

## **Introduction**

This paper will explore how to configure and setup the DUNDi directory service on your Asterisk PBX system. DUNDi is not very hard to configure in Asterisk, however at the time of this writing, there is no step-by-step guide to getting it configured. I decided to write this paper based on my experiences with DUNDi over the last week. It is hoped this paper will help get you up to speed on using this wonderful directory service.

## **What is DUNDi?**

From the DUNDi website, "DUNDi™ is a peer to peer system for locating Internet gateways to telephony services. Unlike traditional centralized services (such as the remarkably simple and concise ENUM standard), DUNDi is fully distributed with no centralized authority whatsoever".

What DUNDi does is allow you to create a mesh of servers which can share information about their extensions with each other and how they can be reached. For example, lets say that I have 2 sites – Montreal and Toronto. Each of these sites has multiple local extensions and access to the local PSTN. We want to create a peering system which allows us to share information about our local extensions and local PSTN numbers. This creates a directory of numbers that are reachable on both peers. We can then perform lookups on the remote destinations directory service to see if the number we are dialing is either reachable there, or reachable through there (such as in the case when we want to allow our peer to connect to the PSTN through our Asterisk server).

DUNDi makes life much easier when we want to advertise extensions between servers as it alleviates the need to make configuration changes in the dialplan on all our servers.

## **Configuring Asterisk for use with DUNDi**

There 3 files which need to be configured for DUNDi: [dundi.conf](#), [iax.conf](#) and [extensions.conf](#). These files will be explored and configuration examples will be given for peering with the DUNDi Test group and creating a private group for internal extensions (which are not advertised to the DUNDi Test group).

The first thing we are going to do is setup a peering arrangement between our Toronto and Montreal offices so that we can call the local extensions between offices. We have dedicated the 1000 block to the Toronto office (numbers 1000 -> 1999) and the 2000 block for the Montreal office.

Lets dive into the dundi.conf file and get our Toronto office ready to go. The example dundi.conf file is very well done. It explains most things and a lot can be left as the default. Comments are interspersed where things need to change and it is assumed that you have read the example comments (in order to avoid duplicating explanations).

# dundi.conf

## General Configuration

```
;
; DUNDi configuration file
;
;
[general]
;
; The "general" section contains general parameters relating
; to the operation of the dundi client and server.
;
; The first part should be your complete contact information
; should someone else in your peer group need to contact you.
;
```

Make this section meaningful.

```
department=IT
organization= toronto.example.com
locality=Toronto
stateprov=ON
country=CA
email=support@toronto.example.com
phone=+19055551212
;
;
; Specify bind address and port number. Default is
; 4520
;
;bindaddr=0.0.0.0
port=4520
;
; Our entity identifier (Should generally be the MAC address of the
; machine it's running on. Defaults to the first eth address, but you
; can override it here, as long as you set it to the MAC of *something*
; you own!)
;
```

I generally make this the MAC address of my primary external interface.

```
entityid=00:48:54:1D:E6:34
;
; Define the max depth in which to search the DUNDi system (also max # of
; seconds to wait for a reply)
;
ttl=32
;
; If we don't get ACK to our DPREQUEST within 2000ms, and autokill is set
; to yes, then we cancel the whole thing (that's enough time for one
; retransmission only). This is used to keep things from stalling for a long
; time for a host that is not available, but would be ill advised for bad
; connections. In addition to 'yes' or 'no' you can also specify a number
; of milliseconds. See 'qualify' for individual peers to turn on for just
; a specific peer.
;
autokill=yes
;
; pbx_dundi creates a rotating key called "secret", under the family
; 'secretpath'. The default family is dundi (resulting in
; the key being held at dundi/secret).
;
;secretpath=dundi
```

## Creating DUNDi Mapping Contexts

DUNDi has contexts similar to the contexts that you find in extensions.conf. It is essentially a way of creating different directory groups. For instance, our example is going to contain two different contexts – one for the dundi-test group and one for private extensions. The contexts in the mapping section point to contexts in the extensions.conf file. In the extensions.conf file contexts, you will control the numbers which you advertise. When a peer is created, you will define which mapping contexts you will allow this peer to search. You do this with the `permit` statement.

For our Toronto and Montreal office peers we will permit them to search for numbers within our private and dundi-test groups. For our DUNDi Test group peers, we will only allow them to search for numbers included in the mapping context called dundi-test, which points to a context in extensions.conf listing the PSTN numbers which we own or have leased and have the right to advertise. These must be in the format of `<country_code><area_code><prefix><number> - 14165551212`.

```
[mappings]
;
; The "mappings" section maps DUNDi contexts
; to contexts on the local asterisk system. Remember
; that numbers that are made available under the e164
; DUNDi context are regulated by the DUNDi General Peering
; Agreement (GPA) if you are a member of the DUNDi E.164
; Peering System.
;
; dundi_context => local_context,weight,tech,dest[,options]]
;
; dundi_context is the name of the context being requested
; within the DUNDi request
;
; local_context is the name of the context on the local system
; in which numbers can be looked up for which responses shall be given.
;
; tech is the technology to use (IAX, SIP, H323)
;
; dest is the destination to supply for reaching that number. Note
; that the variable ${NUMBER} shall be replaced by the number being
; requested.
;
; Further options may include:
;
; nounsolicited: No unsolicited calls of any type permitted via this
;                 route
; nocomunsolicit: No commercial unsolicited calls permitted via
;                 this route
; residential:   This number is known to be a residence
; commercial:    This number is known to be a business
; mobile:        This number is known to be a mobile phone
; nocomunsolicit: No commercial unsolicited calls permitted via
;                 this route
; nopartial:     Do not search for partial matches
;
; There *must* exist an entry in mappings for DUNDi to respond
; to any request, although it may be empty.
;
;e164 => dundi-e164-canonical,0,IAX2,dundi:${SECRET}@${IPADDR}/${NUMBER},nounsolicited,nocomunsolicit,nopartial
;e164 => dundi-e164-customers,100,IAX2,dundi:${SECRET}@${IPADDR}/${NUMBER},nounsolicited,nocomunsolicit,nopartial
;e164 => dundi-e164-via-pstn,400,IAX2,dundi:${SECRET}@${IPADDR}/${NUMBER},nounsolicited,nocomunsolicit,nopartial
```

Here we define our DUNDi mapping contexts. We have two groups; private and dundi-test. The private context will contain our locally reachable extension numbers. In Toronto it will be the 1000 block, and in Montreal, the 2000 block.

```
priv => dundi-priv-
local,0,IAX2,priv:${SECRET}@toronto.example.com/${NUMBER},nounsolicited,nocomunsolicit,nopartial
```

When a peer does a lookup for a number which we are advertising, they are passed back information for how to connect to that number. For example, if someone in the Montreal office wanted to connect to extension 1000, the Asterisk system would do a lookup in the private dundi group. The Toronto office would return how to connect to that extension, something like the following. You can perform dundi lookups at the CLI of Asterisk by typing 'dundi lookup <number>'. Note that local lookups do not work from the CLI, however, remote lookups do.

```
1. 0 IAX2/dundi:qR3WLYh7+gtN3exqjPwpLg@toronto.example.com/1000
(EXISTS|NOUNSLCTD|NOCOMUNSLTD)
    from 00:00:00:00:00:00, expires in 787 s
DUNDi lookup completed in 731 ms
```

Below we define our mapping context for the dundi-test group. We can only advertise numbers for which are reachable. A quick search on the definition of canonical returns, "reduced to the simplest and most significant form possible without loss of generality".

Here we have two dundi-test groups, both of which are pointing at different contexts with different weights. We advertise any number within the [dundi-test-canonical] extensions with a weight of 0. This means that these numbers terminate locally. Because I have access to the 416 and 905 area codes, I have advertised these numbers as accessible from my box with a weight of 100. This ensures that anyone else who has better access to the PSTN network for the 905 and 416 area codes make themselves a more preferred route.

```
dundi-test => dundi-test-
canonical,0,IAX2,dundi:${SECRET}@toronto.example.com/${NUMBER},nounsolicited,nocomunsolicit,nopartial
dundi-test => dundi-pstn-local,100,IAX2,dundi:${SECRET}@toronto.example.com/${NUMBER},nounsolicited,nocomunsolicit
```

## Defining DUNDi Peers

Here we will define our peers in the dundi.conf file. Peers are identified by a unique layer 2 MAC address of an interface on the system. This is where we define what peers we use when doing a lookup for a group.

```
;
; The remaining sections represent the peers
; that we fundamentally trust. The section name
; represents the name and optionally at a specific
; DUNDi context if you want the trust to be established
; for only a specific DUNDi context.
;
; inkey - What key they will be authenticating to us with
;
; outkey - What key we use to authenticate to them
;
; host - What their host is
;
; order - What search order to use. May be 'primary', 'secondary',
;         'tertiary' or 'quartary'. In large systems, it is beneficial
;         to only query one up-stream host in order to maximize caching
;         value. Adding one with primary and one with secondary gives you
;         redundancy without sacraficing performance.
;
; include - Includes this peer when searching a particular context
;           for lookup (set "all" to perform all lookups with that
;           host.
;
; noinclude - Disincludes this peer when searching a particular context
```

```

;           for lookup (set "all" to perform no lookups with that
;           host.
;
; permit - Permits this peer to search a given DUNDi context on
;           the local system. Set "all" to permit this host to
;           lookup all contexts.
;
; deny -   Denies this peer to search a given DUNDi context on
;           the local system. Set "all" to deny this host to
;           lookup all contexts.
;
; model - inbound, outbound, or symmetric for whether we receive
;           requests only, transmit requests only, or do both.
;
; The '*' peer is special and matches an unspecified entity
;

[FF:FF:FF:FF:FF:FF] ; Montreal Office
model = symmetric
host = montreal.example.com
inkey = montreal
outkey = toronto
include = private
include = dundi-test
permit = private
permit = dundi-test
qualify = yes
dynamic=yes

```

Here we have an example of a dundi peer. The remote peers identifier is enclosed in square brackets [ ]. The inkey and outkey are the public/private keypairs that we use for authentication. These are generated with the astgenkey script located in the `./asterisk/contrib/scripts/` source directory. Be sure to use the `-n` flag so that you don't have to initialize passwords every time you start Asterisk.

```

# cd /var/lib/asterisk/keys
# /usr/src/asterisk/contrib/scripts/astgenkey -n toronto

```

This will place a `toronto.pub` and `toronto.key` in your `/var/lib/asterisk/keys/` directory. The `toronto.pub` file is your public key that you should post on a webserver. When you go to peer, you can give them the http accessible public key where they would place it in their `/var/lib/asterisk/keys/` directory.

After you have downloaded the keys you must reload the `res_crypto.so` and `pbx_dundi.so` modules in Asterisk.

```

*CLI> reload res_crypto.so
-- Reloading module 'res_crypto.so' (Cryptographic Digital Signatures)
-- Loaded PRIVATE key 'toronto'
-- Loaded PUBLIC key 'toronto'

*CLI> reload pbx_dundi.so
-- Reloading module 'pbx_dundi.so' (Distributed Universal Number Discovery (DUNDi))
== Parsing '/etc/asterisk/dundi.conf': Found

```

We must now create the dundi and priv users to allow connections into our Asterisk system. When a call is authenticated, the extension number being requested is passed to a context in the `extensions.conf` file where the call is then handled by Asterisk.

## iax.conf

Below are the user definitions for the priv and dundi users. Note that the dbsecret is the same for both users. Asterisk regenerates keys every 3600 seconds (1 hour). This is the key which is advertised in the `SECRET` variable defined within the mapping context lines in dundi.conf. You can see the current keys for all peers including your local public and private keys by doing a 'show keys' at the Asterisk CLI.

```
[priv]
type=user
dbsecret=dundi/secret
context=dundi-priv-incoming
disallow=all
allow=ulaw
allow=g726

[dundi]
type=user
dbsecret=dundi/secret
context=dundi-test-local
disallow=all
allow=ulaw
allow=g726
```

The context entry is where authorized callers are sent in extensions.conf. From there we can manipulate the call. Examples will be shown in the following section.

## extensions.conf

The extensions.conf file handles what numbers you advertise and what you do with the calls that connect to them. The following examples are meant to be used in an already existing dialplan – namely mine. Your logic may differ. For my situation, all local extensions use SIP. All outgoing calls are also sent out over VoIP so there won't be any mention of zaptel devices. However it should be fairly trivial to add them in.

```
-----
; Macro Block
;-----
[macro-stdexten]
; standard extension macro
exten => s,1,Answer
exten => s,2,Dial(SIP/${ARG1},25,t)
exten => s,3,Goto(s-${DIALSTATUS},1)
exten => s-NOANSWER,1,Voicemail(u${ARG1})
exten => s-NOANSWER,2,Hangup
exten => s-BUSY,1,Voicemail(b${ARG1})
exten => s-BUSY,2,Hangup
exten => _s.,1,Goto(s-NOANSWER,1)
exten => a,1,VoicemailMain(${ARG1})

[macro-dundi-lookup]
; Goto the extension number. Check the local context first, followed by lookup
; dundi-priv-lookup is a pointer to the switch statement which will look for
; extensions on other machines. This allows the convergence of multiple
; Asterisk servers with different extension number blocks. Very cool!
;
exten => s,1,Goto(${ARG1},1)
include => dundi-priv-local
include => dundi-priv-lookup
```

The macros above are what do most of the work of calling numbers. The `stdexten` macro is used when we want to call local extensions. The `dundi-lookup` macro is used to call extensions which are accessible by DUNDi means.

Below we have the directory service contexts. This is where we define our locally reachable extensions, PSTN numbers which resolve locally to us and the PSTN numbers we are advertising access to.

When a call comes in from the `dundi` user (defined in `iax.conf`) it is sent into the `[dundi-test-local]` context. Contained within is both the `[dundi-test-canonical]` and `[dundi-pstn-local]` contexts. The extension number is passed to this context for matching; first in the `[dundi-test-canonical]` followed by `[dundi-pstn-local]`. A similar operation is performed for the `priv` user in `iax.conf`.

The `[dundi-pstn-local]` context advertises the 416 and 905 area codes through the use of a pattern matching definition. The example below limits the number of simultaneous calls to 2. If there is already 2 calls, then the caller is dropped. This is not a very nice way of handling calls, but for now will at least limit the number of calls so that the service isn't abused (and to keep call quality high).

The switch statement in the `[dundi-priv-lookup]` context is what allows us to do a lookup for remote extensions. When a local extensions dials an extension number in the range 1000 through 2999, it is parsed by the `[local]` context which calls the `dundi-lookup` macro. This macro is essentially just a Goto statement but will match the `[dundi-priv-local]` context followed by `[dundi-priv-lookup]` which contains the switch statement that performs the channel lookup. If something is returned by the DUNDi lookup, then Asterisk will connect to it. For example, if we try to call extension 2001 from the Toronto office, then it should do a lookup to our Montreal office peer. The Montreal DUNDi service would then return how to access extension 2001 and the caller would be connected.

```
-----  
; Directory Service Contexts  
-----  
[dundi-test-canonical]  
; these are the numbers we advertise to dundi with a weight of 0 (directly  
; connected) These would be numbers that we own or lease.  
exten => 19050000000,1,Goto(pstn-in,s,1)  
exten => 19050000001,1,Goto(pstn2-in,s,1)  
exten => 14160000000,1,Goto(pstn2-in,s,1)  
  
[dundi-test-local]  
; if we want to do a local lookup of numbers we advertise to dundi-test, then  
; we can just look here  
include => dundi-test-canonical  
include => dundi-pstn-local  
  
[dundi-test-lookup]  
; if we want to lookup an external number, use this context. The switch  
; statement lets us search our peers for the number we are requesting  
switch => DUNDi/dundi-test  
  
[dundi-pstn-local]  
; allow access to the 416 and 905 area, but set a max channel ceiling of 2  
; ***NOTE***  
; This is very dirty and not very nice. We just dropped the caller without  
; telling them why. Eventually the caller will be placed into a queue  
; so that when a line becomes free, it will automatically direct them out  
exten => _1416NXXXXXX,1,SetGroup(PSTN-OUTBOUND) ; increase PSTN-OUTBOUND +1
```

```

exten => _1416NXXXXXX,2,CheckGroup(2)           ; check if <=1, else n+101
exten => _1416NXXXXXX,3,Dial(${LOCALTRUNK}/${EXTEN:1})
exten => _1416NXXXXXX,103,Wait(1)              ; too many callers, drop
exten => _1416NXXXXXX,104,Playback(goodbye)

exten => _1905NXXXXXX,1,SetGroup(PSTN-OUTBOUND)
exten => _1905NXXXXXX,2,CheckGroup(2)
exten => _1905NXXXXXX,3,Dial(${LOCALTRUNK}/${EXTEN:1})
exten => _1905NXXXXXX,103,Wait(1)
exten => _1905NXXXXXX,104,Playback(goodbye)

[dundi-priv-local]
; we only have extensions 1000 -> 1999 locally
exten => _1XXX,1,Macro(stdexten,${EXTEN})

[dundi-priv-lookup]
; Check our private peers for the exten #. Search 'priv' dundi context
switch => DUNDI/priv

[dundi-priv-incoming]
; when we get an incoming call from a private peer, it gets directed here
include => dundi-priv-local

;-----
; Outgoing Calls Contexts
;-----
[local]
; For extensions starting with 1000 -> 2999 do a dundi-lookup (private extens)
exten => _[1-2]XXX,1,Macro(dundi-lookup,${EXTEN})

```

## Conclusion

We have explored a few of the uses of the DUNDi protocol within Asterisk. I still consider this document in the rough draft stage, so any and all comments would be appreciated. The document can be found at <http://leifmadsen.com> and comments can be sent to [leif@leifmadsen.com](mailto:leif@leifmadsen.com).