Configuration of a distributed system as emerging behavior of autonomous agents

Configuration of a distributed system as emerging behavior of autonomous agents :

Questo documento illustra la strategia di gestione della configurazione della rete distribuita di Autistici/Inventati, modellata come una macchina a stati.

Sommario

Capitolo 1. Introduzione

Prima una brevissima nota di nomenclatura: chiameremo *servizio* un applicativo che ci permette di offrire delle risorse all'utilizzo pubblico, e *account* la specifica istanza di un servizio relativa ad un particolare utente.

Il modello di rete distribuita di Autistici/Inventati prevede svariate tipologie di informazione sulla configurazione del sistema, eccole elencate qua appresso in ordine decrescente di astrazione:

il database LDAP

> Qui sono contenute le informazioni su ciascun account abilitato, le quali si riflettono direttamente in altrettante configurazioni dei vari servizi.

la configurazione dei servizi

> L'altra fonte della configurazione sono i dati (file e procedure) gestiti mediante cfengine, che invece comprendono le informazioni relative sia a quegli account "speciali" (o *strutturali*), cioè quelli che appartengono all'infrastruttura e che dunque sono troppo particolari e complessi per rientrare nelle rigide possibilità offerte dal database, sia alla configurazione generale dei vari servizi, quella che integra ai vari *frammenti* di configurazione suddetti.

i dati degli utenti

> Infine ci sono i dati veri e propri, in genere questi sono memorizzati direttamente in qualche collocazione standard nel file system.

Queste informazioni devono in qualche modo essere utilizzate per generare la forma finale della configurazione dei servizi (i normali files di configurazione in `/etc`, per intendersi).

Il sistema di gestione degli account e dei servizi deve dunque permettere di intervenire sui dati ai vari livelli di astrazione necessari, e adeguare corrispondentemente il comportamento complessivo del sistema.

Capitolo 2. Principi

Questo capitolo introduce i vari principi che hanno guidato la progettazione dei meccanismi di configurazione di Autistici / Inventati.

2.1. Convergenza

Per *convergenza* di un meccanismo di configurazione si intende il suo essere libero da *feedback*: ad esempio, deve lasciare il sistema in uno stato noto, ed accettare qualsiasi stato iniziale (anche non corretto), per poter ripartire correttamente anche in caso di errore.

Il sistema potrà avere una capacità più o meno grande di correggere automaticamente gli errori presenti in uno stato iniziale non corretto, ciò che importa è però che gli errori non vengano propagati.

2.2. Modularità

La *modularità* di un sistema complesso come quello in questione è la caratteristica che ne permette la comprensione umana e perciò la verifica. Ogni singolo meccanismo è implementato separatamente (utilizzando quanto più possibile degli strumenti comuni) e isolato, in termini di effetti e di requisiti.

L'isolamento permette di verificare più facilmente il corretto funzionamento di un particolare meccanismo, ad esempio generando dei casi di test su misura.

Una decisione chiave del design del sistema è stata per l'appunto quella di utilizzare una suddivisione quanto più *ortogonale* possibile, intendendo con questo il fatto che ciascun meccanismo che opera sul sistema non debba dipendere dall'esito corretto di nessun altro processo.

2.3. Autonomia

In aggiunta al punto precedente, un altro aspetto fondamentale dell'impostazione del sistema è la *separazione* tra i vari agenti rispetto alla loro interconnessione. Avendo a che fare con una rete di agenti collegati fra loro da un mezzo sostanzialmente inaffidabile come Internet, è importante che essi siano autonomi e separati, cioè che ciascuno di loro sia in grado di agire sul sistema in modo indipendente dalla raggiungibilità degli altri.

Nella particolare situazione della rete distribuita è infatti possibile che uno o tutti gli altri nodi non siano raggiungibili, e che dunque una o più delle fonti di informazione elencate nel capitolo precedente non sia disponibile. In questo stato di cose (le condizioni "ambientali", diciamo), la strategia migliore ci è parsa quella di costruire sistemi che siano in grado di operare solo finché ci sono informazioni disponibili in merito: quando poi le informazioni complete diventino disponibili, sarà automatico proseguire con i compiti necessari.

Questa considerazione ci ha portato ad elaborare dei meccanismi che siano sempre *locali*, in cui l'unica interconnessione passa attraverso l'uniformità dei dati presenti nel database. In questo modo si accorciano i tempi di convergenza della configurazione, e si può essere sicuri che ciascun nodo sia autonomamente consistente.

Capitolo 3. La macchina a stati

Un sistema che rispetti i principi fin qui individuati può essere modellizzato come una *macchina a stati finiti*, in cui gli stati in questione siano memorizzati all'interno del database, e vi siano *agenti* su ogni nodo incaricati di manipolare questi stati ed agire sul sistema corrispondentemente.

Il comportamento complessivo di un sistema siffatto può essere anche molto complesso, ma lo schema di azione di ogni singolo agente è semplice e facilmente comprensibile.

Cominciamo prima a definire più esattamente quali sono le informazioni attribuibili ad un oggetto nel database.

3.1. Oggetti nel database

Il database LDAP (vedi Cap. 1) è essenzialmente un insieme di *oggetti* che corrispondono a differenti *account*, strutturati in un'organizzazione gerarchica atta a facilitarne l'amministrazione. Della gerarchia non ci occuperemo adesso, ci interessa approfondire le caratteristiche di questi oggetti che ne consentono l'integrazione nella macchina a stati.

Ogni oggetto possiede un *tipo* e un attributo che lo identifica univocamente (ad esempio per una casella di posta questo sarà l'indirizzo corrispondente), uno *stato*, che esprime il tipo di azione che la macchina a stati dovrà operare, e una serie di parametri ausiliari, come ad esempio il nodo su cui un determinato oggetto dev'essere attivato.

Ad ogni oggetto sono poi associati dei dati (quelli denominati *dati dell'utente* nel Cap. 1), in un modo che è strettamente dipendente dalla tipologia dell'oggetto stesso. Questa relazione non è necessariamente espressa in modo completo nel database, ma, nel nostro caso, è parte della logica pertinente agli agenti (per fare un esempio sciocco prendendo sempre una casella di posta, sono gli agenti a sapere che ad un determinato valore dell'attributo *mailStorageQualcosa* corrisponono i files e le directory sotto `/home/mail/VALORE`).

3.2. Stati possibili

Compatibilmente con le nostre necessità, abbiamo individuato un numero piuttosto esiguo di stati possibili per un oggetto nel database.

active

: Stato attivo. Un oggetto in questo stato deve essere configurato sul server a cui è assegnato.

inactive

: Stato inattivo. L'oggetto corrispondente è stato disabilitato. La presenza nel database di oggetti in questo stato (anziché la loro cancellazione completa e immediata) è utile sia per poter facilmente procedere alla riattivazione senza passare attraverso la procedura di creazione di un nuovo oggetto (che cancellerebbe i contenuti associati ad esso), sia per mantenere un contesto storico (ad esempio, per prevenire la creazione di una mailbox con lo stesso nome di una che è stata appena chiusa).

temporary

: Stato "temporaneo". Questo stato è utilizzato per gestire una particolare tipologia di trasferimento degli oggetti da un nodo ad un altro. Di questo parleremo più in dettaglio nel Capitolo X.

Il comportamento degli agenti è esclusivamente determinato dal valore del parametro di stato, dal tipo dell'oggetto (agenti differenti agiscono su oggetti differenti, in nome della modularità del sistema), e dai parametri che identificano il nodo a cui un oggetto è attualmente assegnato e, eventualmente, quello a cui era *precedentemente* assegnato.

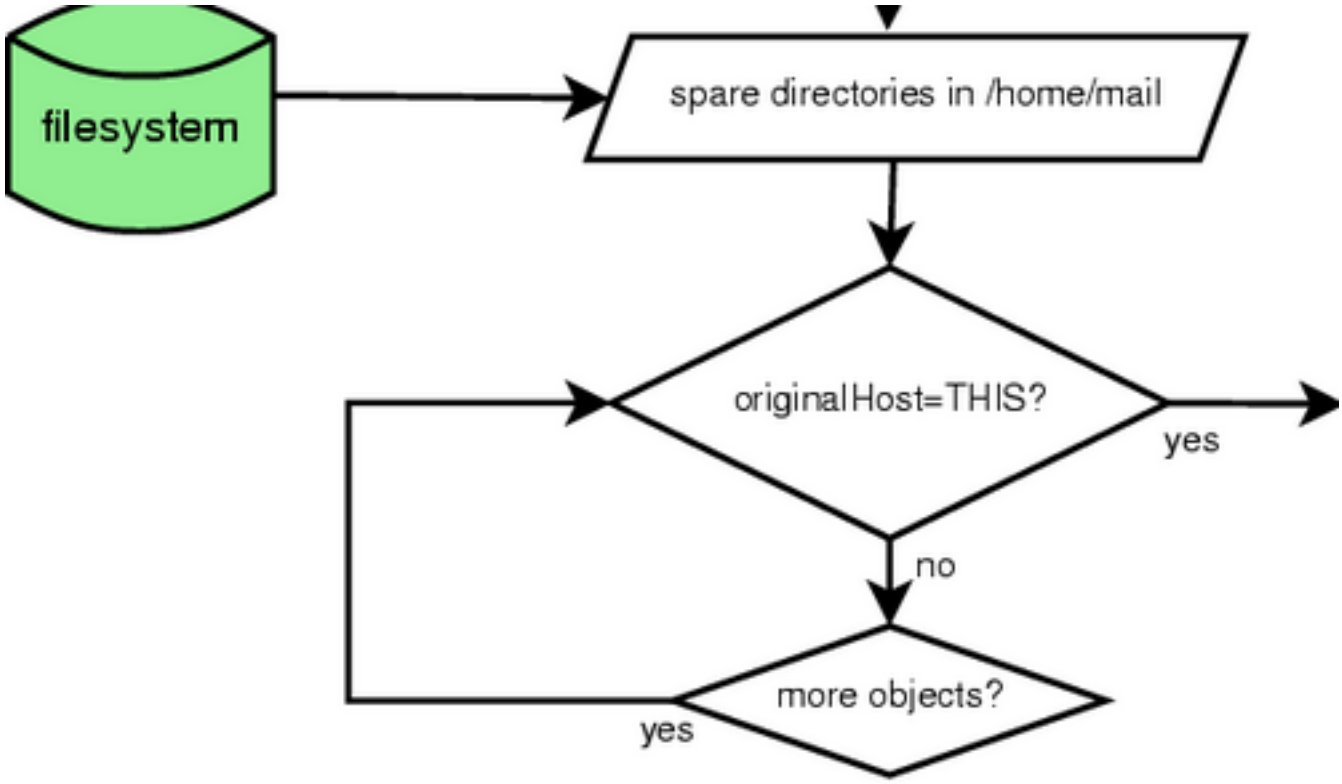
Questi parametri costituiscono dunque un *meta-stato*, che ne è la composizione cartesiana. Dato che ciascuno di questi parametri ha un insieme limitato di valori, il totale dei meta-stati è enumerabile. In particolare il parametro che associa l'oggetto ad un nodo può essere ridotto a due valori: uno quando il nodo in questione è lo stesso dell'agente, un altro quando è differente (dato che tutti i nodi sono equivalenti).

Appendice A. Esempi di workflow per gli agenti

Questa sezione fornisce alcuni esempi del flusso logico di alcuni agenti di configurazione. Ciascuno di essi ha una logica molto semplice, che generalmente parte dall'individuazione, mediante una query LDAP, del meta-stato di origine, e procede all'esecuzione di un'operazione specifica sugli oggetti in quel meta-stato. In alcuni limitati casi è possibile che, in caso l'operazione si sia conclusa con successo, il database LDAP venga aggiornato.

A.1. create-maildirs

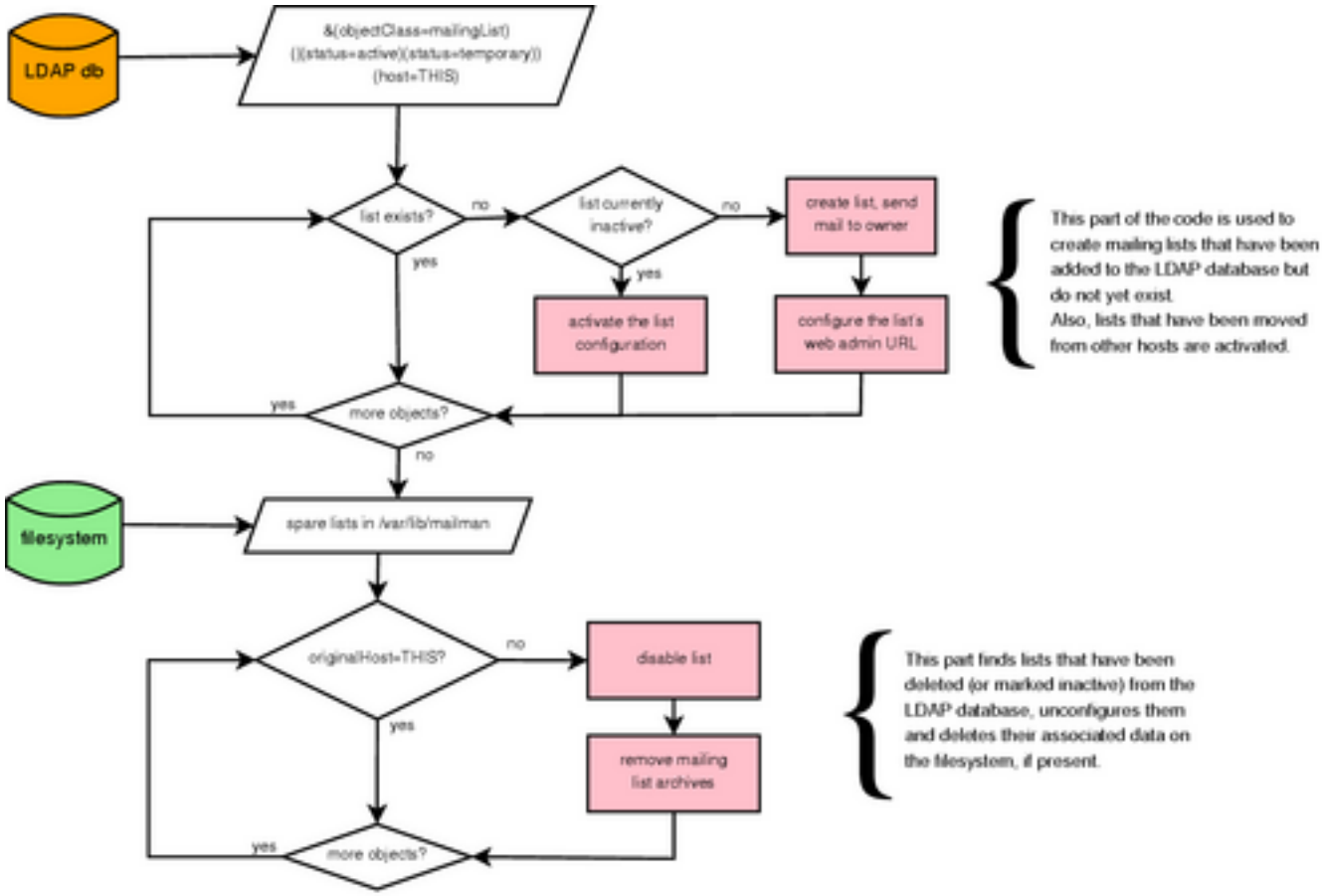
Scopo: creazione, nel filesystem, delle mailbox che sono state create nel database (o trasferite da un altro server).



create-maildirs flow

A.2. create-lists

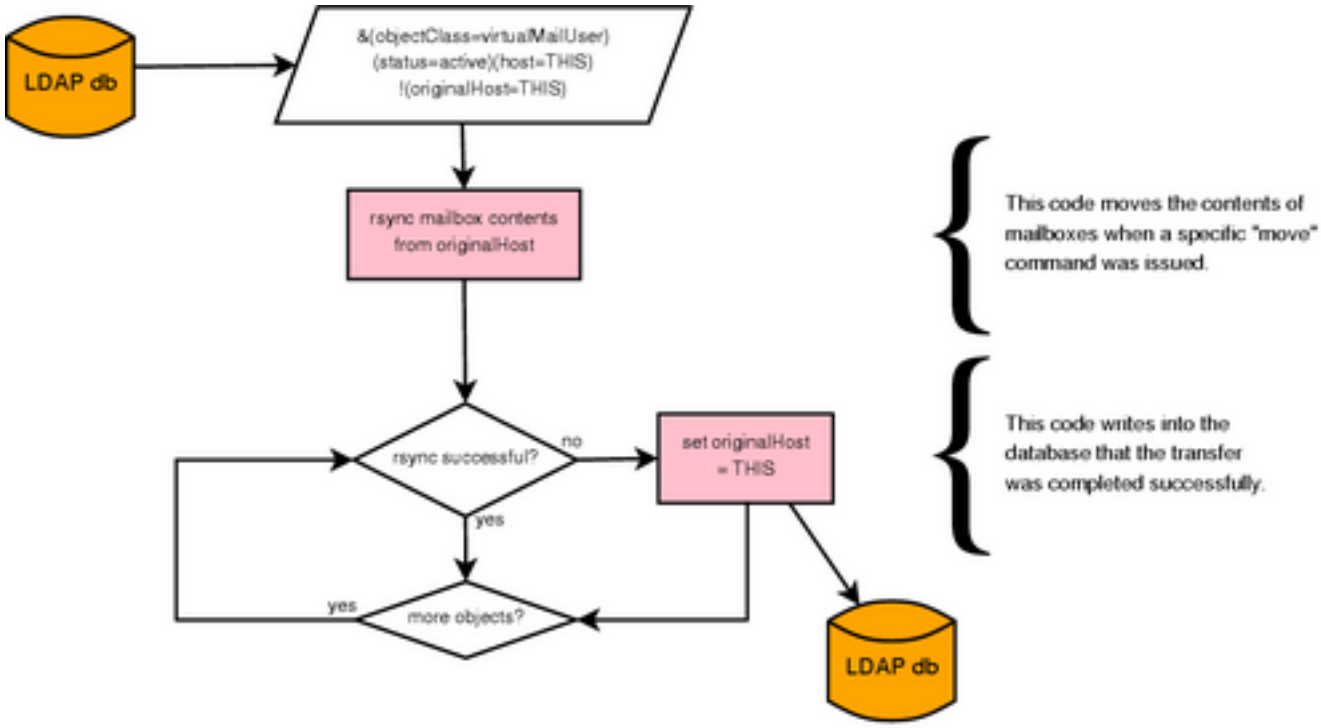
Scopo: creazione delle nuove liste di Mailman che sono state create nel database.



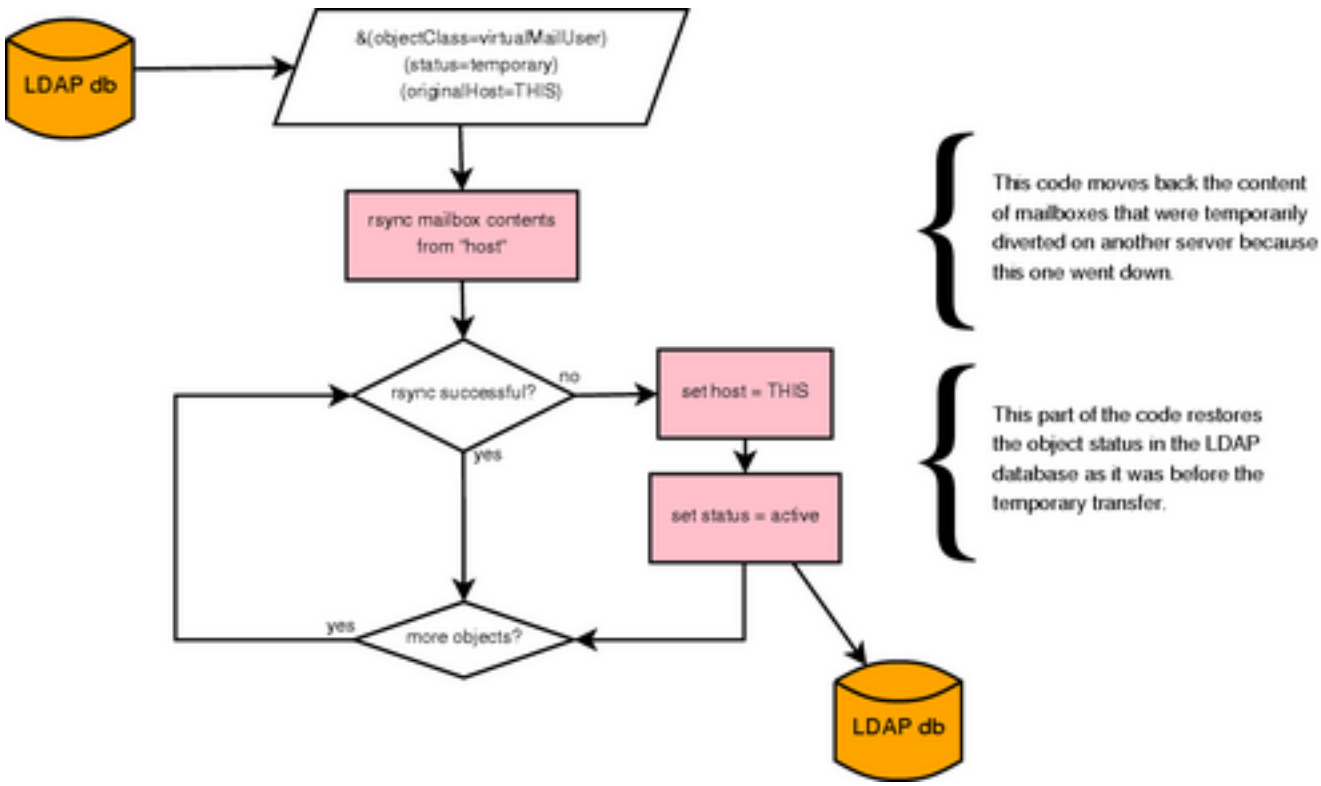
create-lists flow

A.3. move-maildirs

Scopo: trasferimento dei contenuti delle mailbox. Questo può avvenire in due direzioni: dall'host vecchio al nuovo (per gli spostamenti espliciti), oppure viceversa dall'host nuovo al vecchio, con conseguente riattivazione, per i trasferimenti temporanei.



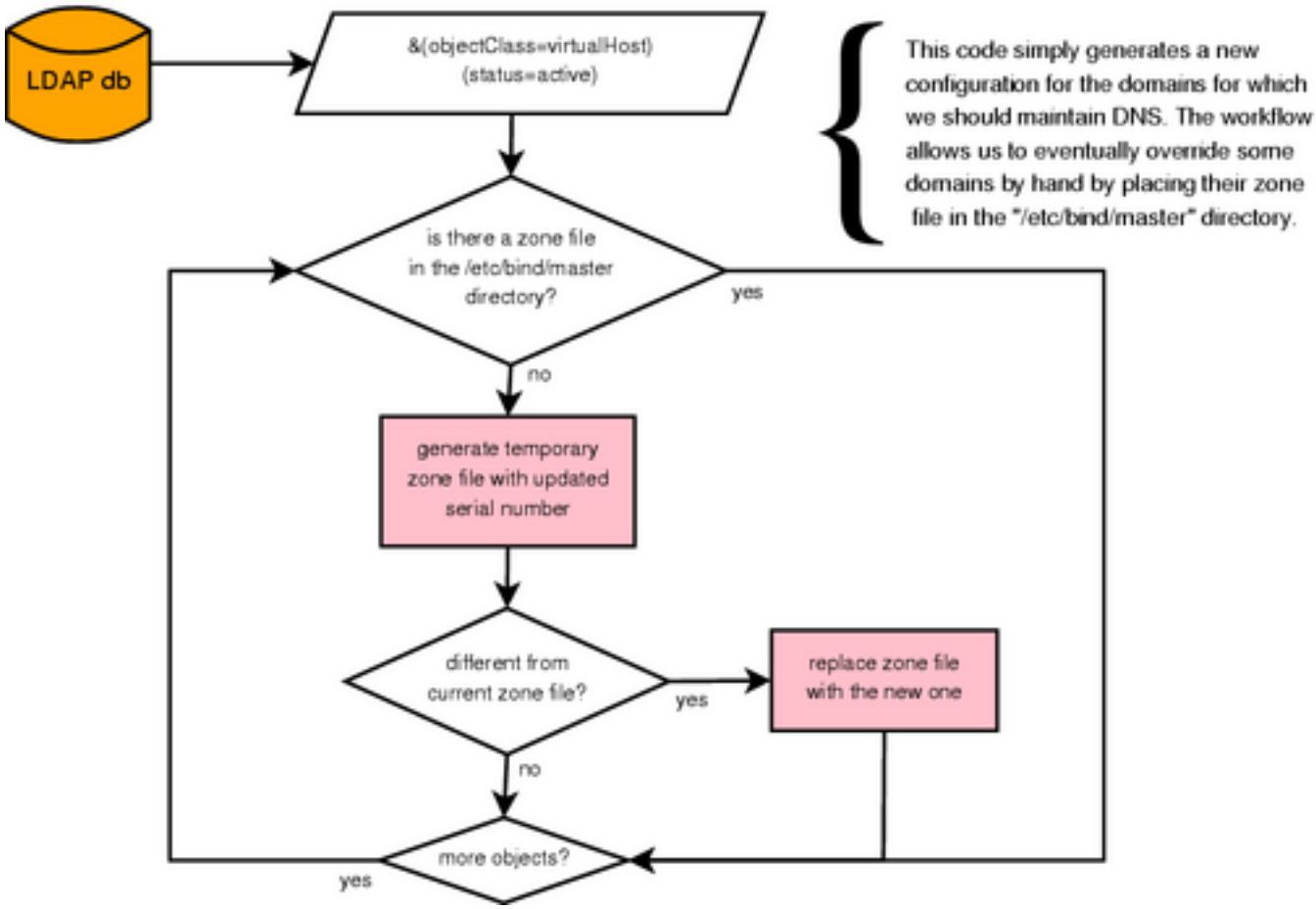
move-maildirs forward flow



move-maildirs reverse flow

A.4. config-dns

Scopo: configurazione automatica delle zone DNS per i domini presenti nel database.



config-dns flow