix - the `db` utility did not correctly commit changes with some versions/installations of the underlying Berkeley DB implementation.
- Bugfix - CANCEL’s now retransmitted on cancelled legs of forked calls for which a 180 or 183 had been received. Previously, the CANCEL was transmitted once only and not retransmitted if a 200 OK to the CANCEL was not received.

- 3.2b7

- Redirects now handled internally by Slipper.
- “Slipper::Pickup” now sets the Diversion header (which in a default configuration is then included in the text portion of the Caller-ID) so one can tell whose call they are about to pickup.
- If a retransmit of a 200 OK to an INVITE is received and we already have received the ACK from the other end (same branch, which includes call leg tags and CSeq), short-circuit network ops and retransmit the stored ACK.
- Variable “PID_File” now treats the value “NONE” specially, and Variable “PID_File.Fail_OK” added.
- Variable “Suppress.Empty.Packet.Reports” added.
- “Slipper::DNS” now supports CNAME’s correctly, and won’t go into an infinite loop if the CNAME’s loop.
- Bugfix - Contact headers in responses with tags inside `{}’s` caused corruption of the header. Bug had been introduced in 3.2b6.

- Bugfix - missing “;” from the RFC 3261 branch prefix additions in 3.2b6.
 - Bugfix - now correctly handles sending a 180/183 update when another possibility in a forked call that had returned 180/183 and was being used, later issues an error and drops out of the list of possibilities.
- 3.2b8
 - “Slipper::Sticky” added to the distribution.
 - :r modifier added to `rtp_send` for Music on Hold.
- 3.2b9
 - “Slipper::HelperApp” enhancements for end-of-session detection.
 - “Slipper::HelperApp” - introduce delay of 1/4 of a second when starting up before sending audio to make sure other end is ready to receive data.
 - “Slipper::HelperApp” modified to handle Cisco IOS sending multiple RTP DTMF end packets with the same timestamp but different sequence numbers (everything else when sending duplicate packets sends identical duplicate packets).
 - “Slipper::OutboundRegistration” modified to bring it in to line with what more VoIP carriers expect.
- 3.2b10
 - Quality of Protection support added to outbound authentication code.
 - Support for `WWW-Authenticate` headers added to outbound authentication code.
 - Changed default of `Suppress_Empty_Packet_Reports` to `True`.
 - “Slipper::HelperApp” now detects an application that is in a loop and not detecting end of media and will throw an exception.
 - Bugfix - duplicate “;tag=” prevention in 3.2b6 could be a little overzealous and remove more than it should.
- 3.2b11
 - Bugfix - External Server Timeouts not sent, introduced in 3.2b6.
 - “Slipper::HideAddress” module added.
 - Documentation - added programming interface for “Slipper::HelperApp”
 - Documentation - added “SIPphone” section
- 3.2b12
 - Bugfix - handling of strict routing with three or more routes present was broken.

- Bugfix “Slipper::HelperApp” - corrected use of SIP headers when generating a BYE.
- 3.2b13
 - “slipper-admin.cgi” has “Encode” option added to the “get-voicemail” database.
 - Added ability to zero variables and lists.
 - “Slipper::Pickup” added Group discrimination and comma support.
 - Documentation “Slipper::Pickup” added.
 - Bugfix “ha/get-voicemail” - fixed problem where canonical username is different from that returned by the “SimpleVM Get” ruleset.
- 3.2b14
 - Bugfix - “Rewrite ” Headers rulesets not called correctly in a transmitted request if a retransmission of the original request is received.
 - Bugfix “Slipper::HelperApp” - workaround for a problem on some Perl ports (perl/kernel integration issues) where the UDP port is reported as :0 in various headers.
- 3.2b15
 - “Slipper::NAT” module added to distribution
 - “Slipper::RingToneSelection” module added to distribution
- 3.2b16
 - Protection from mis-interpretation of valid regular expressions in some headers.
 - Bugfix - Packet loss of DNS requests to non-local servers was not handled correctly.
 - Bugfix - Internal status information for calls was removed after Call.Completed_Expiry time instead of Call_Expiry time if Status_File was not defined.
 - Bugfix - Slipper would not remove a call from Status_File if the it received an error code response to a BYE - usually triggered by a Cisco 7905/7912 bug.
- 3.2b17
 - DNS performance improvements.
 - Security Bugfix: NAPTR records with perl “execute code” regular expressions now discarded with an error message.
 - Variables “Type_of_Service_Signalling” and “Type_of_Service_Media” added.
 - The Contact header is now copied into an ACK if it is to the same URI as the original INVITE, and the ACK is missing the Contact header. Mainly for use by plug-in modules so they don’t lose information.

- Bugfix - new ports specified in reconfiguration now handled better.
 - Bugfix - memory leak in reconfiguration (old configuration kept around due to circular references) has been resolved.
 - Bugfix “ha/get-voicemail” - typo.
 - Bugfix “Slipper::HelperApp” - problem with transferring of calls introduced in 3.2b14.
 - Bugfix “Slipper::Pickup” - removed some spurious warnings.
- 3.2b18
 - Now Inherits authorisation information already verified in other calls specified in a Replaces header. Useful for context-based authorisation when the Call-ID is not maintained.
 - Variable “Fixup_Refer_To” (and the functionality it controls) added. Currently disabled by default as a transition mechanism.
 - “Slipper::ReferToFixup” added to the distribution. Combined with the variable “Fixup_Refer_To”, the transfer-rewrite database is no longer needed, and attended transfers to call pickups now work correctly.
 - Packet dumps of packets with null’s now has non-printable and unusual characters mapped to their hex equivalents when printing.
 - “slipper-admin.cgi” has “Encode” option added to the “auth” database.
 - Reformatting of the “slipper-admin.cgi” web page to decrease data downloads and thus improve performance over wide area links.
 - Bugfix - Some ENUM lookups were being incorrectly dropped.
 - 3.2b19
 - Group restriction feature added to Call Pickup module.
 - Changes to Codec negotiation to support payload size negotiation as used by Cisco’s Call Manager 4.x
 - Adding socket to the Record-Route header by default to work around what appears to be a bug Cisco’s Call Manager 4.x.
 - 3.2b20
 - OPTIONS requests made of the server itself now supported.
 - “On Register” ruleset added.
 - “Slipper::MWIcache” added to the distribution.
 - “pingsip” added to the distribution.
 - Slipper’s replication feature now supports databases other than the registrations database. This is intended to be used by “Slipper::MWIcache” and similar modules. There are now two replication formats (registration only and all database types), Slipper currently sends registrations in the old format for backwards compatibility, although this will change in a future release.

- “Slipper::Support::send_replication” added to “Slipper::Support”
 - Bugfix: Locally-generated final responses are now retransmitted to cope with cases where a 100 Trying is sent and received but a subsequent 407 Proxy Authentication Required message is lost in transit.
 - Bugfix: The Diversion header (calculated during expansions) was not being copied to “update” packets arriving later that re-used the existing expansion.
- 3.2b21
 - Change to the top of the Canonify Ruleset to handle messages from Cisco’s 79[46]0 7.5 code that has multiple `i;` pairs on the one line during registration.
 - Optimisation: Slipper now tries harder to deal with backed up quantities of input packets.
 - Bugfix: Slipper had moved forward expansion call legs to trigger sooner than they should if new DNS information became available for other potential call legs.
- 3.2b22
 - Bugfix: CANCEL’s to expansions would cause backup legs to legs that had not yet been activated (due to time delay) to trigger rather than never occurring. Bug introduced as part of DNS fixes in 3.2b21.
 - Bugfix: Only the first variable reference in the left hand side of a ruleset would be expanded.
 - `compile-c7905-tftp` now supports ATA’s.
 - `ata-template.txt` added to the distribution.
- 3.2b23
 - Variable “Registration_Grace_Time” added.
- 3.2b24
 - Behaviour change: Contact and Refer-To headers that fail canonification in a request now result in a bounce to the originator rather than the headers being silently stripped.
 - New: Variable “Authorise_All-Headers” added.
 - Bugfix: Fixes to HelperApp framework where the primary Slipper signalling port isn’t 5060.
- 3.2b25
 - Enhancement: Codec manipulation now supports a mix of specified and unspecified codecs.
 - Compatibility: loose or strict source routing now handles another proxy incorrectly stripping port information regardless of the presence of angle brackets (previously only worked if angle brackets not present.)

- Bugfix: HelperApp framework no longer waits longer than .25 of a second to receive a start RTP packet before returning to the calling application. Cisco 7905/7912 phones in monitor mode were not sending an RTP start packet and the application would wait for them.
 - Bugfix: HideAddress module was hiding the To header under some circumstances when it shouldn't have been.
- 3.2b26
 - New: “Input Channel” and “Output Channel” rulesets added.
 - Enhancement: HelperApp framework: Use of colon to select which DTMF tones can interrupt audio playback added.
 - Enhancement: HelperApp framework: arguments to “report_port” now accepted.
 - Enhancement: “Slipper::Sticky”: `accelerate:` option added.
 - Enhancement: `autoconf` added to Module in “Slipper::RuleSets”.
 - Enhancement: “get-voicemail” HelperApp now allows recording of personalised greeting.
 - Compatibility: MusicOnHold now handles a wider variety of “Call placed on hold” indicators.
 - Bugfix: Duplicate INVITE's were being sent to devices that had already responded with a 180 ringing after a DNS lookup had completed on another item in the same expansion. This was triggering a bug in Cisco 79[46]0 phones causing them to intermittently not send a 200 OK on picking up a call.
 - 3.2b27
 - New: `media_proxy` added to “Slipper::NAT” module.
 - New: `rtp_proxy` added to distribution.
 - New: `ha/state-ivr` added to distribution.
 - Bugfix: A device returning multiple 200 OK's to Slipper with a deep input queue could cause Slipper to not send CANCEL's to other devices in the same expansion.
 - 3.2b28
 - Bugfix: Duplicate ACK's were being unnecessarily sent to members of an expansion that did not pick up the call when subsequent requests were sent which required a DNS lookup.
 - 3.2b29
 - Enhancement: Slipper::Helperapp: arguments as filenames now supported.
 - New: `ha/new-number` added to the distribution.
 - 3.2b30

- New: “Slipper::Queueing” added to the distribution.
 - Enhancement: “Slipper::Support::write_to_log” added to “Slipper::Support” and to the “Null” ruleset to assist in debugging.
 - Enhancement: Slipper::MusicOnHold: now supports multiple source music files.
- 3.3r0
 - Directory structure reorganised.
 - Enhancement: Conferencing added to the HelperApp framework.
 - Enhancement: `call-accounting`: `-e` flag added.
 - Enhancement: `status-file` now has an enhanced format embedding sufficient information that a call transfer or call hangup can be initiated by an external application.
 - New: `cgi/console.cgi` added to the distribution.
 - New: Variable “Status_File_Format” added.
 - New: `bin/migrate-3.2-to-3.3` added to distribution to aid the directory migration process.
 - New: `bin/conference-bridge` added to support enhancements to the HelperApp framework.
 - 3.3r2
 - Bugfix: Slipper::DNS bugfix on negatively cached entries.
 - New: Variable “Remove_Excess_Codecs” added.
 - 3.3r3
 - New: Slipper::NAT added `nat.db` to store outbound information from REGISTERS to fix up incorrect NAT responses to inbound calls.
 - New: Variable “Keepalive_Period” added.
 - New: Variable “NAT_Expiry_Period” added.
 - New: Variable “Strip_Transport” added.
 - New: `bin/udp_proxy.c` added as a lower overhead alternative to the Perl `bin/rtp_proxy`.
 - New: Slipper::HelperApp added functions `play_number` and `play_timestamp`.
 - Bugfix: Handling of mixed loose-source routing and strict-source routing in some combinations was not correct.
 - 3.3r4
 - New: Support for routegroups added.
 - New: Variable “NAT_Modify_Registration_Expiry” added.
 - Enhancement: Minor changes for hybrid environments with Slipper 4.

- Enhancement: `Slipper::ReferToFixup` now handles a wider range of mistakes by devices, primarily new behaviour by Cisco phones doing semi-attended transfers.
- Enhancement: Now overstamp all Date headers to work around interaction issues between devices that don't set it correctly and those that use it to set their internal clocks.
- Enhancement: `ha/voicemail` now supports IMAP4 as well as POP3.
- Bugfix: DTMF tones received while playing a prompt for which DTMF tones were ignored were accidentally queued up and then handed to the next thing that is interested in DTMF tones

- 3.3r5

- New: Variable “`Registration_Min_Time`” added.
- Enhancement: `Slipper::NAT` reworked to better handle multiple devices registering with the same username behind non-SIP aware firewalls.
- Documentation: State-IVR documentation (previously a separate document) merged into this document.

Appendix B

License

Following is the Australian License - contact Iagu or your local distributor for license terms in other countries.

END USER SOFTWARE LICENCE (LICENCE)

1. GRANT OF SUB-LICENCE

1.1 The Distributor, as Licensee, grants to you, the sub-Licensee, on the terms set out in this Licence a nonexclusive and personal right to use (for internal use only within the Commonwealth of Australia and subject to payment in full of any specified licence or multi-user fee) the Iagu Pty Ltd ACN 093 036 410 trading as Iagu Networks (IAGU) SLIPPER software delivered herewith (the Software) solely on the Distributors equipment purchased by you and for which such Software was designed, configured or adapted.

1.2 Your use and/or installation of the Software constitutes your acceptance of the terms of this Licence. If you do not accept the terms of this Licence you will have no right to use and will immediately cease using the Software.

2. OWNERSHIP OF SOFTWARE

2.1 As the sub-Licensee, you own the magnetic or other physical media on which the Software is originally or subsequently recorded or fixed, but is an express condition of this Licence is that IAGU retains title and ownership of the Software and all copies of the Software, regardless of the form or media in or on which the original and other copies may exist.

2.2 No intellectual property or other right in the Software is transferred to you by this Licence.

3. COPY RESTRICTIONS

3.1 The Software and any accompanying written material are the subject of copyright vested in IAGU. Unauthorized copying of the Software, including Software which has been modified, merged, or included with other software, or of the written materials and any alteration of the copyright or other proprietary notices appearing on the Software is expressly forbidden.

3.2 You may be held legally responsible by IAGU or the Distributor for any copyright infringement which is caused or encouraged by failure to abide by the terms of this Licence.

3.3 Subject to the restrictions above, you may take one (1) copy of the Software solely for backup purposes if and to the extent required.

3.4 You must reproduce and include the copyright notice on the backup copy.

4. USE RESTRICTIONS

4.1 You may not modify, reverse engineer (except to the extent permitted by mandatory public law), decompile, or create derivative works based on the Software or accompanying written materials without the prior written consent of IAGU and the Distributor.

5. TRANSFER RESTRICTIONS

5.1 The Software is Licensed only to you, the sub-Licensee, and may not be transferred to anyone without the prior written consent of IAGU and the Distributor.

5.2 Any transferee of the Software must agree in writing to be legally bound by the terms and conditions of this Licence.

6. DISCLAIMER OF WARRANTIES

6.1 Except as expressly provided in the agreement or purchase order acceptance for the equipment for which the Software was designed, and with which the Software must be used, the Software and accompanying written materials (including instructions for use) are provided as is without warranty of any kind.

6.2 IAGU and the Distributor do not warrant or make any representations regarding the use, or the results of use, of the Software, or written materials in terms of correctness accuracy, reliability, currentness, or otherwise.

6.3 All other warranties, expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose, are excluded to the full extent permitted at law.

7. LIMITATION OF LIABILITY

7.1 Neither IAGU nor the Distributor nor anyone else who has been involved in the creation, production or delivery of the Software shall be liable for any direct, indirect, consequential or incidental damages (including damages for loss of business profits, business interruption, loss of business information, and the like), arising out of the use or inability to use such product even if IAGU or the Distributor have been advised of the possibility of such damages.

8. TERMINATION

8.1 This Licence is effective until terminated.

8.2 This Licence will terminate automatically without notice from IAGU or the Distributor if you fail to comply with any provision hereof.

8.3 Upon termination you shall destroy the written materials and all copies of the Software, including back-up copies, if any.

9. ENTIRE AGREEMENT

9.1 This Licence and limited warranty is the complete and exclusive agreement between the parties and supersedes all proposals or prior agreements, oral or written, as to its subject matter.

10. GOVERNING LAW & EXCLUSIVE JURISDICTION

10.1 This Licence shall be governed and construed in accordance with the laws of South Australia and the Commonwealth of Australia.

10.2 Exclusive jurisdiction and venue for all matters relating to this Licence shall be in courts located in the State of South Australia.

10.3 You as sub-Licensee consent to such exclusive jurisdiction and venue by your use of the software.

Index

- Ahead Software, 98
- Call Pickup, 37
- Calling Line ID, 9
- Cisco
 - 7905 IP Phones, 98
 - 7912 IP Phones, 98
 - 7940 IP Phones, 94
 - 7960 IP Phones, 94
 - Analogue Telephone Adapters, 97
 - CDP, 94
 - IOS
 - Known Problems, 92
 - Routers, 86
- Code, 28
- Comments, 33
- Dial Plan, 5, 26
- DNS SRV, 8, 11, 62
- Error, 30
 - Ignore, 30
- Expansion, 4
- Files
 - ld-template.txt, 64, 98
 - replication.rules, 43, 63
 - slipper.bmp, 103
 - slipper.cfg, 8
 - slipper.gif, 103
 - slipper.rules, 8, 12, 36, 64, 68
 - slipper.webconfig, 36, 64
- H.323, 4
 - Gatekeeper, 4
- Header
 - Call-ID, 35
 - Contact, 16, 35
 - Remote-Party-ID, 15
- Header.Refer-To, 16
- HelperApp
 - close.audio, 53
 - collect_dtmf, 52
 - decode_dtmf, 54
 - find_vm_target, 52
 - media_inactive, 53
 - new_stream, 50
 - play_audio, 51
 - read_audio_file, 53
 - receive_end_of_media, 54
 - record_audio, 51, 52
 - report_port, 50
 - send_dtmf, 51
 - send_mwi, 53
 - send_rtp, 53
 - send_vm, 52, 54
 - transfer, 52
 - vm_get_from, 52, 54
 - vm_get_reason, 52, 54
- IETF, 4
- Instant Messaging, 5, 99
- Interactive Voice Response, 50
- Introduction, 4
- Keyword
 - Program, 59
- Keywords
 - AUTH, 14
 - DISPLAY, 16
 - JUNK, 14, 19
 - REGISTER, 15
 - URI, 15
- Line Continuation, 34
- Media Proxy, 69
- Method, 28
- Microsoft
 - XP Messenger, 99
 - Known Problems, 99
- Mitel, 100
- Music on Hold, 38
- NAT, 69
- OutboundRegister, 39
- play_audio, 53
- Programs
 - call-accounting, 65, 107
 - check_ruleset, 68
 - db, 32, 67
 - slipper-admin.cgi, 64, 98

- slipper.sh, 10, 61
- Reason, 28
- Redirects, 33
- Replication, 62
- RFC
 - 2543, 4
 - 3261, 4
- Rulesets, 4
 - Auth Digest, 13, 39
 - Authenticate, 12
 - Authorise, 12
 - Canonify, 13
 - Codec, 13
 - Expand, 14
 - Force Socket, 14, 42
 - Group, 14
 - Input Channel, 14
 - Input Filter, 14
 - On Register, 15
 - Output Channel, 15
 - Output Filter, 15
 - Register, 15
 - Rewrite Contact, 16
 - Rewrite Display, 16
 - Rewrite Refer-To, 16
 - Rewrite URI, 15
 - Route, 16, 103
- SimpleVM, 40
- SIP
 - proxy server, 4
- SIPphone, 101
- SIPPS, 98
- State-IVR, 55
 - Audio
 - ReadAmount, 57
 - ReadCharacters, 57
 - ReadDigits, 57
 - ReadLetters, 57
 - ReadNumber, 57
 - Caller ID, 55
 - Keyword
 - Audio, 57
 - Collect, 59
 - Default, 58
 - DTMF Digits, 58
 - If, 58
 - Immediate, 58
 - Masks, 60
 - MaxStateCount, 60
 - Set, 56
 - State, 56
 - Transfer, 59
 - Log File, 55
 - Process ID, 55
 - Timestamp, 55
- URI, 28
- Variables
 - Alias_Resolve, 17
 - Aliases, 24
 - Allow_Unauthenticated_ACK, 17, 103
 - Auth_User, 28
 - Authorise_All_Headers, 17
 - Bind, 24
 - Call_Completed_Expiry, 18
 - Call_Expiry, 17
 - Comma_Delay, 18, 31
 - Debug_Level, 18
 - Default_Register_Expires, 18
 - Digest_Validity, 18
 - Fixup_Refer_To, 18, 43
 - HelperAppContact, 19
 - HideAdress_Grace_Time, 44
 - HideAdress_Host, 44
 - Invite_Retransmit_Delay, 19
 - Junk_For, 19
 - Keep_Sticky, 40
 - Keepalive_Period, 19
 - KeepAlive_Period, 69
 - Log_File_Name, 19
 - MAIN_PROCESS, 19
 - Max_Delay_Factor, 19
 - Max_Header_Length, 20, 35
 - Max_Retransmits, 20
 - NAT_Expiry_Period, 20
 - NAT_Modify_Registration_Expiry, 20
 - PID_File, 20
 - PID_File_Fail_OK, 20
 - Proxy_Auth_Header, 28
 - Proxy_Auth_Require, 28
 - Random_Cluster, 20, 35
 - Realm_Name, 12, 20, 62
 - Registration_Grace_Time, 21
 - Registration_Min_Time, 21
 - Registrations_File, 21
 - Remote, 28
 - Remote_Group, 28
 - Remove_Excess_Codecs, 21
 - Replication, 25, 62

Replication_Aliases, 25, 62
Replication_Key, 21, 62
Replication_Tolerance, 21, 62
Request_Retransmit_Delay, 21
Rewrite_From_Header, 15, 21
Rewrite_Remote_Party_ID_Header,
15, 21
Rewrite_To_Header, 15, 21
Server_Name, 22
Status_File, 22
Status_File_Format, 22
Stay_In_Path, 23, 99
Strip_Transport, 23
Suppress_Empty_Packet_Reports, 23
Tick, 25
Type_of_Service_Media, 24
Type_of_Service_Signalling, 24
Voicemail, 50