
ISC


BIND 9

BIND 9 管理员参考手册

发布 BIND 9.16.8（**稳定版**）

Internet Systems Consortium

翻译: sunguonian@yahoo.com

2021 **年** 02 **月** 16 **日**

Contents

第一章　介绍 1
第 1.1 节 文档范围 . . . 1
第 1.2 节 本文档的组织 . . . 1
第 1.3 节 本文档中的惯例 . . . 1
第 1.4 节 域名系统（DNS） . . . 2
1.4.1 DNS 基础 . . . 2
1.4.2 域和域名 . . . 2
1.4.3 区 . . . 3
1.4.4 权威名字服务器 . . . 3
1.4.5 缓存名字服务器 . . . 5
1.4.6 名字服务器的多个角色 . . . 5

第二章　BIND 资源要求 7
第 2.1 节 硬件要求 . . . 7
第 2.2 节 CPU 要求 . . . 7
第 2.3 节 内存要求 . . . 7
第 2.4 节 字服务器增强环境问题 . . . 7
第 2.5 节 支持的操作系统 . . . 8

第三章　名字服务器配置 9
第 3.1 节 样例配置 . . . 9
3.1.1 一个只缓存名字服务器 . . . 9
3.1.2 一个只权威名字服务器 . . . 10
第 3.2 节 负载均衡 . . . 11
第 3.3 节 名字服务器操作 . . . 11
3.3.1 用于名字服务器后台进程的工具 . . . 11
3.3.2 信号 . . . 14
第 3.4 节 插件 . . . 14
3.4.1 配置插件 . . . 14

3.4.2 开发插件 15

第四章 BIND 9 配置参考 17

第 4.1 节 配置文件元素 17

4.1.1 地址匹配表 19

4.1.2 注释语法 20

第 4.2 节 配置文件语法 21

4.2.1 acl 语句语法 22

4.2.2 acl 语句定义和用法 22

4.2.3 controls 语句语法 23

4.2.4 controls 语句定义和用法 23

4.2.5 include 语句语法 24

4.2.6 include 语句定义和用法 24

4.2.7 key 语句语法 24

4.2.8 key 语句定义和用法 24

4.2.9 logging 语句语法 25

4.2.10 logging 语句定义和用法 25

4.2.11 masters 语句语法 34

4.2.12 masters 语句定义和用法 34

4.2.13 options 语句语法 34

4.2.14 options 语句定义和用法 42

4.2.15 server 语句语法 95

4.2.16 server 语句定义和用法 96

4.2.17 statistics-channels 语句语法 98

4.2.18 statistics-channels 语句定义和用法 98

4.2.19 trust-anchors 语句语法 99

4.2.20 dnssec-keys 语句定义和用法 99

4.2.21 managed-keys 语句语法 104

4.2.22 managed-keys 语句定义和用法 104

4.2.23 trusted-keys 语句语法 105

4.2.24 trusted-keys 语句定义和用法 105

4.2.25 view 语句语法 105

4.2.26 view 语句定义和用法 105

4.2.27 zone 语句语法 107

4.2.28 zone 语句定义和用法 113

第 4.3 节 区文件 125

4.3.1 资源记录的类型及使用时机 125

4.3.2 对 MX 记录的讨论 . . . 127
4.3.3 设置 TTL . . . 128
4.3.4 IPv4 中的反向映射 . . . 128
4.3.5 其它区文件指令 . . . 129
4.3.6 BIND 主文件扩展：$GENERATE 指令 . . . 130
4.3.7 附加文件格式 . . . 132
第 4.4 节 BIND 9 统计 . . . 132
4.4.1 统计文件 . . . 133
4.4.2 统计计数器 . . . 134

第五章 高级 DNS 特征 139
第 5.1 节 通知（Notify） . . . 139
第 5.2 节 动态更新（Dynamic Update） . . . 139
5.2.1 日志文件 . . . 140
第 5.3 节 增量区传送（IXFR） . . . 140
第 5.4 节 分割 DNS . . . 141
5.4.1 分割 DNS 设置的例子 . . . 141
第 5.5 节 TSIG . . . 145
5.5.1 生成一个共享密钥 . . . 146
5.5.2 装载一个新密钥 . . . 146
5.5.3 指示服务器使用一个密钥 . . . 147
5.5.4 基于 TSIG 的访问控制 . . . 147
5.5.5 错误 . . . 147
第 5.6 节 TKEY . . . 148
第 5.7 节 SIG(0) . . . 148
第 5.8 节 DNSSEC . . . 149
5.8.1 生成密钥 . . . 149
5.8.2 对区签名 . . . 150
5.8.3 为 DNSSEC 配置服务器 . . . 150
第 5.9 节 DNSSEC，动态区，和自动化签名 . . . 152
5.9.1 从不安全转换到安全 . . . 152
5.9.2 动态 DNS 更新方法 . . . 153
5.9.3 完全自动化区签名 . . . 154
5.9.4 私有类型记录 . . . 155
5.9.5 DNSKEY 轮转 . . . 156
5.9.6 动态 DNS 更新方法 . . . 156
5.9.7 自动密钥轮转 . . . 156

5.9.8 通过 UPDATE 轮转 NSEC3PARAM . . . 156
5.9.9 从 NSEC 转换到 NSEC3 . . . 156
5.9.10 从 NSEC3 转换到 NSEC . . . 157
5.9.11 从安全转换为不安全 . . . 157
5.9.12 定期重签名 . . . 157
5.9.13 NSEC3 和 OPTOUT . . . 157
第 5.10 节 动态信任锚管理 . . . 157
5.10.1 验证解析器 . . . 158
5.10.2 权威服务器 . . . 158
第 5.11 节 PKCS#11(Cryptoki) 支持 . . . 159
5.11.1 先决条件 . . . 159
5.11.2 原生 PKCS#11 . . . 159
5.11.3 基于 OpenSSL 的 PKCS#11 . . . 160
5.11.4 PKCS#11 工具 . . . 164
5.11.5 使用 HSM . . . 165
5.11.6 在命令行指定引擎 . . . 166
5.11.7 以自动区重签的方式运行 named . . . 167
第 5.12 节 DLZ (Dynamically Loadable Zones，动态加载区) . . . 167
5.12.1 配置 DLZ . . . 168
5.12.2 样例 DLZ 驱动 . . . 169
第 5.13 节 DynDB（动态数据库） . . . 169
5.13.1 配置 DynDB . . . 170
5.13.2 样例 DynDB 模块 . . . 170
第 5.14 节 目录区 . . . 171
5.14.1 操作原理 . . . 171
5.14.2 配置目录区 . . . 172
5.14.3 目录区格式 . . . 173
第 5.15 节 BIND 9 对 IPv6 的支持 . . . 174
5.15.1 使用 AAAA 记录查找地址 . . . 175
5.15.2 使用半字节格式从地址查名字 . . . 175

第六章 BIND 9 安全考虑 177
第 6.1 节 访问控制表 . . . 177
第 6.2 节 Chroot 和 Setuid . . . 179
6.2.1 chroot 环境 . . . 180
6.2.2 使用 setuid 函数 . . . 180
第 6.3 节 动态更新的安全 . . . 180

第七章 排除故障 181
第 7.1 节 常见问题 . . . 181
7.1.1 它不工作；我如何判定哪里出错了？ . . . 181
7.1.2 EDNS 合规性问题 . . . 181
第 7.2 节 增加和修改序列号 . . . 182
第 7.3 节 从哪里获得帮助？ . . . 182

第八章 发行注记 183
第 8.1 节 介绍 . . . 185
第 8.2 节 关于版本编号的注释 . . . 185
第 8.3 节 支持的平台 . . . 185
第 8.4 节 下载 . . . 185
第 8.5 节 BIND 9.16.8 注记 . . . 186
8.5.1 新特性 . . . 186
8.5.2 特性变化 . . . 186
8.5.3 漏洞修补 . . . 186
第 8.6 节 BIND 9.16.7 注记 . . . 187
8.6.1 新特性 . . . 187
8.6.2 漏洞修补 . . . 187
第 8.7 节 BIND 9.16.6 注记 . . . 187
8.7.1 安全修补 . . . 187
8.7.2 新特性 . . . 188
8.7.3 特性变化 . . . 188
8.7.4 漏洞修补 . . . 188
第 8.8 节 BIND 9.16.5 注记 . . . 189
8.8.1 新特性 . . . 189
8.8.2 漏洞修补 . . . 189
第 8.9 节 BIND 9.16.4 注记 . . . 190
8.9.1 安全修补 . . . 190
8.9.2 新特性 . . . 190
8.9.3 特性变化 . . . 191
8.9.4 漏洞修补 . . . 191
第 8.10 节 BIND 9.16.3 注记 . . . 192
8.10.1 已知问题 . . . 192
8.10.2 特性变化 . . . 192
8.10.3 漏洞修补 . . . 193
第 8.11 节 BIND 9.16.2 注记 . . . 193

8.11.1 安全修补 . . . 193
8.11.2 已知问题 . . . 193
8.11.3 特性变化 . . . 193
8.11.4 漏洞修补 . . . 194
第 8.12 节 BIND 9.16.1 注记 . . . 194
8.12.1 已知问题 . . . 194
8.12.2 特性变化 . . . 194
8.12.3 漏洞修补 . . . 194
第 8.13 节 BIND 9.16.0 注记 . . . 195
8.13.1 新特性 . . . 195
8.13.2 特性变化 . . . 196
8.13.3 去掉的特性 . . . 197
第 8.14 节 许可证 . . . 197
第 8.15 节 生命周期结束 . . . 197
第 8.16 节 谢谢您 . . . 197

第九章 DNS 和 BIND 的简要历史 199

第十章 通用 DNS 参考信息 201
第 10.1 节 IPv6 地址（AAAA） . . . 201
第 10.2 节 参考书目（和建议读物） . . . 201
10.2.1 请求注释（RFC） . . . 201
第 10.3 节 互联网标准 . . . 202
第 10.4 节 建议标准 . . . 202
第 10.5 节 信息参考类 RFC . . . 206
第 10.6 节 试验性 RFC . . . 207
第 10.7 节 当前最佳实践 RFC . . . 208
第 10.8 节 已成历史的 RFC . . . 209
第 10.9 节 关于“未知”类型的 RFC . . . 209
第 10.10 节 已废弃并且未实现的试验性 RFC . . . 209
第 10.11 节 BIND 9 中不再支持的 RFC . . . 211
10.11.1 注释 . . . 211
10.11.2 互联网草案 . . . 212
10.11.3 其它关于 BIND 的文档 . . . 213

第十一章 手册页 215
第 11.1 节 rndc.conf - rndc 的配置文件 . . . 215
11.1.1 概要 . . . 215

11.1.2 描述 . . . 215
11.1.3 例子 . . . 216
11.1.4 名字服务器配置 . . . 217
11.1.5 参见 . . . 217
第 11.2 节 rndc - 名字服务器控制工具 . . . 217
11.2.1 概要 . . . 217
11.2.2 描述 . . . 218
11.2.3 选项 . . . 218
11.2.4 命令 . . . 219
11.2.5 限制 . . . 225
11.2.6 参见 . . . 225
第 11.3 节 nsec3hash - 生成 NSEC3 哈希 . . . 226
11.3.1 概要 . . . 226
11.3.2 描述 . . . 226
11.3.3 参数 . . . 226
11.3.4 参见 . . . 226
第 11.4 节 dnstap-read - 以人可读的格式输出 dnstap 数据 . . . 226
11.4.1 概要 . . . 226
11.4.2 描述 . . . 227
11.4.3 选项 . . . 227
11.4.4 参见 . . . 227
第 11.5 节 named-nzd2nzf - 转换一个 NZD 数据库到 NZF 文本格式 . . . 227
11.5.1 概要 . . . 227
11.5.2 描述 . . . 227
11.5.3 参数 . . . 227
11.5.4 参见 . . . 228
第 11.6 节 named-journalprint - 以人可读的格式打印区日志文件 . . . 228
11.6.1 概要 . . . 228
11.6.2 描述 . . . 228
11.6.3 参见 . . . 228
第 11.7 节 mdig - DNS 流水线查找工具 . . . 228
11.7.1 概要 . . . 228
11.7.2 描述 . . . 229
11.7.3 任意位置选项 . . . 229
11.7.4 全局选项 . . . 229
11.7.5 本地选项 . . . 231
11.7.6 参见 . . . 232

第 11.8 节 named-rrchecker - 针对单个 DNS 资源记录的语法检查器 233
11.8.1 概要 233
11.8.2 描述 233
11.8.3 参见 233
第 11.9 节 arpaname - 将 IP 地址翻译成对应的 ARPA 名字 233
11.9.1 概要 233
11.9.2 描述 233
11.9.3 参见 233
第 11.10 节 dnssec-revoke - 设置一个 DNSSEC 密钥中的 REVOKED 位 234
11.10.1 概要 234
11.10.2 描述 234
11.10.3 选项 234
11.10.4 参见 234
第 11.11 节 dnssec-cds - 基于 CDS/CDNSKEY 修改一个子区的 DS 记录 235
11.11.1 概要 235
11.11.2 描述 235
11.11.3 选项 236
11.11.4 退出状态 237
11.11.5 例子 237
11.11.6 参见 238
第 11.12 节 dnssec-keygen: DNSSEC 密钥生成工具 238
11.12.1 概要 238
11.12.2 描述 238
11.12.3 选项 238
11.12.4 定时选项 241
11.12.5 生成的密钥 241
11.12.6 例子 242
11.12.7 参见 242
第 11.13 节 dnssec-keyfromlabel - DNSSEC 密钥生成工具 242
11.13.1 概要 242
11.13.2 描述 243
11.13.3 选项 243
11.13.4 定时选项 245
11.13.5 生成的密钥文件 246
11.13.6 参见 246
第 11.14 节 dnssec-verify - DNSSEC 区验证工具 246
11.14.1 概要 246

11.14.2 描述 246
11.14.3 选项 246
11.14.4 参见 247
第 11.15 节 dnssec-settime: 为一个 DNSSEC 密钥设置密钥定时元数据 248
11.15.1 概要 248
11.15.2 描述 248
11.15.3 选项 248
11.15.4 定时选项 249
11.15.5 密钥状态选项 250
11.15.6 打印选项 250
11.15.7 参见 251
第 11.16 节 dnssec-importkey - 从外部系统导入 DNSKEY 记录从而可对其进行管理 . . 251
11.16.1 概要 251
11.16.2 描述 251
11.16.3 选项 251
11.16.4 定时选项 252
11.16.5 文件 252
11.16.6 参见 252
第 11.17 节 dnssec-signzone - DNSSEC 区签名工具 253
11.17.1 概要 253
11.17.2 描述 253
11.17.3 选项 253
11.17.4 例子 257
11.17.5 参见 258
第 11.18 节 dnssec-dsfromkey - DNSSEC DS 资源记录生成工具 258
11.18.1 概要 258
11.18.2 描述 258
11.18.3 选项 259
11.18.4 例子 259
11.18.5 文件 260
11.18.6 注意 260
11.18.7 参见 260
第 11.19 节 dnssec-checkds - DNSSEC 授权一致性检查工具 260
11.19.1 概要 260
11.19.2 描述 260
11.19.3 选项 260
11.19.4 参见 261

第 11.20 节 dnssec-coverage - 检查一个区 DNSKEY 将来的覆盖 261
11.20.1 概要 261
11.20.2 描述 261
11.20.3 选项 262
11.20.4 参见 263
第 11.21 节 dnssec-keymgr - 为一个基于一个已定义策略的区确保正确的 DNSKEY 覆盖 264
11.21.1 概要 264
11.21.2 描述 264
11.21.3 选项 264
11.21.4 策略配置 265
11.21.5 剩余工作 267
11.21.6 参见 267
第 11.22 节 filter-aaaa.so - 当 A 记录存在时从 DNS 响应中过滤 AAAA 记录 267
11.22.1 概要 267
11.22.2 描述 267
11.22.3 选项 268
11.22.4 参见 268
第 11.23 节 ddns-confgen - ddns 密钥生成工具 269
11.23.1 概要 269
11.23.2 描述 269
11.23.3 选项 269
11.23.4 参见 270
第 11.24 节 rndc-confgen - rndc 密钥生成工具 270
11.24.1 概要 270
11.24.2 描述 270
11.24.3 参数 270
11.24.4 例子 271
11.24.5 参见 271
第 11.25 节 delv - DNS 查找和验证工具 271
11.25.1 概要 271
11.25.2 描述 272
11.25.3 简单用法 272
11.25.4 选项 273
11.25.5 请求选项 274
11.25.6 文件 275
11.25.7 参见 276
第 11.26 节 nsupdate - 动态 DNS 更新工具 276

11.26.1 概要 . . . 276
11.26.2 描述 . . . 276
11.26.3 选项 . . . 277
11.26.4 输入格式 . . . 278
11.26.5 例子 . . . 280
11.26.6 文件 . . . 280
11.26.7 参见 . . . 281
11.26.8 缺陷 . . . 281
第 11.27 节 host - DNS 查找工具 . . . 281
11.27.1 概要 . . . 281
11.27.2 描述 . . . 281
11.27.3 选项 . . . 281
11.27.4 IDN 支持 . . . 283
11.27.5 文件 . . . 283
11.27.6 参见 . . . 283
第 11.28 节 dig - DNS 查找工具 . . . 283
11.28.1 概要 . . . 283
11.28.2 描述 . . . 284
11.28.3 简单用法 . . . 284
11.28.4 选项 . . . 285
11.28.5 请求选项 . . . 286
11.28.6 多个请求 . . . 291
11.28.7 IDN 支持 . . . 291
11.28.8 文件 . . . 291
11.28.9 参见 . . . 292
11.28.10 缺陷 . . . 292
第 11.29 节 nslookup - 交互式请求互联网名字服务器 . . . 292
11.29.1 概要 . . . 292
11.29.2 描述 . . . 292
11.29.3 参数 . . . 292
11.29.4 交互命令 . . . 293
11.29.5 返回值 . . . 294
11.29.6 IDN 支持 . . . 295
11.29.7 文件 . . . 295
11.29.8 参见 . . . 295
第 11.30 节 named - 互联网域名服务器 . . . 295
11.30.1 概要 . . . 295

11.30.2 描述 295
11.30.3 选项 295
11.30.4 信号 298
11.30.5 配置 298
11.30.6 文件 298
11.30.7 参见 298
第 11.31 节 pkcs11-keygen - 在一个 PKCS#11 设备上生成密钥 298
11.31.1 概要 298
11.31.2 描述 299
11.31.3 参数 299
11.31.4 参见 299
第 11.32 节 pkcs11-tokens - 列出 PKCS#11 的可用符号 300
11.32.1 概要 300
11.32.2 描述 300
11.32.3 参数 300
11.32.4 参见 300
第 11.33 节 pkcs11-list - 列出 PKCS#11 对象 300
11.33.1 描述 300
11.33.2 参数 300
11.33.3 参见 301
11.33.4 概要 301
11.33.5 描述 301
11.33.6 参数 301
11.33.7 参见 301
第 11.34 节 named-checkconf - named 配置文件语法检查工具 302
11.34.1 概要 302
11.34.2 描述 302
11.34.3 选项 302
11.34.4 返回值 303
11.34.5 参见 303
第 11.35 节 named-checkzone, named-compilezone - 区文件正确性检查和转换工具 . 303
11.35.1 概要 303
11.35.2 描述 303
11.35.3 选项 303
11.35.4 返回值 305
11.35.5 参见 306

第一章　介绍

互联网域名系统（DNS）由以下几个部份组成：在互联网中以层次体系方式指定实体名字的语法，用于在名字之间授权的规则，和实际完成从名字到互联网地址映射的系统实现。DNS 数据是维护在一组分布式层次数据库中。

第 1.1 节　文档范围

伯克利互联网名字域（Berkeley Internet Name Domain，BIND）在一些操作系统上实现了一个名字服务器。本文档为系统管理员提供了安装和维护互联网系统联盟（Internet Systems Consortium，ISC）的 BIND 版本 9 软件包的基本信息。

本手册涵盖了 BIND 版本 BIND 9.16.8（稳定版）。

第 1.2 节　本文档的组织

在本文档中，**第一章**介绍 DNS 和 BIND 的基本概念。**第二章**描述了在不同环境中运行 BIND 对资源的要求。**第三章**中的信息以 **面向任务**的方式表述并按功能组织，目标是为 BIND 9 软件的安装过程提供帮助。面向任务部份之后是 **第四章**，它是按照参考手册组织的，以帮助正在进行的软件维护。**第五章**包含更多的系统管理员实现某种选项可能会用到的高级概念。**第六章**指明安全考虑，**第七章**包含排错帮助。文档主体后面是几个 **附录**，其中包含了一些有用的参考信息，例如一个 **参考书目**以及与 BIND 和域名系统相关的历史信息。

第 1.3 节　本文档中的惯例

在本文档中，我们使用以下的排版惯例：

要描述的内容：	我们使用的风格：
一个路径，文件名，URL，主机名，邮件列表名，或者新的术语或概念	Fixed width
用户输入的文字	Fixed Width Bold
程序输出	Fixed Width

下列惯例用于描述 BIND 的配置文件：

要描述的内容：	我们使用的风格：
关键字	Fixed Width
变量	Fixed Width
可选的输入	[Text is enclosed in square brackets]

第 1.4 节　域名系统（DNS）

本文档的目的是解释 BIND （Berkeley Internet Name Domain ）软件包的安装和维护，我们通过回顾域名系统（Domain Name System，DNS）基础及其与 BIND 的关系来作为开始。

1.4.1　DNS 基础

域名系统（DNS）是一个层次化的分布式数据库。它存储用于互联网主机名与 IP 地址相互映射的信息，邮件路由信息，及其它互联网应用所用到的数据。

客户端通过调用一个 **解析器**库来在 DNS 中查找信息，解析器向一个或多个 **名字服务器**发出请求并解释响应。BIND 9 软件分发包中包括一个名字服务器，named ，和一个相关工具的集合。

1.4.2　域和域名

存储在 DNS 中的数据以 **域名**来标识，并根据类别或行政区被组织成一颗树。树中的每个节点由一个标记表示，被称为一个 **域**。节点的域名就是将从这个节点到 **根**节点的路径所经过的所有标记串接而成。它被表示成的书面格式是从右至左以点分隔的标记串。一个标记在其父域之内是唯一的。

例如，一个名叫 Example, Inc. 的公司中的一台主机的域名可以是 ourhost.example.com ，这里 com 是 ourhost.example.com 所属域的顶级域，example 是 com 下的一个子域，而 ourhost 是主机名。

为管理考虑，名字空间划分为名为 **区**的单位，每个区从一个节点开始，向下扩展到叶子节点或其它区开始的节点。每个区的数据存储在一个 **名字服务器**中，名字服务器使用 DNS **协议**回答对区的查询。

每个域名相关的数据以 **资源记录**（resource records，RR）的形式存储。一些所支持的资源记录类型在资源记录的类型及使用时机 中描述。

更多关于 DNS 设计和 DNS 协议的详细的信息，请参见请求注释（RFC） 中所列的标准文档。

1.4.3 区

正确运行一个名字服务器，理解一个 **区**和一个 **域**之间的差别是很重要的。

正如前面所说，区是 DNS 树的一个授权点。一个区是由域树中那些邻接部份组成，名字服务器具有这部份域树的全部信息，并是这部份域树的权威。它包含域树中自某个节点之下的所有域名，除那些被授权到其它区的之外。一个授权点在父区中以一个或多个 NS **记录**标记，它应该与位于被授权区顶点的等效 NS 记录相匹配。

例如，考虑 example.com 域，它可以包含这样的名字：host.aaa.example.com 和 host.bbb.example.com ，即使 example.com 区只包括 aaa.example.com 和 bbb.example.com 区的授权。区可以精确对应到一个单一的域，也可以对应到域的一部份，而其它部份则可以授权到其它名字服务器。DNS 树中的每个名字都是一个 **域**，即使它是 **终端节点**，只不过这样就没有 **子域**。每个子域都是一个域，而除了根之外的所有域同时都是一个子域。术语不太形象，我们建议你阅读 RFC 1033 ， RFC 1034 和 RFC 1035 以获得对这个艰难而微妙问题的完整理解。

虽然 BIND 被称为一个 “域名服务器”，但它主要处理的是区。在 named.conf 文件中声明的主服务器和辅服务器都是指的区，而不是域。当你询问一些别的站点是否愿意充当你的 **域**的辅服务器时，你实际上是想要对方为一些区来担当辅服务器的服务。

1.4.4 权威名字服务器

每个区至少由一台 **权威名字服务器**来服务。后者包含了这个区的全部数据。为了使 DNS 能在服务器和网络故障时照常工作，大多数区都有两个或更多的权威服务器，并且分布在不同的网络中。

权威服务器的应答数据包中包括 “权威回答”（authoritative answer, AA）位。这在使用像 dig（诊断工具 ）这样的工具来调试 DNS 配置时容易鉴别。

主服务器

维护有原始区数据的权威服务器被称为 **主服务器**，或简称 **主**。典型情况下，它从某个本地文件装载区数据，这个本地文件是由手工编辑，或者由某个手工编辑的其它本地文件所生成。这个文件叫做 **区文件**或 **主文件**。

然而，在某些情况下，主文件可能完全不是手工编辑而成，而是 **动态更新**操作的结果。

辅服务器

其它的权威服务器，**辅**服务器（slave，也被称为 secondary）通过一个名叫 **区传送**（zone transfer）的复制过程从另一台服务器中取得区的内容。典型情况下，数据直接从主服务器传送，但是也可能从另一台辅服务器传送。换句话说，一个辅服务器本身也可以充当一个二级辅服务器的主服务器。

辅服务器必须定期发出一个刷新请求来决定区内容是否需要更新。这是通过请求这个区的 SOA 记录并检查 SERIAL 字段是否被更新了来完成的；如果已更新，就发起一个新的区传送请求。这些刷新请求的时间是由 SOA 的 REFRESH 和 RETRY 字段控制的，但是可以被 max-refresh-time，min-refresh-time，max-retry-time 和 min-retry-time 选项覆盖。

如果区数据不能在 SOA 的 EXPIRE 选项（最大到硬编码的 24 周）所指定的时间内更新，辅区将会过期并且不再响应请求。

隐藏服务器

通常区的所有权威服务器都在上级区的 NS 记录中列出。这些 NS 记录组成了上级对这个区的 **授权**。权威服务器也在自身的区文件中列出，位置在区的 **顶级**（top level）或 **顶点**（apex）。你可以在区的顶级 NS 记录中列出在上级区中没有 NS 授权的服务器，但是你不能在上级区中对不在本级区顶级中出现的服务器授权。

一个 **隐藏服务器**就是指是一个区的权威服务器但却没有出现在区的 NS 记录中。隐藏服务器可以用来保存一个区的本地拷贝，以加速对区记录的访问，或者在区的所有“官方”服务器都无法访问时使区仍然可用。

一个主服务器本身是作为一个隐藏服务器配置时，通常被称为一个“隐藏主服务器”配置。这种配置的一个用途是主服务器在一个防火墙的后面而不直接与外面的世界通信。

1.4.5 缓存名字服务器

由大多数操作系统所提供的解析器库叫做 **存根解析器**，意思是它们没有通过直接与权威服务器通信而执行完整域名解析过程的能力。作为代替，它们依赖一个本地名字服务器来为它们执行解析。这个本地服务器称为 **递归的**名字服务器；它为本地客户端执行 **递归查找**。

为增强性能，递归服务器缓存它们所执行查找的结果。由于递归过程和缓存是密切相联的，术语 **递归服务器**和 **缓存服务器**通常是作为同义词使用的。

在一个缓存名字服务器的缓存中，一个记录被保留的时间长短是由与每个资源记录相关的生存期（Time To Live, TTL）字段所控制的。

转发

即使一个缓存名字服务器也可以不需要由其本身来执行递归查找。作为代替，它可以将其自身缓存中没有的一些或全部请求 **转发**到另一个缓存服务器，后者通常被称为一个 **转发服务器**。

使用转发服务器的典型情况是，当一个管理员不希望一个站点的所有服务器都与互联网上的其它服务器直接打交道时。例如，一个通常景象是当多个内部 DNS 服务器在一台互联网防火墙之后。防火墙之后的服务器将它们的请求转发到能够访问外部的服务器，后者代表内部服务器查询互联网上的 DNS 服务器。

另外一个场景（现在主要被响应策略区域取代）是先将查询发送到一台定制服务器进行 RBL 处理，然后再将它们转发到更广泛的互联网。

在给定的设置中可以有一个或多个转发服务器。在 named.conf 中所列出的转发者的顺序并不决定其被查询的顺序；相反，named 使用先前请求的响应时间来选择响应最快的服务器。对于尚未被查询的服务器，将给出一个初始的小随机响应时间，以确保其至少被尝试一次。根据所记录的响应时间进行的动态调整确保所有的转发者都被请求，即使其具有更慢的响应时间。这允许基于服务器的响应特性而改变行为（译注：分配的请求）。

1.4.6 名字服务器的多个角色

BIND 名字服务器可以同时作为一些区的主服务器，另一些区的辅服务器以及为一些本地客户端充当缓存（递归）服务器。

然而，由于权威名字服务和缓存/递归名字服务的功能在逻辑上是分离的，通常将它们分别运行在分离的服务器上更有利些。一个只提供权威名字服务的服务器（一个 **只权威**服务器）可以关掉递归功能运行，这样增强了可靠性和安全性。一个不为任何区作权威服务器并且只为本地客户端提供递归服务的服务器（一个 **只缓存**服务器）不需要全面开放给互联网，可以被放在一个防火墙内部。

第二章　BIND 资源要求

第 2.1 节　硬件要求

传统上，DNS 对硬件的要求十分适度。对于许多配置，从日常服务中淘汰下来的服务器就可以极好地胜任 DNS 服务器。

然而，BIND 9 的 DNSSEC 特性需要更多的 CPU 能力，所以更多使用这些特性的机构可能需要为其应用考虑更大的系统。BIND 9 是完全支持多线程的，可以完整地利用为其需要而配置的多个处理器。

第 2.2 节　CPU 要求

BIND 9 的 CPU 要求范围可以从服务于几个静态区并且没有缓存服务的 i386 级机器变化到企业级的机器，需要处理多个动态更新且有 DNSSEC 签名的区，并且可以达到每秒数千个请求。

第 2.3 节　内存要求

服务器的内存必须足够大，以适合缓存和将区数据从硬盘装载到内存。max-cache-size 选项可以用于限制缓存所使用的内存数量，代价是减少缓存的命中率并带来更大的 DNS 流量。保持足够的内存以装载所有的区并在内存中缓存数据仍然是最好的实践；遗憾的是，对给定配置来决定这个的最好方法是在运行中观察名字服务器。经过几周的运行，服务器进程能够达到一个相对稳定的大小，这时，缓存中因过期而被丢掉的条目与进入缓存的条目的速度一样。

第 2.4 节　字服务器增强环境问题

对于名字服务器增强的环境，用到两种可供选择的配置。第一种是客户端和所有的二级内部名字服务器请求一个主名字服务器，后者有足够的内存来建立一个巨大的缓存。这个方法最大限度减少了

外部名字查找所用到的带宽。第二种是设置内部的二级服务器来各自独立进行查询。在这个配置中，不需要某台主机有像第一种方法那样需要巨大的内存和 CPU 能力，但是这个方法的缺点是需要更多的外部查询，因为所有的名字服务器都不共享它们缓存的数据。

第 2.5 节　支持的操作系统

ISC BIND 9 可以在大多数类 UNIX 操作系统和微软 Windows Server 2012 R2，2016 和 Windows 10 上编译和运行。查看所支持系统的最新名单，参见 BIND 9 源码分发包的顶级目录中的 PLATFORMS.md 文件。

第三章　名字服务器配置

在本章，我们提供一些建议的配置和与之相关的准则。我们建议给某些选项设置合理的值。

第 3.1 节　样例配置

3.1.1　一个只缓存名字服务器

下列配置样例适合用于一个只缓存名字服务器，由一个公司的内部客户端所使用。通过使用 allow-query 选项，所有外来客户端的请求都被拒绝。另外一个选择是，使用合适的防火墙规则也可以取得同样的效果。

```
// Two corporate subnets we wish to allow queries from.
acl corpnets { 192.168.4.0/24; 192.168.7.0/24; };
options {
    // Working directory
    directory "/etc/namedb";

    allow-query { corpnets; };
};
// Provide a reverse mapping for the loopback
// address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
    type master;
    file "localhost.rev";
    notify no;
};
```

3.1.2 一个只权威名字服务器

这个样例配置是给只权威服务器的，在这里服务器作 example.com 的主服务器和其子域 eng.exmaple.com 的辅服务器。

```
options {
    // Working directory
    directory "/etc/namedb";
    // Do not allow access to cache
    allow-query-cache { none; };
    // This is the default
    allow-query { any; };
    // Do not provide recursive service
    recursion no;
};

// Provide a reverse mapping for the loopback
// address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
    type master;
    file "localhost.rev";
    notify no;
};
// We are the master server for example.com
zone "example.com" {
    type master;
    file "example.com.db";
    // IP addresses of slave servers allowed to
    // transfer example.com
    allow-transfer {
     192.168.4.14;
     192.168.5.53;
    };
};
// We are a slave server for eng.example.com
zone "eng.example.com" {
    type slave;
    file "eng.example.com.bk";
    // IP address of eng.example.com master server
    masters { 192.168.4.12; };
};
```

第 3.2 节　负载均衡

使用 DNS 对一个名字配置多个记录（例如多个 A 记录）可以完成原始形式的负载均衡。

例如，如果你有 3 台 HTTP 服务器分别使用 10.0.0.1，10.0.0.2 和 10.0.0.3 的网络地址，像以下这个记录集就意味着客户端将会分别对每台机器有三分之一的连接时间：

Name	TTL	CLASS	TYPE	Resource Record (RR) Data
www	600	IN	A	10.0.0.1
	600	IN	A	10.0.0.2
	600	IN	A	10.0.0.3

当一个解析器请求这些记录时，BIND 将滚动这三个记录，以一个不同的顺序响应请求。在上面的例子中，不同的客户端将会随机收到以 1，2，3；2，3，1 和 3，1，2 的顺序的记录。大多数客户端将使用所得到的第一个记录而丢弃其余的。

关于将响应排序的更详细的内容，检查 options 语句的 rrset-order 子语句，参见资源记录集排序。

第 3.3 节　名字服务器操作

3.3.1　用于名字服务器后台进程的工具

本节描述了几个绝对必要的诊断，管理和监控工具，它们是提供给系统管理员进行控制和调试名字服务器的。

诊断工具

dig，host 和 nslookup 程序都是用于手工向服务器发请求的命令行工具。他们在输入风格和输出格式上不相同。

dig dig 是这些查询工具中最多功能的。它有两种模式：简单交互模式用于单个请求，批处理模式对有多个请求的名单中的每一个执行请求。所有选项都可以从命令行访问。

dig [@server] domain [query-type][query-class][+query-option][-dig-option][%comment]

经常使用的 dig 的简单形式是

dig @server domain query-type query-class

关于更多的可用命令和选项的清单，参见 dig 的手册页。

host host 工具强调简单和易用。缺省情况下，它在主机名和互联网地址之间互相转换，但它的功能也可使用选项扩展。

host [-aCdlnrsTwv][-c class][-N ndots][-t type][-W timeout][-R retries][-m flag][-4][-6] hostname [server]

关于更多的可用命令和选项的清单，参见 host 的手册页。

nslookup nslookup 有两种模式：交互式和非交互式。交互模式允许使用者向名字服务器发出对各种主机和域名信息的查询请求，或者打印出一个域下面的主机列表。非交互模式只能用于打印所查询的某个主机或域的名字和请求信息。

nslookup [-option][[host-to-find]|[-[server]]]

在没有给出参数（使用缺省的名字服务器）或者第一个参数是连字符（- ）并且第二个参数是主机名或者是一台名字服务器的 IP 地址时，就进入了交互模式。

当需要查找的名字或者主机的 IP 地址作为第一个参数给出时，就使用非交互模式。可选的第二个参数指定主机名或者一台名字服务器的地址。

由于其神秘的用户界面和频繁的不一致表现，我们不推荐使用 nslookup 。而使用 dig 替代。

管理工具

管理工具在一个服务器的管理中扮演一个不可或缺的角色。

named-checkconf named-checkconf 程序用来检查一个 named.conf 文件的语法。

named-checkconf [-jvz][-t directory][filename]

named-checkzone named-checkzone 程序用来检查一个主文件的语法和一致性。

named-checkzone [-djqvD][-c class][-o output][-t directory][-w directory][-k (ignore|warn|fail)][-n (ignore|warn|fail)][-W (ignore|warn)] zone [filename]

named-compilezone 这个工具与 named-checkzone 相似，但它总是将区的内容转储到一个指定的文件（通常是一个与区文件不同的格式）。

rndc 远程名字服务控制（remote name daemon control，rndc ）程序允许系统管理员控制一个名字服务器的运行。如果你不带任何参数运行 rndc ，它将显示出以下的用法信息：

rndc [-c config][-s server][-p port][-y key] command [command...]

关于可用的 rndc 命令细节，参见rndc - 名字服务器控制工具 。

rndc 需要一个配置文件，由于所有与服务器的通信都使用依赖共享密钥的数字签名来认证，并且没有其它方式可以比配置文件提供更好的保密方式。rndc 配置文件的缺省路径是 /etc/rndc.conf ，但也可以使用 -c 选项来指定一个其它的路径。如果 rndc 没有找到配置文件，它将会查找 /etc/rndc.key （或者是 BIND 构建时由 sysconfdir 所定义的其它目录）。rndc.key 文件是由 rndc-confgen -a 所生成，如在controls 语句定义和用法 中所描述。

配置文件的格式类似于 named.conf 的格式，但是只有四个语句，options ，key ，server 和 include 语句。这些语句都是与密钥相关的，服务器使用这些密钥共享密钥。语句的顺序没有关系。

options 语句有三个子句：default-server ，default-key 和 default-port 。default-server 需要一个主机名或 IP 地址参数，它表示一个要通信的缺省的服务器，如果服务器未在命令行中以 -s 选项指定的情况。default-key 以一个密钥的名字作为其参数，密钥是在 key 语句中定义的。default-port 指定 rndc 用到的缺省端口，它在命令行或 server 语句中没有指定端口的情况下生效。

key 语句定义 rndc 同 named 进行认证时要用到的密钥。其语法与 named.conf 中的 key 语句相同。key 关键字后跟一个密钥名，它可以是一个有效的域名，尽管它并不需要一个层次结构；因而，一个像 “rndc_key ” 这样的字符串也是一个有效的名字。key 语句有两个子句：algorithm 和 secret 。配置分析器将接受任何字符串作为算法参数，当前只有字符串 hmac-md5 , hmac-sha1 , hmac-sha224 , hmac-sha256 , hmac-sha384 和 hmac-sha512 有意义。这个密钥是一个在 RFC 3548 中所指定的 Base64 编码的字符串。

server 语句将一个由 key 语句定义的密钥与一台服务器结合起来。关键字 server 后跟一个主机名或 IP 地址。server 语句有两个子句：key 和 port 。key 子句指定用于同这个服务器通信的密钥，port 子句用于指定 rndc 用来连接到这个服务器所用的端口。

以下是一个最小配置文件的样例：

```
key rndc_key {
   algorithm "hmac-sha256";
   secret
    "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K";
};
options {
   default-server 127.0.0.1;
   default-key   rndc_key;
};
```

这个文件，如果是作为 /etc/rndc.conf 安装，允许以下命令：

```
$ rndc reload
```

经过 953 端口连接到 127.0.0.1 并使名字服务器重新装载，如果一个运行在本机的名字服务器使用了以下的控制语句：

```
controls {
  inet 127.0.0.1
    allow { localhost; } keys { rndc_key; };
};
```

并且它有一个对应于 rndc_key 的 key 语句的话。

运行 rndc-confgen 程序将会方便地创建一个 rndc.conf 文件，并且会显示所需要添加到 named.conf 中的相关的 controls 语句。另外一个选择是，可以运行 rndc-confgen -a 来建立一个 rndc.key 文件，就一点也不用修改 named.conf 了。

3.3.2 信号

某些 UNIX 信号将使名字服务器执行特定的动作，如下表所列。这些信号可以由 kill 命令发出。

SIGHUP	使服务器读 named.conf 并重新装载数据库。
SIGTERM	使服务器清理并退出。
SIGINT	使服务器清理并退出。

第 3.4 节 插件

插件是一个实验动态加载库扩展 named 功能的机制。通过使用插件，对多数用户，服务器的核心功能可以保持简单；更复杂的代码实现可选的特性，只被那些需要这些特性的用户安装。

插件接口是一项正在进行中的工作，预期进化成添加更多的插件。当前，仅支持 “请求插件”；这些改变了名字服务器的请求逻辑。未来可能增加其它的插件类型。

当前唯一包含进 BIND 的插件是 filter-aaaa.so ，它替代了先前已存在的 named 自然组成部份的 filter-aaaa 特性。这个特性的代码已经从 named 中去掉了，并且不再能使用标准的 named.conf 语法配置，但是链接 filter-aaaa.so 插件提供同样的功能。

3.4.1 配置插件

插件通过在 named.conf 中使用 plugin 语句配置：

```
plugin query "library.so" {
  parameters
};
```

在这个例子中，文件 library.so 是插件库。query 指示这是一个请求插件。

可以指定多个 plugin 语句，以加载不同的插件或者同一个插件的多个实例。

parameters 会作为一个不透明的字符串传送给插件的初始化程序。配置语法依不同的模块而有差异。

3.4.2 开发插件

每个插件实现了四个功能：

- plugin_register 申请内存，配置一个插件实例，并绑定到 named 内的钩子点，
- plugin_destroy 拆卸插件实例并释放内存，
- plugin_version 检查插件是否兼容于当前版本的插件 API，
- plugin_check 测试插件参数的语法正确性。

在 named 源代码的不同位置，有一些“钩子点”，插件在此注册自身。在 named 运行时，当运行到钩子点，会检查是否有任何插件在此注册了它们自身；如果注册过，就调用相应的“钩子动作”- 这是插件库中的一个功能 - 例如，修改发送回一个客户端的响应或者强制终止一个请求。更多细节可以在文件 lib/ns/include/ns/hooks.h 中找到。

第四章　BIND 9 配置参考

第 4.1 节　配置文件元素

以下是贯穿在 BIND 配置文件的文档中的元素清单：

acl_name　由 acl 语句所定义的一个 address_match_list 的名字。

address_match_list　一个或多个 ip_addr，ip_prefix，key_id 或 acl_name 元素的列表，参见地址匹配表 。

masters_list　一个或多个带有 key_id 和/或 ip_port 选项的 ip_addr 的命名列表。一个 masters_list 可以包含其它 masters_list。

domain_name　用作一个 DNS 名字的引号内字符串，如 “my.test.domain”。

namelist　一个或多个 domain_name 的列表。

dotted_decimal　以点(‘.’)分隔的一个到四个 0 到 255 之间的整数，如 123，45.67 或 89.123.45.67。

ip4_addr　一个 IPv4 地址，严格含有 4 个元素的 dotted_decimal 符号。

ip6_addr　一个 IPv6 地址，如 2001:db8::1234。IPv6 范围地址在其范围区间具有模糊性，必须由一个带有百分号（‘%’）作分隔符的合适的区 ID 来澄清。强烈推荐使用字符串区名字而不是数字标识符，以在系统配置更改时更健壮。然而，由于没有映射这种名字和标识符值的标准，当前仅仅支持将接口名作为连接标识符。例如，一个连接的本地的地址 fe80::1 连接到接口 ne0 可以指定为 fe80::1%ne0 。注意在大多数系统的连接到本地的地址都有模糊性，需要澄清。

ip_addr　一个 ip4_addr 或者 ip6_addr。

ip_dscp　一个介于 0 到 63 之间的 number，用于给支持差分服务代码点（DSCP）的操作系统的出向流量选择一个 DSCP 值。

ip_port　一个 IP 端口 number。这个 number 范围为 0 到 65535，1024 以下端口被限制为以 root 身份运行的进程所使用。在某些情况下星号（*）字符用作占位符，以选择大数目的端口。

ip_prefix 一个 IP 网络地址，由一个 ip_addr 后跟一个斜线（‘/’）和一个代表掩码位数的数字所指定。ip_addr 后面的零将被忽略。例如，127/8 表示网络 127.0.0.0，掩码为 255.0.0.0，1.2.3.0/28 表示网络 1.2.3.0，掩码为 255.255.255.240。当指定的一个前缀涉及一个 IPv6 范围地址，这个范围可以被忽略，在这个情况，前缀将会匹配来自任何范围的包。

key_id 一个 domain_name，代表一个共享密钥的名字，用在事务安全中。

key_list 一个或多个 key_id 的列表，以分号分隔和结束。

number 一个非负 32 位整数（即 0 到 4294967295（含）之间的整数）。它所能接受的值可能更多地受其使用的上下文所限制。

fixedpoint 一个非负实数，可以被指定为精确到百分之一。小数点前最大五位数，小数点后最大两位，即最大值为 99999.99。可接受的值受到其使用上下文中的更多限制。

path_name 一个引用的字符串，用作路径名，例如 zones/master/my.test.domain。

port_list 一个 ip_port 或者端口范围的列表。一个端口范围以 range 后跟两个 ip_port 的形式指定，即 port_low 和 port_high，表示从 port_low 到 port_high （含）的端口号。port_low 必须小于或等于 port_high。例如，range 1024 65535 表示从 1024 到 65535 的端口。在两种情况下，星号（‘*’）字符都不是被允许的有效 ip_port。

size_spec 一个 64 位无符号整数、关键词 unlimited 或 default。整数可以取值的范围为 0 <= value <= 18446744073709551615，虽然某些参数（如 max-journal-size ）在这个范围中使用更受限的区间。在大多数情况下，设置一个值为 0 不意谓着字面上的零；它表示 “未定义” 或 “尽可能的大”，具体是什么取决于上下文。参见特殊参数的解释以获取 size_spec 在如何解释其用法的详细信息。数字值后面可以选择跟比例因数：K 或 k 表示千字节，M 或 m 表示兆字节，G 或 g 表示吉字节，其比例分别为乘 1024，1024*1024 和 1024*1024*1024。 ‘ unlimited 通常表示 “尽可能的大”，并且通常是设置一个大数时的最佳方式。default 使用服务器启动时的有效限制。

size_or_percent size_spec 或者整数值后跟表示百分比的’%’。这个特性与 “size_spec “完全相同，但是 “size_or_percent “也允许指定一个正整数后跟一个表示百分比的’%’符号。

yes_or_no yes 或 no。单词 true 和 false 也可以，同样还有数字 1 和 0。

dialup_option yes，no，notify，notify-passive，refresh 或 passive 之一。当使用在一个区中时，notify-passive，refresh 和 passive 被限制在只能用于辅区和存根区中。

4.1.1 地址匹配表

语法

```
address_match_list = address_match_list_element ; ...

address_match_list_element = [ ! ] ( ip_address | ip_prefix |
   key key_id | acl_name | { address_match_list } )
```

定义和用法

地址匹配表主要用于决定对各种服务器操作的访问控制。它们通常也用在 listen-on 和 sortlist 语句中。组成一个地址匹配表的元素可以是以下的任何一种：

- 一个 IP 地址（IPv4 或 IPv6）
- 一个 IP 前缀（以‘/’表示法）
- 一个密钥 ID，由 key 语句所定义的
- 使用 acl 语句定义的地址匹配表的名字
- 包含在花括号中的嵌套的地址匹配表

元素可以使用惊叹号（!）取反，匹配表名“any”、“none”、“localhost”和“localnets”是预定义的。可以从 acl 语句的描述中找到这些名字的更多信息。

增加 key 子句使这个句法元素的名字有些怪异，因为安全密钥可以校验访问而不需考虑一个主机或网络的地址。虽然如此，“地址匹配表”这个术语仍然贯穿整个文档。

当一个给定的 IP 地址或前缀与一个地址匹配表进行比较时，比较需要大致 O(1) 的时间。然而，密钥比较要求遍历密钥链表，直到找到一个匹配的密钥，这样就会较慢一些。

一次匹配的解释依赖于列表是否被使用于访问控制、定义 listen-on 端口，或者在一个 sortlist 中，以及元素是否被取反。

当用作一个访问控制表时，非否定的匹配允许访问而否定的匹配拒绝访问。如果没有匹配，则拒绝访问。子句 allow-notify，allow-recursion，allow-recursion-on，allow-query，allow-query-on，allow-query-cache，allow-query-cache-on，allow-transfer，allow-update，allow-update-forwarding，blackhole 和 keep-response-order 都使用地址匹配表。类似地，listen-on 选项将使服务器拒绝任何发到不与列表匹配的机器地址的请求。

插入的顺序是重要的。如果一个 ACL 中有超过一个元素与给定的 IP 地址或前缀匹配，那么在 ACL 中 **首先**定义的就会优先。由于这个首次匹配的特性，作为列表中某个元素的子集的元素应该放在列表的前面，而不考虑是否是否定的条目。例如，在 1.2.3/24; ! 1.2.3.13; 中，元素 1.2.3.13 完全是无用的，因为算法会将所有对 1.2.3.13 的查找匹配到元素 1.2.3/24 上。使用 ! 1.2.3.13; 1.2.3/24 修正了这个问题，通过否定符阻止了 1.2.3.13，而让其它的 1.2.3.* 主机都通过。

4.1.2 注释语法

BIND 9 注释语法允许注释可以出现在一个 BIND 配置文件中空白字符可以出现的任何位置。应各种类型的程序员的要求，注释可以写成 C，C++ 或 shell/perl 的风格。

语法

```
/* This is a BIND comment as in C */
```

```
// This is a BIND comment as in C++
```

```
# This is a BIND comment as in common Unix shells
# and perl
```

定义和用法

在一个 BIND 配置文件中，注释可以出现在任何空白字符可以出现的地方。

C 风格的注释以两个字符 /*（斜线，星号）开始，以 */（星号，斜线）结束。因为其完全以这些字符划界，所以它们可以用于在一行的一部份或跨越多行时的注释。

C 风格的注释不能嵌套。例如，以下是无效的，因为全部注释在第一个 */时结束：

```
/* This is the start of a comment.
   This is still part of the comment.
/* This is an incorrect attempt at nesting a comment. */
   This is no longer in any comment. */
```

C++ 风格的注释以两个字符 //（斜线，斜线）开始并持续到一行的结束。它们不能继续并跨越多个行；要想使一个注释跨越多行，必须在每行都使用 //。例如：

```
// This is the start of a comment.  The next line
// is a new comment, even though it is logically
// part of the previous comment.
```

Shell 风格（或者称为 perl 风格，如果你愿意）的注释以字符 #（井号）开始并持续到一行的结束，与 C++ 注释一样。例如：

```
# This is the start of a comment.  The next line
# is a new comment, even though it is logically
# part of the previous comment.
```

警告： 不能使用分号（;）字符来开始一个注释，这与在一个区文件中不同。分号表示一个配置语句的结束。

第 4.2 节　配置文件语法

BIND 9 的配置文件由语句和注释组成。语句以分号结束。语句和注释是仅有的可以出现在花括号之外的元素。许多语句包含由子语句组成的块，子语句也以分号结束。

以下是所支持的语句：

acl 定义一个命名的 IP 地址匹配列表，用于访问控制或其它用途。

controls 声明控制通道，用于 rndc 应用程序。

dnssec-policy 为区描述一个 DNSSEC 密钥和签名策略。参见dnssec-policy Grammar 获取详细信息。

include 包含一个文件。

key 指定在使用 TSIG 时，用于认证和授权的密钥信息。

logging 指定服务器记录哪些日志，和在哪里记录日志消息。

masters 定义一个命名的主服务器列表，一般包含在存根区和辅区的 masters 或 also-notify 列表中。

options 控制全局服务器配置和为其它语句设置缺省参数。

server 为基于单个服务器的配置设置某个配置选项。

statistics-channels 声明通信通道，以访问 named 统计信息。

trust-anchors 定义 DNSSEC 信任锚：如果使用 initial-key 或 initial-ds 关键字定义，信任锚将通过使用 RFC 5011 信任锚维护保持更新；如果使用 static-key 或 static-ds，密钥将不变。

managed-keys 与 trust-anchors 相同；这个选项已被废弃，倾向使用 trust-anchors 并带 initial-key 关键字，在未来的版本可能被去掉。

trusted-keys 定义永久受信任的 DNSSEC 密钥；这个选项已被废弃，倾向使用 trust-anchors 并带 static-key 关键字，在未来的版本可能被去掉。

view 定义一个视图。

zone 定义一个区。

logging 和 options 语句在每个配置文件中只能出现一次。

4.2.1 acl 语句语法

```
acl <string> { <address_match_element>; ... };
```

4.2.2 acl 语句定义和用法

acl 语句将一个地址匹配列表赋值给一个符号名字。其名字来源于地址匹配列表的主要用途：访问控制表（Access Control Lists，ACLs）。

下列 ACL 是内建的：

any 匹配所有主机。

none 匹配空主机。

localhost 匹配系统的所有网络接口的 IPv4 和 IPv6 地址。当有地址被添加或删除时，localhost ACL 元素被更新以反映变化。

localnets 匹配一个系统所在的 IPv4 或 IPv6 网络上的所有主机。当有地址被添加或删除时，localnets ACL 元素被更新以反映变化。一些系统不提供决定本地 IPv6 地址的前缀长度的方法。在这样的情况下，localnets 只匹配本地 IPv6 地址，如同 localhost 一样。

4.2.3 controls 语句语法

```
controls {
    inet ( <ipv4_address> | <ipv6_address> |
      * ) [ port ( <integer> | * ) ] allow
      { <address_match_element>; ... } [
      keys { <string>; ... } ] [ read-only
      <boolean> ];
    unix <quoted_string> perm <integer>
      owner <integer> group <integer> [
      keys { <string>; ... } ] [ read-only
      <boolean> ];
};
```

4.2.4 controls 语句定义和用法

controls 语句声明控制通道，其被系统管理员用来控制对名字服务器的操作。这些控制通道由 rndc 应用程序使用，它可以发送命令到名字服务器及从名字服务器获取一些非 DNS 的结果。

一个 inet 控制通道是一个监听在 ip_addr 地址的 ip_port 端口的 TCP 套接字，可以是 IPv4 或 IPv6 地址。一个值为 * （星号）的 ip_addr 被解释成 IPv4 通配地址；到系统的任何 IPv4 地址的连接都将被接受。要监听 IPv6 的通配地址，使用值为 :: 的 ip_addr 。如果 rndc 只用于本机，使用环回地址（127.0.0.1 或 ::1 ）是最为安全的推荐做法。

如果没有指定端口，就使用 953 端口。星号“* ”不能用于 ip_port 。

通过控制通道发送命令的能力是被 allow 和 keys 子句所限制的。通过基于 address_match_list 来允许到控制通道的连接。这只是一个简单的基于 IP 地址的过滤；address_match_list 中的任何 key_id 元素都被忽略。

一个 unix 控制通道是一个 UNIX 域套接字，它监听文件系统中指定的路径。对这个套接字的访问由 perm 、owner 和 group 子句指定。注意在某些平台（SunOS 和 Solaris）上，权限（perm ）是应用在上级目录，而在套接字本身上的权限是被忽略的。

命令通道的主要授权机制是 key_list ，它是一个 key_id 的列表。key_list 中的每个 key_id 都被授权能够通过控制通道执行命令。关于在 rndc 中配置密钥的信息，参见管理工具 中的信息。

如果 read-only 子句开启，控制通道被限制在下列只读命令集中：nta -dump ， null ， status ， showzone ， testgen 和 zonestatus 。缺省地，read-only 未开启，控制通道允许读写访问。

如果没有提供 controls 语句，named 将设置一个缺省的控制通道，监听在环回地址 127.0.0.1 及其

IPv6 的对应者::1 上。在这样的情况下，当有 controls 语句但是没有提供一个 keys 子句时，named 将试图从 /etc（或者是 BIND 编译时 sysconfdir 所指定的目录）下的 rndc.key 文件中装载命令通道密钥。要建立一个 rndc.key 文件，运行 rndc-confgen -a 。

要禁用命令通道，使用一个空的 controls 语句：controls { }; 。

4.2.5 include 语句语法

```
include filename;
```

4.2.6 include 语句定义和用法

include 语句将指定的文件（或多个文件，如果检测到一个有效的通配表达式）插入到 include 语句出现的点。include 语句使配置文件的管理更加容易，可以允许读或者写某些内容，没有其它用途。例如，这个语句可以包含进只有名字服务器才可以读的私钥。

4.2.7 key 语句语法

```
key <string> {
    algorithm <string>;
    secret <string>;
};
```

4.2.8 key 语句定义和用法

key 语句定义了一个共享密钥，用于 TSIG（参见TSIG ）或命令通道（参见controls 语句定义和用法 ）中。

key 语句可以放在配置文件的顶级或一个 view 语句的内部。定义在顶级 key 语句中的密钥可以用于所有视图中。想要用于 controls 语句（参见controls 语句定义和用法 ）中的密钥必须定义在顶级中。

key_id ，即密钥的名字，是一个唯一表示一个密钥的域名。它可以用于一个 server 语句中，它使发向那台服务器的请求都用这个密钥签名，或者用于地址匹配表中，以检验所收到的请求是由与这个名字、算法和秘密相匹配的密钥所签名。

algorithm_id 是一个指定安全/认证算法的字符串。named 服务器支持 hmac-md5，hmac-sha1，hmac-sha224，hmac-sha256，hmac-sha384 和 hmac-sha512 TSIG 认证。通过在尾部增加一个以减号开始的符合要求位数的最小数字来支持截断散列，如：hmac-sha1-80 。secret_string 是算法所用到的秘密，被当作一个 Base64 编码的字符串。

4.2.9 logging 语句语法

```
logging {
    category <string> { <string>; ... };
    channel <string> {
        buffered <boolean>;
        file <quoted_string> [ versions ( unlimited | <integer> ) ]
          [ size <size> ] [ suffix ( increment | timestamp ) ];
        null;
        print-category <boolean>;
        print-severity <boolean>;
        print-time ( iso8601 | iso8601-utc | local | <boolean> );
        severity <log_severity>;
        stderr;
        syslog [ <syslog_facility> ];
    };
};
```

4.2.10 logging 语句定义和用法

logging 语句为名字服务器配置了广泛的日志选项种类。其 channel 短语将输出方法、格式选项和严重级别与一个可以用于 category 短语中的名字联系起来，用以选择多种类别的消息应当如何记入日志。

只能使用一条 logging 语句，它可以定义多个想要的通道和类别。如果没有 logging 语句，日志配置将会是：

```
logging {
    category default { default_syslog; default_debug; };
    category unmatched { null; };
};
```

如果 named 启动时带有 -L 选项，它将日志记录到启动时指定的文件，而不是使用系统日志。在这个情况，日志配置将会是：

```
logging {
   category default { default_logfile; default_debug; };
   category unmatched { null; };
};
```

日志配置只是在全部的配置文件被分析完成之后才建立。在服务器启动时，所有关于配置文件中语法错误的日志消息都被写到缺省通道，或者在指定 -g 选项时被写到标准错误中。

channel 短语

所有的日志输出都写到一个或多个 channels ； 对能够建立的通道数量没有限制。

每个通道定义必须包含一个目标子句，这个子句中说明此通道的消息是到一个文件、到特定的 syslog 设施、到标准错误输出流还是被丢弃。作为可选项，定义还可以限定此通道所接受的消息严重级别（缺省为 info ），以及是否包含 named 所生成的时间戳，类别名字和/或严重级别（缺省是不包含任何）。

null 目标子句使得所有发送到此通道的消息被丢弃；在此情况下，通道的其它选项都没有意义。

file 目标子句将通道定向到一个磁盘文件。它可以包含附加的参数指定文件在轮转到一个备份文件之前允许增长到多大（size ），指定在每次轮转时可以保存多少个文件的备份版本（versions ），以及用于命名备份版本的格式（suffix ）。

size 选项用于限制日志文件的增长。如果文件超过了指定的大小，named 将会停止写到文件，除非它有一个 versions 选项。如果保持备份版本，文件将按照下面描述的方式轮转。如果没有 versions 选项，就不会写入更多的数据到日志中，除非有某些带外的机制删出日志或者截断日志使其小于最大大小。缺省行为是不限制文件大小。

文件轮转仅发生在文件超过 size 选项指定的大小时。缺省不保有备份版本；任何存在的日志文件都是简单地添加。versions 选项指定文件应该保持多少备份版本。如果设置为 unlimited ，就没有限制。

suffix 选项可以被设置为 increment 或者 timestamp 。如果被设置为 timestamp ，当一个日志文件轮转时，它被保存为以当前时间戳作为文件的后缀。如果被设置为 increment ， 备份文件被保存为以增加的数字作为后缀；轮转时，更旧的文件都被更名。例如，如果 versions 被设置为 3 并且 suffix 被设置为 increment ，那么当 filename.log 达到 size 所指定的大小时，filename.log.1 被更名为 filename.log.2 ， filename.log.0 被更名为 filename.log.1 ， 而 filename.log 被更名为 filename.log.0 ， 同时一个新的 filename.log 被打开。

size ， versions 和 suffix 选项用法例子：

```
channel an_example_channel {
  file "example.log" versions 3 size 20m suffix increment;
  print-time yes;
  print-category yes;
};
```

syslog 目标子句将通道导向系统日志。它的参数是在 syslog 手册页中所描述的系统日志的设施。知名的设施有 kern，user，mail，daemon，auth，syslog，lpr，news，uucp，cron，authpriv，ftp，local0，local1，local2，local3，local4，local5，local6 和 local7，但是，不是所有的操作系统上都支持所有的设施。syslog 如何处理发向这些设施的消息在 syslog.conf 手册页中描述。在一个使用非常旧的 syslog 版本的系统上，只支持使用两个参数调用 openlog() 函数，这时这个子句会被静默地忽略。

在 Windows 机器上，syslog 消息直接定向到事件查看器（EventViewer）。

severity 子句工作起来像 syslog 的“优先级”，只有一点不同，就是如果直接写到文件而不是使用 syslog 时，也可以使用它们。至少达到所使用的严重级别的消息将才会送到此通道；大于严重级别的消息会被接受。

如果使用 syslog，syslog.conf 优先级也会决定最终的通过量。例如，定义一个通道设施，其严重级别为 daemon 和 debug，但是通过 syslog.conf 设置只记录 daemon.warning，将会导致严重级别为 info 和 notice 的消息被扔掉。在相反的情况下，named 只记录 warning 或更高级别的消息，syslogd 会记录所有它从这个通道收到的消息。

stderr 目标子句将通道引导到服务器的标准错误流。其意图是在服务器作为一个前台进程运行时使用的，例如调试一个配置时。

服务器在调试模式时可以支持扩展调试信息。如果服务器的全局调试级别大于 0，调试模式将会被激活。全局调试级别可以通过在启动 named 服务时带 -d 标志并后跟一个正整数，或者通过运行 rndc trace 。全局调试级别可以设为 0，调试模式被关闭，或者通过运行 rndc notrace 。服务器中的所有调试信息都有一个调试级别，越高的调试级别给出越详细的输出。在通道内指定一个特定的调试严重级别，例如：

```
channel specific_debug_level {
  file "foo";
  severity debug 3;
};
```

在服务器工作在调试模式的任何时间时，得到第 3 级或更低的调试输出，而无论全局调试级别的设置是多少。严重级别为 dynamic 的通道使用服务器的全局调试级别来决定输出哪些消息。

print-time 可以被设置为 yes , no 或一个时间格式说明符, 它可以是 local , iso8601 或 iso8601-utc 之一。如果设置为 no ， 就不将日期和时间记入日志。如果设置为 yes 或者 local ， 日期和时间就以人可读的格式，使用本地时区记入日志。如果设置为 iso8601 ， 本地时间以 ISO8601 格式记入日志。如果设置为 iso8601-utc ， 日期和时间以 ISO8601 格式记入日志，时区设置为 UTC。缺省是 no 。

print-time 可以用于 syslog 通道，但是它通常不这样用，因为 syslog 也记录日期和时间。

如果要 print-category ， 消息的类别也将被记入日志。最后，如果打开 print-severity ， 消息的严重级别将被记入日志。print- 选项可以用在任何组合，并总是按照下列顺序打印：时间，类别，严重性。这里是所有三个 print- 选项都打开的一个例子：

28-Feb-2000 15:05:32.863 general: notice: running

如果打开了 buffered ， 到文件的输出不会在每次日志进入时都刷新。缺省是所有日志消息都被刷新。

有四个预定义的通道供 named 缺省日志使用，具体如下。如果 named 启动时带有 -L 选项，就会添加第五个通道 default_logfile 。如何使用它们在category 短语 中描述。

```
channel default_syslog {
  // send to syslog's daemon facility
  syslog daemon;
  // only send priority info and higher
  severity info;
};

channel default_debug {
  // write to named.run in the working directory
  // Note: stderr is used instead of "named.run" if
  // the server is started with the '-g' option.
  file "named.run";
  // log at the server's current debug level
  severity dynamic;
};

channel default_stderr {
  // writes to stderr
  stderr;
  // only send priority info and higher
  severity info;
};
```

(下页继续)

(续上页)

```
channel null {
  // toss anything sent to this channel
  null;
};

channel default_logfile {
  // this channel is only present if named is
  // started with the -L option, whose argument
  // provides the file name
  file "...";
  // log at the server's current debug level
  severity dynamic;
};
```

default_debug 通道具有专门的属性，它只在服务器的调试级别不为零时才有输出。通常情况下，它会写到服务器工作目录下面一个文件名为 named.run 的文件。

由于安全原因，当使用了 -u 命令行选项时，named.run 文件只在 named 改变为新的 UID 之后才被创建，并且在 named 启动过程中还以 root 运行时所产生的调试输出都被丢弃。为捕捉这些输出，在运行服务时使用 -L 选项指定一个缺省的日志文件，或者使用 -g 选项记录到标准错误，这可以重定向它到一个文件。

一旦一个通道被定义后，它不能被重定义。不能直接修改内建的通道，但是可以修改缺省的日志，即，将类别指向所定义的通道。

category 短语

存在许多类别，这样期望的日志可以发送到各处，而忽略不想要的日志。如果没有为一个类别指定一个通道名单，这个类别下面的日志消息将会被发送到 default 类别。如果没有指定一个缺省的类别，就使用下列“缺省的缺省”：

```
category default { default_syslog; default_debug; };
```

如果你启动 named 时带有 -L 选项，缺省类别是：

```
category default { default_logfile; default_debug; };
```

作为一个例子，我们假设一个用户想将安全事件记录到一个文件，但是也想保持缺省的日志行为。

他就会如下设定：

```
channel my_security_channel {
  file "my_security_file";
  severity info;
};
category security {
  my_security_channel;
  default_syslog;
  default_debug;
};
```

想丢弃一个类别中的所有消息，指定 null 通道：

```
category xfer-out { null; };
category notify { null; };
```

以下是可用类别及其所包含的日志信息类型的简单描述。将来的 BIND 发行版会添加更多的类别。

client 对客户端请求的处理。

cname 由于是一个 CNAME 记录而非一个 A/AAAA 记录而被跳过的名字服务器。

config 配置文件分析及处理。

database 与数据库相关的消息，数据库指名字服务器内部用于存储区和缓存数据。

default 为没有被明确定义的配置的类别指定的日志选项。

delegation-only 被强制成 NXDOMAIN 的请求，它是一个只授权区或一个转发、提示或存根区定义中有 delegation-only 的结果。

dispatch 分发收到的包到将要处理它们的服务器模块。

dnssec DNSSEC 和 TSIG 协议处理。

dnstap "dnstap" DNS 流量捕获系统。

edns-disabled 记录由于超时而被强制使用普通 DNS 的请求日志。这通常是因为远端服务器不是兼容 RFC 1034 （对 EDNS 请求和其不明白的 DNS 其它扩展并不总是返回 FORMERR 或类似的东西）。换句话说，这是瞄准服务器不能响应其不明白的 DNS 请求的。

注意：日志消息也可能是因为包丢失。在报告服务器不兼容 RFC 1034 之前，应该再测试它们以决定不兼容的类别。这个测试应当阻止或减少不正确报告的数目。

注意：最后，named 必须不将这样的超时归咎为不兼容 RFC 1034 而将其当作普通的包丢失。

将包丢失错误地归咎为不兼容 RFC 1034 会影响 DNSSEC 验证，后者要求对 DNSSEC 记录返回 EDNS。

general 为许多仍未被分类到某个类别中的捕捉所有。

lame-servers 远端服务器上的错误配置，是在解析过程中试图向其发请求时由 BIND 9 所发现的。

network 网络操作。

notify NOTIFY 协议。

nsid 从上游服务器收到的 NSID 选项。

queries 请求应当被记录的位置。

在启动时，指定 queries 类别也会启动对请求的日志，除非设定 querylog 选项。

请求日志条目首先以 @0x< 十六进制数字 > 格式报告一个客户端对象标识符。接下来，它报告客户端的 IP 地址和端口，请求的名字，类和类型。接下来，它报告是否设置了期望递归（Recursion Desired）的标志（如果设置是 +，如果未设置是-），请求是否被签名（S），是否使用 EDNS 并伴随 EDNS 版本号（E(#)），是否使用 TCP（T），DO 位（DNSSEC Ok）是否被设置（D），CD 位（Checking Disables）是否被设置（C），是否接收了一个有效的 DNS 服务器 COOKIE（V），以及在没有出现一个有效的服务器 COOKIE 的情况下是否有一个 DNS COOKIE 选项（K）。在这之后，记录请求被发送到的目标地址。最后，如果客户端请求中存在任何 CLIENT-SUBNET 选项，它会以格式 [ECS address/source/scope] 被包含在方括号中。

client 127.0.0.1#62536 (www.example.com): query: www.example.com IN AAAA +SE

client ::1#62537 (www.example.net): query: www.example.net IN AAAA -SE

（这条日志消息的第一部份，显示了客户端地址/端口号和请求名，在随后所有与同样请求相关的日志消息中都被重复。）

query-errors 关于导致失败的请求信息。

rate-limit 一个响应流的比率限制的启动、周期性及结束通知将以 info 级别被写入这个类别。这些消息包含响应域名的一个散列值和域名自身，没有足够的内存来记录结束通知的名字这种情况除外。结束通知通常延迟到比率限制停止一分钟之后。内存不足可能导致仓促发出结束通知，这种情况以一个初始星号（*）指明。各种内部事件以调试级别 1 或更高级别记录日志。

对单个请求的比率限制记录在 query-errors 类别中。

resolver DNS 解析，例如由缓存名字服务器所进行的递归查询给客户端的影响。

rpz 关于响应策略区文件中错误，重写响应以及在最高级别的 debug 中的试图重写响应等的信息。

security 同意和拒绝请求。

serve-stale 指示是否一个旧的答复被用于跟随一个解析器失败。

spill 被终止的请求，要么被丢弃，要么响应 SERVFAIL，是解析限制配额被超过后的结果。

trust-anchor-telemetry named 所收到的信任锚遥测（trust-anchor-telemetry）请求。

unmatched named 不能够决定的类别，或者没有合适的 view 与之匹配的消息。一个单行的摘要也记入 client 类别。这个类别最好发送到一个文件或标准错误；缺省时它被发送到 null 通道。

update 动态更新。

update-security 同意和拒绝更新请求。

xfer-in 服务器所接受的区传送。

xfer-out 服务器所发出的区传送。

zoneload 装载区并创建自动的空区。

query-errors 类别

query-errors 类别是用于指示特定的请求为什么以及怎样导致错误响应。通常，这些消息以 debug 日志级被记录；注意！然而如果请求日志是活跃的，一些消息将以 info 级别记录。日志级别描述如下：

在 debug 级别 1 或更高 - 或者当请求日志活跃时的 info - 每个带有响应码 SERVFAIL 的响应将按如下方式记录日志：

client 127.0.0.1#61502: query failed (SERVFAIL) for www.example.com/IN/AAAA at query.c:3880

这表示在源文件 query.c 的第 3880 行检测到一个导致 SERVFAIL 的错误。这个级别的日志消息对识别一个权威服务器中导致 SERVFAIL 的原因特别有帮助。

在 debug 的级别 2 或更高级别时，会记录导致 SERVFAIL 的递归解析的详细上下文信息。日志消息将像下面这样：

```
fetch completed at resolver.c:2970 for www.example.com/A
in 10.000183: timed out/success [domain:example.com,
referral:2,restart:7,qrysent:8,timeout:5,lame:0,quota:0,neterr:0,
badresp:1,adberr:0,findfail:0,valfail:0]
```

在冒号之前的第一部份表示一个对 www.example.com 的 AAAA 记录的递归解析在 10.000183 秒内完成，并最终结果导致一个 SERVFAIL，它是由源文件 resolver.c 的第 2970 行决定的。

下一个部份显示了所检测到的最终结果和 DNSSEC 验证的最近结果。当不试图进行验证时，后者总是“成功”。在这个例子中，这个请求可能导致 SERVFAIL，是因为所有的名字服务器都宕机或不可达，并在 10 秒内产生了一个超时。很可能不试图进行 DNSSEC 验证。

最后部份，包含在方括号中，显示了这次特定的解析过程中搜集的统计。domain 字段显示解析器所到达的最深的区；它是最终检测到错误的区。其它字段的含义在下面的列表中总结。

referral 在解析过程中解析器所收到的引用个数。在上面的 example.com 中是 2。

restart 解析器试探 domain 区的远程服务器循环次数。在每一个循环，解析器向 domain 区的每个已知的名字服务器发送一个请求（有可能重发，取决于响应）。

qrysent 解析器发送到 domain 区的请求数。

timeout 解析器收到最后响应之后的超时次数。

lame 解析器所检测到的 domain 区的跛服务器数目。一个服务器被检测为跛服务器，要么是因为一个不正确的响应，要么是在 BIND 9 的地址数据库（ADB）中的查找结果，这个数据库缓存了跛服务器。

quota 解析器不能发出请求的次数，由于它超过了针对某一服务器的取数据限额。

neterr 解析器在发送请求到 domain 区时遇到的错误结果数目。一个通常的情形是远程服务器不可达，解析器收到一个 ICMP 不可达错误消息。

badresp 解析器在发送请求到 domain 区时所收到的意料之外的响应（lame 之外）数目。

adberr 在 ADB 中查找 domain 区的远程服务器地址时的失败次数。一个通常的此类情形是远程服务器的名字没有任何地址记录。

findfail 解析远程服务器地址时的失败次数。这是贯穿解析过程的失败总数。

valfail DNSSEC 验证失败次数。所计算的验证失败贯穿解析过程（不限于 domain 区），但是应该仅仅是发生在 domain 内的。

在 debug 的级 3 或更高级别时，会记录那些与在调试级别 debug 级别 1 相同的消息，但只对其它错误，而不记录 SERVFAIL。注意，像 NXDOMAIN 这样的否定响应并不是错误，不会记录在这个调试级。

在 debug 的级 4 或更高级别时，对 SERVFAIL 之外的其它错误和 NXDOMAIN 这样的否定响应，会记录 debug 级 2 所记录的详细上下文信息。

4.2.11 masters 语句语法

```
masters <string> [ port <integer> ] [ dscp
  <integer> ] { ( <masters> | <ipv4_address> [
  port <integer> ] | <ipv6_address> [ port
  <integer> ] ) [ key <string> ]; ... };
```

4.2.12 masters 语句定义和用法

masters 列表让一组共同的主服务器易于被用于多个存根区和辅区，通过放入后者的 masters 或 also-notify 列表中。

4.2.13 options 语句语法

这是 named.conf 文件中 options 语句的语法：

```
options {
    allow-new-zones <boolean>;
    allow-notify { <address_match_element>; ... };
    allow-query { <address_match_element>; ... };
    allow-query-cache { <address_match_element>; ... };
    allow-query-cache-on { <address_match_element>; ... };
    allow-query-on { <address_match_element>; ... };
    allow-recursion { <address_match_element>; ... };
    allow-recursion-on { <address_match_element>; ... };
    allow-transfer { <address_match_element>; ... };
    allow-update { <address_match_element>; ... };
    allow-update-forwarding { <address_match_element>; ... };
    also-notify [ port <integer> ] [ dscp <integer> ] { ( <masters> |
      <ipv4_address> [ port <integer> ] | <ipv6_address> [ port
      <integer> ] ) [ key <string> ]; ... };
    alt-transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * )
      ] [ dscp <integer> ];
    alt-transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> |
      * ) ] [ dscp <integer> ];
    answer-cookie <boolean>;
    attach-cache <string>;
    auth-nxdomain <boolean>; // default changed
```

（下页继续）

(续上页)

```
auto-dnssec ( allow | maintain | off );
automatic-interface-scan <boolean>;
avoid-v4-udp-ports { <portrange>; ... };
avoid-v6-udp-ports { <portrange>; ... };
bindkeys-file <quoted_string>;
blackhole { <address_match_element>; ... };
cache-file <quoted_string>;
catalog-zones { zone <string> [ default-masters [ port <integer> ]
  [ dscp <integer> ] { ( <masters> | <ipv4_address> [ port
  <integer> ] | <ipv6_address> [ port <integer> ] ) [ key
  <string> ]; ... } ] [ zone-directory <quoted_string> ] [
  in-memory <boolean> ] [ min-update-interval <duration> ]; ... };
check-dup-records ( fail | warn | ignore );
check-integrity <boolean>;
check-mx ( fail | warn | ignore );
check-mx-cname ( fail | warn | ignore );
check-names ( primary | master |
  secondary | slave | response ) (
  fail | warn | ignore );
check-sibling <boolean>;
check-spf ( warn | ignore );
check-srv-cname ( fail | warn | ignore );
check-wildcard <boolean>;
clients-per-query <integer>;
cookie-algorithm ( aes | siphash24 );
cookie-secret <string>;
coresize ( default | unlimited | <sizeval> );
datasize ( default | unlimited | <sizeval> );
deny-answer-addresses { <address_match_element>; ... } [
  except-from { <string>; ... } ];
deny-answer-aliases { <string>; ... } [ except-from { <string>; ...
  } ];
dialup ( notify | notify-passive | passive | refresh | <boolean> );
directory <quoted_string>;
disable-algorithms <string> { <string>;
  ... };
disable-ds-digests <string> { <string>;
  ... };
disable-empty-zone <string>;
```

(下页继续)

(续上页)

```
dns64 <netprefix> {
    break-dnssec <boolean>;
    clients { <address_match_element>; ... };
    exclude { <address_match_element>; ... };
    mapped { <address_match_element>; ... };
    recursive-only <boolean>;
    suffix <ipv6_address>;
};
dns64-contact <string>;
dns64-server <string>;
dnskey-sig-validity <integer>;
dnsrps-enable <boolean>;
dnsrps-options { <unspecified-text> };
dnssec-accept-expired <boolean>;
dnssec-dnskey-kskonly <boolean>;
dnssec-loadkeys-interval <integer>;
dnssec-must-be-secure <string> <boolean>;
dnssec-policy <string>;
dnssec-secure-to-insecure <boolean>;
dnssec-update-mode ( maintain | no-resign );
dnssec-validation ( yes | no | auto );
dnstap { ( all | auth | client | forwarder |
  resolver | update ) [ ( query | response ) ];
  ... };
dnstap-identity ( <quoted_string> | none |
  hostname );
dnstap-output ( file | unix ) <quoted_string> [
  size ( unlimited | <size> ) ] [ versions (
  unlimited | <integer> ) ] [ suffix ( increment
  | timestamp ) ];
dnstap-version ( <quoted_string> | none );
dscp <integer>;
dual-stack-servers [ port <integer> ] { ( <quoted_string> [ port
  <integer> ] [ dscp <integer> ] | <ipv4_address> [ port
  <integer> ] [ dscp <integer> ] | <ipv6_address> [ port
  <integer> ] [ dscp <integer> ] ); ... };
dump-file <quoted_string>;
edns-udp-size <integer>;
empty-contact <string>;
```

(下页继续)

(续上页)

```
empty-server <string>;
empty-zones-enable <boolean>;
fetch-quota-params <integer> <fixedpoint> <fixedpoint> <fixedpoint>;
fetches-per-server <integer> [ ( drop | fail ) ];
fetches-per-zone <integer> [ ( drop | fail ) ];
files ( default | unlimited | <sizeval> );
flush-zones-on-shutdown <boolean>;
forward ( first | only );
forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address>
  | <ipv6_address> ) [ port <integer> ] [ dscp <integer> ]; ... };
fstrm-set-buffer-hint <integer>;
fstrm-set-flush-timeout <integer>;
fstrm-set-input-queue-size <integer>;
fstrm-set-output-notify-threshold <integer>;
fstrm-set-output-queue-model ( mpsc | spsc );
fstrm-set-output-queue-size <integer>;
fstrm-set-reopen-interval <duration>;
geoip-directory ( <quoted_string> | none );
glue-cache <boolean>;
heartbeat-interval <integer>;
hostname ( <quoted_string> | none );
inline-signing <boolean>;
interface-interval <duration>;
ixfr-from-differences ( primary | master | secondary | slave |
  <boolean> );
keep-response-order { <address_match_element>; ... };
key-directory <quoted_string>;
lame-ttl <duration>;
listen-on [ port <integer> ] [ dscp
  <integer> ] {
  <address_match_element>; ... };
listen-on-v6 [ port <integer> ] [ dscp
  <integer> ] {
  <address_match_element>; ... };
lmdb-mapsize <sizeval>;
lock-file ( <quoted_string> | none );
managed-keys-directory <quoted_string>;
masterfile-format ( map | raw | text );
masterfile-style ( full | relative );
```

(下页继续)

(续上页)

```
match-mapped-addresses <boolean>;
max-cache-size ( default | unlimited | <sizeval> | <percentage> );
max-cache-ttl <duration>;
max-clients-per-query <integer>;
max-journal-size ( default | unlimited | <sizeval> );
max-ncache-ttl <duration>;
max-records <integer>;
max-recursion-depth <integer>;
max-recursion-queries <integer>;
max-refresh-time <integer>;
max-retry-time <integer>;
max-rsa-exponent-size <integer>;
max-stale-ttl <duration>;
max-transfer-idle-in <integer>;
max-transfer-idle-out <integer>;
max-transfer-time-in <integer>;
max-transfer-time-out <integer>;
max-udp-size <integer>;
max-zone-ttl ( unlimited | <duration> );
memstatistics <boolean>;
memstatistics-file <quoted_string>;
message-compression <boolean>;
min-cache-ttl <duration>;
min-ncache-ttl <duration>;
min-refresh-time <integer>;
min-retry-time <integer>;
minimal-any <boolean>;
minimal-responses ( no-auth | no-auth-recursive | <boolean> );
multi-master <boolean>;
new-zones-directory <quoted_string>;
no-case-compress { <address_match_element>; ... };
nocookie-udp-size <integer>;
notify ( explicit | master-only | <boolean> );
notify-delay <integer>;
notify-rate <integer>;
notify-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [
  dscp <integer> ];
notify-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ]
  [ dscp <integer> ];
```

(下页继续)

(续上页)

```
notify-to-soa <boolean>;
nta-lifetime <duration>;
nta-recheck <duration>;
nxdomain-redirect <string>;
pid-file ( <quoted_string> | none );
port <integer>;
preferred-glue <string>;
prefetch <integer> [ <integer> ];
provide-ixfr <boolean>;
qname-minimization ( strict | relaxed | disabled | off );
query-source ( ( [ address ] ( <ipv4_address> | * ) [ port (
  <integer> | * ) ] ) | ( [ [ address ] ( <ipv4_address> | * ) ]
  port ( <integer> | * ) ) ) [ dscp <integer> ];
query-source-v6 ( ( [ address ] ( <ipv6_address> | * ) [ port (
  <integer> | * ) ] ) | ( [ [ address ] ( <ipv6_address> | * ) ]
  port ( <integer> | * ) ) ) [ dscp <integer> ];
querylog <boolean>;
random-device ( <quoted_string> | none );
rate-limit {
    all-per-second <integer>;
    errors-per-second <integer>;
    exempt-clients { <address_match_element>; ... };
    ipv4-prefix-length <integer>;
    ipv6-prefix-length <integer>;
    log-only <boolean>;
    max-table-size <integer>;
    min-table-size <integer>;
    nodata-per-second <integer>;
    nxdomains-per-second <integer>;
    qps-scale <integer>;
    referrals-per-second <integer>;
    responses-per-second <integer>;
    slip <integer>;
    window <integer>;
};
recursing-file <quoted_string>;
recursion <boolean>;
recursive-clients <integer>;
request-expire <boolean>;
```

(下页继续)

(续上页)

```
request-ixfr <boolean>;
request-nsid <boolean>;
require-server-cookie <boolean>;
reserved-sockets <integer>;
resolver-nonbackoff-tries <integer>;
resolver-query-timeout <integer>;
resolver-retry-interval <integer>;
response-padding { <address_match_element>; ... } block-size
  <integer>;
response-policy { zone <string> [ add-soa <boolean> ] [ log
  <boolean> ] [ max-policy-ttl <duration> ] [ min-update-interval
  <duration> ] [ policy ( cname | disabled | drop | given | no-op
  | nodata | nxdomain | passthru | tcp-only <quoted_string> ) ] [
  recursive-only <boolean> ] [ nsip-enable <boolean> ] [
  nsdname-enable <boolean> ]; ... } [ add-soa <boolean> ] [
  break-dnssec <boolean> ] [ max-policy-ttl <duration> ] [
  min-update-interval <duration> ] [ min-ns-dots <integer> ] [
  nsip-wait-recurse <boolean> ] [ qname-wait-recurse <boolean> ]
  [ recursive-only <boolean> ] [ nsip-enable <boolean> ] [
  nsdname-enable <boolean> ] [ dnsrps-enable <boolean> ] [
  dnsrps-options { <unspecified-text> } ];
root-delegation-only [ exclude { <string>; ... } ];
root-key-sentinel <boolean>;
rrset-order { [ class <string> ] [ type <string> ] [ name
  <quoted_string> ] <string> <string>; ... };
secroots-file <quoted_string>;
send-cookie <boolean>;
serial-query-rate <integer>;
serial-update-method ( date | increment | unixtime );
server-id ( <quoted_string> | none | hostname );
servfail-ttl <duration>;
session-keyalg <string>;
session-keyfile ( <quoted_string> | none );
session-keyname <string>;
sig-signing-nodes <integer>;
sig-signing-signatures <integer>;
sig-signing-type <integer>;
sig-validity-interval <integer> [ <integer> ];
sortlist { <address_match_element>; ... };
```

(下页继续)

(续上页)

```
    stacksize ( default | unlimited | <sizeval> );
    stale-answer-enable <boolean>;
    stale-answer-ttl <duration>;
    stale-cache-enable <boolean>;
    startup-notify-rate <integer>;
    statistics-file <quoted_string>;
    synth-from-dnssec <boolean>;
    tcp-advertised-timeout <integer>;
    tcp-clients <integer>;
    tcp-idle-timeout <integer>;
    tcp-initial-timeout <integer>;
    tcp-keepalive-timeout <integer>;
    tcp-listen-queue <integer>;
    tkey-dhkey <quoted_string> <integer>;
    tkey-domain <quoted_string>;
    tkey-gssapi-credential <quoted_string>;
    tkey-gssapi-keytab <quoted_string>;
    transfer-format ( many-answers | one-answer );
    transfer-message-size <integer>;
    transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [
      dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * )
      ] [ dscp <integer> ];
    transfers-in <integer>;
    transfers-out <integer>;
    transfers-per-ns <integer>;
    trust-anchor-telemetry <boolean>; // experimental
    try-tcp-refresh <boolean>;
    update-check-ksk <boolean>;
    use-alt-transfer-source <boolean>;
    use-v4-udp-ports { <portrange>; ... };
    use-v6-udp-ports { <portrange>; ... };
    v6-bias <integer>;
    validate-except { <string>; ... };
    version ( <quoted_string> | none );
    zero-no-soa-ttl <boolean>;
    zero-no-soa-ttl-cache <boolean>;
    zone-statistics ( full | terse | none | <boolean> );
};
```

4.2.14 options 语句定义和用法

options 语句设置 BIND 所使用的全局选项。这个语句在一个配置文件中只能出现一次。如果没有 options 语句，将会使用一个 options 块，其中包含每个选项的缺省值。

attach-cache 这个选项允许多个视图共享一个缓存数据库。缺省时每个视图拥有自己的缓存数据库，但是如果多个视图在名字解析和缓存上有同样的操作策略，通过使用这个选项，这些视图可以共享一个缓存数据库来节约内存，并有可能提高解析效率。

attach-cache 选项也可以在 view 语句中指定，这种情况下，它会覆盖全局的 attach-cache 选项。

cache_name 指定要共享的缓存。当 named 服务器配置要共享一个缓存的视图时，它使用指定的名字为这些共享视图中的第一个视图建立一个缓存。其余的视图只是简单地引用已经建立的视图。

一个通常的共享一个缓存的配置是允许所有的视图共享一个视图。这个可以通过在全局选项中为 attach-cache 指定一个任意名字而完成。

另一个可能的操作是允许全部视图中的一个子集共享一个缓存，而其它的视图使用各自的缓存。例如，如果有三个视图 A，B 和 C，并且只有 A 和 B 应该共享一个缓存，指定 attach-cache 选项作为视图 A（或 B）的选项，使其引用其它的视图名：

```
view "A" {
 // this view has its own cache
 ...
};
view "B" {
 // this view refers to A's cache
 attach-cache "A";
};
view "C" {
 // this view has its own cache
 ...
};
```

共享同一个缓存的视图必须在可能影响缓存的配置参数上有同样的策略。当前的实现要求在这些视图在下列配置选项上要一致：check-names ，dnssec-accept-expired ，dnssec-validation ，max-cache-ttl ，max-ncache-ttl ，max-stale-size ，max-cache-size ，min-cache-size ，min-ncache-size 和 zero-no-soa-ttl 。

注意有可能其它参数也会带来混乱，如果它们在不同的视图共享一个缓存时不一致的话。例

如，如果这些视图定义不同的转发者集合，而这些转发者对同样的问题可能返回不同的回答，共享这些回答可能就没有意义甚至是有害的。确保在不同的视图中的配置差异不会导致对共享缓存的破坏是系统管理员的责任。

directory 这个设置服务器的工作目录。所有在配置文件中出现的非绝对路径都是相对于这个目录的。服务器的大多数输出文件（例如：named.run ）都是以此为缺省位置。如果未指定目录，工作目录缺省为'.'，即服务器开始运行的目录。这个目录应该被指定为一个绝对路径，并且必须对 named 进程的有效用户 ID 是可写的。

dnstap dnstap 是一个捕捉和记录 DNS 流量的快速、灵活的方法。由 Farsight Security 公司的 Robert Edmonds 所开发，并由多个 DNS 实现所支持，dnstap 使用 libfstrm（一个轻量级高速框架库，参见 https://github.com/farsightsec/fstrm ）发送以协议缓冲（libprotobuf-c ，一个由 Google 公司所开发的序列化结构数据的机制；参见 https://developers.google.com/protocol-buffers）编码的事件荷载。

要在编译时开启 dnstap ， fstrm 和 protobuf-c 库必须是可用的，并且 BIND 必须在配置时使用 --enable-dnstap 。

dnstap 选项是一个带括号的、被记录消息类型的列表。这些在每个视图中可以被设置为不同的值。支持的类型是 client ， auth resolver ， forwarder 和 update 。指定类型 all 将使所有 dnstap 消息被日志记录，而不考虑其类型。

每种类型可以接受一个额外的参数来指定是否记录 query 消息或 response 消息；如果未指定，则请求和响应都记录。

例如：要记录所有的权威请求和响应，递归客户端的响应和解析器发出的上游请求，使用：

```
dnstap {
 auth;
 client response;
 resolver query;
};
```

所记录的 dnstap 消息可以使用 dnstap-read 应用程序进行分析（参见dnstap-read - 以人可读的格式输出 dnstap 数据 获取详细信息）。

更多关于 dnstap 的信息，参见 http://dnstap.info。

fstrm 库有许多可调项提供给 named.conf ，并且如果需要增进性能或者阻止数据丢失，可以修改它们。可调项有：

- fstrm-set-buffer-hint ： 在强制刷新一个缓冲区之前允许输出缓存中积累的字节数限额。最小值是 1024，最大值是 65536，缺省值是 8192。

- fstrm-set-flush-timeout： 允许待刷新数据停留在输出缓存中的秒数。最小值是 1 秒，最大值是 600 秒（10 分钟），缺省值是 1 秒。
- fstrm-set-output-notify-threshold： 在唤醒 I/O 线程之前允许存在于一个输入队列中的未解决队列条目的数目。最小值是 1，缺省值是 32.
- fstrm-set-output-queue-model： 控制队列语义以使用队列对象。缺省值是 mpsc（多个生产者，单个消费者）；另外的选项是 spsc （单个生产者，单个消费者）。
- fstrm-set-input-queue-size： 为每个输入队列申请的队列条目的数目。这个值必须是 2 的整数次幂。最小值是 2，最大值是 16384，缺省值是 512。
- fstrm-set-output-queue-size：为每个输出队列申请的队列条目的数目。最小值是 2，最大值依赖于系统并基于 IOV_MAX ，缺省值是 64。
- fstrm-set-reopen-interval： 在重新打开一个已关闭输出流之间需要等待的秒数。最小值是 1 秒，最大值是 600 秒（10 分钟），缺省值是 5 秒。为方便起见，可以为这个值指定 TTL 风格的时间单位后缀。

注意以上所有最小值、最大值和缺省值都由 libfstrm 库设置，并且可能服从这个库未来版本的变化。参见 libfstrm 文档以获得更多信息。

dnstap-output 如果 dnstap 在编译时开启并被激活，这个配置 dnstap 框架流的发送路径。

第一个参数要么是 file ，要么是 unix ，指明目标是一个文件或 UNIX 域套接字。第二个参数是一个文件或者一个 UNIX 域套接字。（注意：当使用一个套接字时，dnstap 消息仅在另一个进程，例如 fstrm_capture （由 libfstrm 提供），监听了这个套接字时才发送。）

如果第一个参数是 file ，就可以最大添加三个附加选项：size 指示一个 dnstap 日志文件在轮转到一个新文件之前可以增长到的最大大小；versions 指定要保持的已经轮转的日志文件的数量；而 suffix 指示是以一个增长的计数器（increment ）还是以当前时间戳（timestamp ）作为已经轮转的日志文件的后缀。这些类似于在一个 logging 通道中的 size ， versions 和 suffix 选项。缺省是允许 dnstap 日志文件增长到任何大小而没有轮转。

dnstap-output 只能在 options 中全局设置。当前，它只能在 named 运行时设置一次；一旦设置，不能通过 rndc reload 或 rndc reconfig 修改。

dnstap-identity 这个指定一个要在 dnstap 消息中发送的 identity 字符串。如果设置为 hostname ，这是缺省的，将发送服务器主机名。如果设置为 none ，不发送标识字符串。

dnstap-version 这个指定一个要在 dnstap 消息中发送的 version 字符串。缺省是 BIND 的版本号。如果设置为 none ，不发送版本字符串。

geoip-directory 当 named 编译时带有 MaxMind GeoIP2 地理位置 API 时，这指定包含 GeoIP 数据库文件的目录。缺省时，这个选项是基于编译 libmaxminddb 模块时用到的前缀：例如，

如果库时安装在 /usr/local/lib ，则缺省的 geoip-directory 是 /usr/local/share/GeoIP 。在 Windows 上，缺省为 named 工作目录。参见acl 语句定义和用法 以获取关于 geoip ACL 更详细的内容。

key-directory 当对安全区执行一次动态更新时，这是能够从中找到 DNSSEC 公钥和私钥文件的目录，如果不是当前工作目录。（注意这个选项对如同 bind.keys ，rndc.key 或 session.key 这样包含非 DNSSEC 密钥的文件的路径时无效。）

lmdb-mapsize 当 named 构建时带有 liblmdb，这个选项为映射 new-zone 数据库（NZD），以 LMDB 数据库格式，设置最大的内存大小。这个数据库用于存储使用 rndc addzone 添加的区的配置信息。注意这不是 NZD 数据库文件的大小，而是数据库可能增长到的最大大小。

由于数据库文件是内存映射，它的大小被 named 进程的地址空间所限制。缺省值选择 32 兆字节适合用于 32 位的 named 构建。最大允许的值是 1 太字节。假设没有复杂 ACL 的典型区配置，一个 32 MB 的 NZD 文件应该能够容纳大约 100,000 个区的配置。

managed-keys-directory 这个指定保存跟踪被管理 DNSSEC 密钥的文件的目录（即，这些配置在一条 trust-anchors 语句中使用 initial-key 或 initial-ds 关键字）。缺省时，它就是工作目录。这个目录对 named 进程的有效用户 ID **必须**是可写的。

如果 named 被配置为没有使用视图，服务器的被管理密钥会保存在一个名为 managed-keys.bind 的文件中。否则，被管理密钥在不同的文件中被跟踪，每个视图一个文件；每个文件名都是视图名（或者，如果视图名包含不能用于文件名的字符，就是视图名的 SHA256 HASH 值），再加上后缀 .mkeys 。

（注意：在先前的版本中，视图对应的文件名总是使用视图名的 SHA256 hash。为确保以后升级的兼容性，如果发现存在一个使用旧名字格式的文件，将使用新格式来替换。）

max-ixfr-ratio 这个设置一个增量传输大小的阈值（表示为完整区的一个百分比），在响应区传送请求时，如果超过了这个阈值，named 会选择一个 AXFR 响应而不是 IXFR。参见增量区传送（IXFR） 。

new-zones-directory 这个指定存放通过 rndc addzone 增加的区的配置参数的存放目录。缺省时，这就是工作目录。如果设置为一个相对路径，它将是相对于工作目录。这个目录 **必须**是 named 进程的有效用户 ID 可写的。

qname-minimization 这个选项控制 BIND 解析器中 QNAME 最小化行为。当设置为 strict 时，BIND 将遵循 QNAME 最小化算法到字母，如同在 RFC 7816 中所描述。将这个选项设置为 relaxed 将使 BIND 回退到普通（非最小化）请求模式，即当它收到 NXDOMAIN 或其它不可预期的响应（如，SERVFAIL，不正确的区截断，REFUSED）时成为一个最小化请求。disabled 完全关闭 QNAME 最小化。当前缺省是 relaxed ，但是在未来的版本中可能被改为 strict 。

tkey-gssapi-keytab 用于 GSS-TSIG 更新的 KRB5 keytab 文件。如果设置了这个选项并且没有设置 tkey-gssapi-credential，允许使用与指定 keytab 中的一个 principal 相匹配的任何密钥进行更新。

tkey-gssapi-credential 安全证书，服务器应该用它来认证 GSS-TSIG 协议所请求的密钥。当前只能使用 Kerberos 5 认证，并且证书是一个 Kerberos 主，服务器可以通过缺省的系统密钥文件，一般是 /etc/krb5.keytab ，来获取它。keytab 文件的位置可以使用 tkey-gssapi-keytab 选项来重载。通常这个主是这样的形式 DNS/server.domain 。要使用 GSS-TSIG，如果没有使用 tkey-gssapi-keytab 设置一个特定的 keytab，就必须设置 tkey-domain 。

tkey-domain 添加到由 TKEY 生成的所有共享密钥名的域名。当一个客户端请求进行 TKEY 交换时，它可以指定所期望的密钥的名字，也可以不指定。如果指定，共享密钥的名字是 client specified part + tkey-domain 。否则，共享密钥的名字是 random hex digits + tkey-domain 。在大多数情况下，domainname 应该是服务器的域名，或者是一个不存在的子域，如 _tkey.domainname 。如果使用了 GSS-TSIG，必须定义这个变量，除非使用 tkey-gssapi-keytab 指定了一个特定的 keytab。

tkey-dhkey 这是服务器所使用的 Diffie-Hellman 密钥，它用来生成与客户端共享的密钥，使用 TKEY 的 Diffie-Hellman 模式。服务器必须能够从工作目录的文件中加载公钥和私钥。在大多数情况下，key_name 应该是服务器的主机名。

cache-file 这个仅用于测试。不要使用。

dump-file 这是在服务器收到 rndc dumpdb 命令时，转储数据到文件的路径名。如果未指定，缺省为 named_dump.db 。

memstatistics-file 这是服务器在退出时，将内存统计写到的文件路径。如果没有指定，缺省是 named.memstats 。

lock-file 这是 named 首次启动时试图在其中获取一个文件锁的一个文件的路径名；如果不成功，服务器将会终止，这是假设另一个服务器已经正在运行中。如果未指定，缺省是 none 。

指定 lock-file none 关闭使用一个锁文件。如果 named 使用 -X 选项运行，lock-file 将被忽略，它被这个选项所覆盖。如果 named 被重新加载或者重新配置，对 lock-file 的修改将被忽略；它仅在服务器初次启动时有效。

pid-file 这是服务器将其进程号写入的文件。如果未指定，缺省为 /var/run/named/named.pid 。PID 文件是用于要向正在运行的名字服务器发送信号的程序。指定 pid-file none 关闭使用一个 PID 文件；不写文件，如果已经有一个，将被删除。注意 none 是一个关键字，不是一个文件名，所以不使用双引号将其包含。

recursing-file 这是在服务器收到 rndc recursing 命令时，转储当前递归请求到文件的路径名。如果未指定，缺省为 named.recursing 文件。

statistics-file 这是在服务器收到 rndc stats 命令时，追加统计数据的文件路径。如果未指定，缺省为服务器当前目录下的 named.stats 文件。这个文件的格式在统计文件 中描述。

bindkeys-file 这是用于覆盖由 named 所提供的内置信任密钥的文件的路径名。更详细的内容参见对 dnssec-validation 的讨论。如果未指定，缺省是 /etc/bind.keys 。

secroots-file 这是在服务器收到 rndc secroots 命令时，转储安全根的目的文件的路径名。如果未指定，缺省为 named.secroots 。

session-keyfile 这是存放一个由 named 所生成的、用于 nsupdate -l 的 TSIG 会话密钥的文件的路径名。如果未指定，缺省是 /var/run/named/session.key 。（参见动态更新策略 ，特别是对 update-policy 语句的 local 选项的讨论，以获取关于这个特征的更多信息。）

session-keyname 这是用于 TSIG 会话密钥的密钥名。如果未指定，缺省是 local-ddns 。

session-keyalg 这是用于 TSIG 会话密钥的算法。有效值为 hmac-sha1，hmac-sha224，hmac-sha256，hmac-sha384，hmac-sha512 和 hmac-md5。如果未指定，缺省为 hmac-sha256。

port 这是服务器用于接收和发送 DNS 协议流量的 UDP/TCP 端口。缺省是 53。这个选项的主要目的是服务器测试；一个使用 53 端口之外的服务器将无法与全球的 DNS 进行通信。

dscp 这是全局的差分服务代码点（DSCP，Differentiated Services Code Point）值，用于在支持 DSCP 的操作系统上分类所发出的 DNS 流量。有效的值为从 0 到 63。缺省未配置。

random-device 这指定一个服务器使用的熵源；这是一个从中读取熵的设备或文件。如果它是一个文件，在文件被耗尽时，请求熵的操作将会失败。

熵被用于加密操作中，例如 TKEY 事务，签名区的动态更新和生成 TSIG 回话密钥。它也用于伪随机数生成器的生成种子和扰动，伪随机数生成器用于较不太关键的要求随机性的函数，如 DNS 消息事务 ID 的生成。

如果未指定 random-device ，或者它被设置为 none ，将会从 BIND 所链接到的加密库（即，OpenSSL 或者一个 PKCS#11 提供者）所提供的随机数生成函数中读取熵。

random-device 选项在服务器启动时的初始配置装载时生效，而在随后的重载将会忽略它。

preferred-glue 如果指定，在一个请求响应的附加部份内的其它粘着记录之前，所列的类型（A 或 AAAA）将被写入。缺省是优先为 A 记录，如果在响应经由 IPv4 到达的请求时，以及优先为 AAAA 记录，如果在响应经由 IPv6 到达的请求时。

root-delegation-only 这使用一个可选的排除列表，在 TLD（顶级域）和根区中打开只授权模式的增强特性。

只授权区期待接受和回复 DS 请求。这样的请求和响应被当成只授权处理的一个例外，并且不能转换 NXDOMAIN 响应，前提是没有找到请求名字的 CNAME 记录。

如果一个只授权区的服务器同时也作为一个子区的服务器，就不可能都能够判断一个应答是来自于只授权区还是子区。SOA 记录、NS 记录和 DNSKEY 记录是区顶点仅有的记录，并且包含这些记录或者 DS 记录的一个匹配的响应被当作是来自于一个子区。同时也检查 RRSIG 记录看其是否被一个子区签名。还要检查权威部份看是否有应答来自子区的迹象。被确认为来自一个子区的应答不会被转换为 NXDOMAIN 响应。尽管有所有这些检查，在同时提供一个子区服务时，仍然存在错误否定响应的可能性。

类似地，错误肯定响应也可能产生于只授权区中的空节点（名字中没有记录）且请求类型不是 ANY 时。

注意一些 TLD 不是只授权的；如："DE"，"LV"，"US" 和 "MUSEUM"。这个列表并不完整。

```
options {
  root-delegation-only exclude { "de"; "lv"; "us"; "museum"; };
};
```

disable-algorithms 这关掉在指定名字下的指定 DNSSEC 算法。允许使用多个 disable-algorithms 语句。只有最与 disable-algorithms 匹配的子句将被用于决定使用哪个算法。

如果所有支持的算法都被关闭，由 disable-algorithms 所覆盖的区将被视为不安全的。

在 trust-anchors （或 managed-keys 或 trusted-keys ）中配置的信任锚如果匹配到被关闭的算法，会被忽略，就如同他们完全未被配置一样。

disable-ds-digests 这关闭在指定的名字及之下名字的指定的 DS 摘要类型。允许使用多个 disable-ds-digests 语句。只有最与 disable-ds-digests 匹配的子句将被用于决定使用哪个摘要类型。

如果所有支持的摘要类型都被关闭，由 disable-ds-digests 所覆盖的区将被视为不安全的。

dnssec-must-be-secure 这指定肯定或者可能不安全（被签名和校验）的层次体系。如果是 yes ，named 将仅仅接受安全的回答。如果为 no ，应用普通的 DNSSEC 校验，也接受不安全的回答。指定的域必须为其定义一个信任锚，例如在一个 trust-anchors 语句中，或者 dnssec-validation auto 必须被激活。

dns64 这条指令指示 named 在没有找到 AAAA 记录时，返回由 IPv4 地址所映射到的 AAAA 查询。其意图是用于和 NAT64 相配合。每个 dns64 定义一个 DNS64 前缀。可以定义多个 DNS64 前缀。

按照 RFC 6052 ，兼容的 IPv6 前缀可以有 32，40，48，56，64 和 96 这些长度。其中的位 64..71 必须为 0，且带有位置 0 的前缀最高位。

另外，使用合成的 CNAME 记录为前缀创建一个反向 IP6.ARPA 区，提供了一个从 IP6.ARPA

名字到对应的 IN-ADDR.ARPA 名字的映射。dns64-server 和 dns64-contact 可以用于指定服务器的名字和区的联系人。这些也可以在 view/options 级中设置，而不能在每个前缀的基础上设置。

每个 dns64 支持一个可选的 clients 访问控制表，它决定哪些客户端受这条指令的影响。如果未设置，缺省值为 any; 。

每个 dns64 支持一个可选的 mapped 访问控制表，它在相关的 A 资源记录集中选择哪些 IPv4 地址会被映射。如果未设置，缺省值为 any; 。

通常地，DNS64 不会应用于一个拥有一个或多个 AAAA 记录的域名；只是简单地返回这些记录。可选的 exclude 访问控制表允许规定一个 IPv6 地址的列表，如果它们出现在一个域名的 AAAA 记录中，将被忽略，并且 DNS64 将被用于这个域名所拥有的任何 A 记录。如果未定义，exclude 缺省为::ffff:0.0.0.0/96。

也可以定义一个可选的 suffix ，用来设置映射到 IPv4 地址位的结尾部分。缺省时这些位被设置为 :: 。这些位匹配的前缀和映射的 IPv4 地址必须为零。

如果 recursive-only 被设置为 yes ，DNS64 合成仅仅发生在递归请求。缺省是 no 。

如果 break-dnssec 被设置为 yes ，将会进行 DNS64 合成，即使结果在验证时会导致一个 DNSSEC 验证失败。如果这个选项被设置为 no （缺省值），进来的请求中带有 DO 位，在可用的记录中有 RRSIG，这时不会进行 DNS64 合成。

```
acl rfc1918 { 10/8; 192.168/16; 172.16/12; };

dns64 64:FF9B::/96 {
  clients { any; };
  mapped { !rfc1918; any; };
  exclude { 64:FF9B::/96; ::ffff:0000:0000/96; };
  suffix ::;
};
```

dnssec-loadkeys-interval 当一个区配置了 auto-dnssec maintain; ，必须定期检查其密钥仓库，以查看是否有新密钥被添加或者任何现有密钥的时间元数据被更新（参见dnssec-keygen: DNSSEC 密钥生成工具 和dnssec-settime: 为一个 DNSSEC 密钥设置密钥定时元数据 ）。dnssec-loadkeys-interval 选项设置仓库自动检查的频率，以分钟计。缺省是 60 （1 小时），最小是 1 （1 分钟），最大是 1440 （24 小时）；任何更大的值将被静默地减小。

dnssec-policy 它为区指定使用哪个密钥和签名策略（KSAP）。这是一个指向一个 dnssec-policy 语句的字符串。有两个内建的策略：default ，这使用缺省的策略，和 none ，这表示没有 DNSSEC 策略，而区不签名。缺省为 none 。更详细的信息参见dnssec-policy Grammar 。

dnssec-update-mode 如果在一个 master 类型的区中这个选项被设为其缺省值 maintain，而这个区是 DNSSEC 签名的并被配置为允许动态更新（参见动态更新策略），并且如果 named 能够访问区的私有签名密钥，named 就会自动对所有新的或修改过的记录签名并通过重新生成 RRSIG 记录维护区的签名，无论其是否过期。

如果这个选项被改为 no-resign，named 将只对所有新的或修改过的记录签名，但不安排对签名的维护。

对这两个设置中的任何一种，如果签名密钥是未激活或者对 named 不可用，named 都会拒绝对一个 DNSSEC 签名区的更新。（一个计划的第三个选项，external，将会关闭所有自动签名并允许将 DNSSEC 数据提交给一个通过动态更新的区；这个还未实现。）

nta-lifetime 这个为由 rndc nta 所添加的否定信任锚指定以秒计的缺省生命周期。

一个否定信任锚选择性关闭对已知将会因为错误配置而非攻击导致验证失败的区的 DNSSEC 验证。当要被验证的数据处于一个活跃的 NTA（及其之上任何其它已配置的信任锚）或之下，named 将终止 DNSSEC 验证过程并将数据作为非安全的（insecure）而不是伪造的（bogus）。这会持续直到 NTA 的生命周期结束。NTA 的持久化会跨越 named 的重启。

为方便起见，TTL 风格的时间单位后缀可以用于指定以秒，分钟或小时为单位的 NTA 生命周期。它也接受 ISO 8601 的持续时间格式。

nta-lifetime 缺省是一小时。它不能超过一周。

nta-recheck 这指定检查由 rndc nta 所添加的否定信任锚是否仍然需要的频繁程度。

一个否定信任锚通常用在一个域由于操作错误而停止验证；它暂时关闭对这个域的 DNSSEC 验证。为了确保 DNSSEC 验证尽快恢复，named 将周期性发送一个请求到这个域，忽略否定信任锚，以发现其是否可以被验证。如果可以，允许否定信任锚提前过期。

对某个 NTA 的正确性检查可以通过使用 rndc nta -f 关闭，或者将 nta-recheck 设置为零关闭所有 NTA。

为方便起见，TTL 风格的时间单位后缀可以用于指定以秒，分钟或小时为单位的 NTA 重检查间隔。它也接受 ISO 8601 的持续时间格式。

缺省是五分钟。它不能超过 nta-lifetime （后者不能超过一周）。

max-zone-ttl 这指定一个最大可能的以秒计的 TTL 值。为方便起见，TTL 风格的时间单位后缀可以用于指定最大值。在使用一个 masterfile-format 为 text 或者 raw 装载一个区文件时，任何带有超过 max-zone-ttl 的 TTL 值的记录都将导致区被拒绝。

这在 DNSSEC 签名区中很有用，因为在轮转到一个新 DNSKEY 时，旧密钥需要保持可用，直到 RRSIG 记录从缓存中过期。max-zone-ttl 选项保证区中的最大 TTL 不会比所设置的值更大。

（注意：因为 map 格式的文件直接装载到内存中，这个选项不适应它。）

缺省值是 unlimited 。一个为 0 的 max-zone-ttl 被当成 unlimited 看待。

stale-answer-ttl 这为旧答复指定 TTL。缺省是 1 秒。最小的允许值也是 1 秒；一个为 0 的值将被静默地更改为 1 秒。

对要返回的旧答复，它们必须打开，要么在配置文件中使用 stale-answer-enable 或者通过 rndc serve-stale on 。

serial-update-method 被配置为动态 DNS 的区可以使用这个选项设置用于更新 SOA 记录中区序列号的更新方法。

使用缺省的设置 serial-update-method increment; ，SOA 序列号将会在每次区被更新时加一。

当设置为 serial-update-method unixtime; 时，SOA 序列号将被设置为 UNIX 纪元以来的秒数，除非序列号已经大于或等于那个值，这时它只是简单的加一。

当设置为 serial-update-method date; 时，新的 SOA 序列号将是 “YYYYMMDD” 格式的当前日期，后跟两个零，除非已存在的序列号已经大于或等于那个值，这时它只是加一。

zone-statistics 如果为 full ，服务器搜集所有区的统计数据，除非在每个区的配置中被关掉，这可以通过在 zone 语句中指定 zone-statistics terse 或 zone-statistics none 来实现。例如，统计数据包括 DNSSEC 签名操作和每种查询类型的权威响应的数量。缺省是 terse ，提供对区的最小统计（包括名字和当前序列号，但不含请求类型计数器）。

这些统计可以通过 statistics-channel 访问到，或使用 rndc stats 来访问，后者将把数据存放在 statistics-file 所指定的文件中。参见统计文件 。

为向后兼容早期的 BIND 9 版本，zone-statistics 选项也可接受 yes 或 no ； yes 与 full 有同样的含义。对于 BIND 9.10，no 与 none 有同样的含义；之前的版本，它与 terse 相同。

布尔选项

automatic-interface-scan 如果为 yes 且操作系统支持，在接口地址被添加或删除时自动扫描网络接口。缺省是 yes 。这个配置选项不影响基于时间的 interface-interval 选项，并且推荐在操作员确定操作系统支持自动接口扫描时，将基于时间的 interface-interval 设置为 0。

automatic-interface-scan 的实现使用了路由套接字（原文：routing sockets）进行网络接口发现，因此，要使这个特性能够工作，操作系统必须支持路由套接字。

allow-new-zones 如果为 yes ，就可以在运行时通过 rndc addzone 添加区。缺省为 no 。

新添加区的配置参数被存储，这样在服务器重启后也可以持久化。配置信息保存在一个名为 viewname.nzf （或者，如果 named 是带 liblmdb 编译，在一个名为 viewname.nzd 的 LMDB 数据库文件中）。“viewname” 是视图的名字，除非视图名包含不能用于文件名中的字符，这种情况下使用一个视图名的加密 hash 作为替代。

运行时添加的区的配置要么存储在一个 new-zone 文件（NZF），要么存储在一个 new-zone 数据库（NZD）中，取决于 named 在编译时是否链接了 liblmdb。参见rndc - 名字服务器控制工具 以获取关于 rndc addzone 的更多细节。

auth-nxdomain 如果为 yes ，总是在 NXDOMAIN 响应中设置 AA 位，即使服务器并非实际的权威服务器。缺省为 no 。如果使用非常旧的 DNS 软件，可能需要将其设为 yes 。

deallocate-on-exit 这个选项是用于 BIND 8 中，以打开对退出时的内存泄漏检查。BIND 9 忽略这个选项并总是进行检查。

memstatistics 这个在退出时写内存统计到由 memstatistics-file 所指定的文件。缺省是 no ，除非在命令行指定’-m record’，在这种情况下，它是 yes 。

dialup 如果为 yes ，服务器将所有区当成通过即时拨号线路进行区传送，拨号连接是由此服务器的流量所引发。虽然这个设置依据不同的区类型有不同的效果，它压缩了区维护，以使每个动作快速发生，每个 heartbeat-interval 进行一次，理想情况在一次连接中完成。它也抑制了一些通常的区维护流量。缺省为 no 。

如果在 view 和 zone 语句中指定了 dialup 选项，会覆盖全局的 dialup 选项。

如果区是一个主区，服务器将会向其所有辅发出一个 NOTIFY 请求（缺省的）。这将会触发辅服务器（假设其支持 NOTIFY）检查区的序列号，允许辅服务器在连接有效时对区进行验证。NOTIFY 所发向的服务器可以通过 notify 和 also-notify 来进行控制。

如果区是辅区或存根区，服务器将会超越普通的“区更新”（refresh）请求，而只会在 heartbeat-interval 过期时执行，附带发送 NOTIFY 请求。

可以通过使用 notify，它仅仅发送 NOTIFY 消息；使用 notify-passive，它发送 NOTIFY 消息并禁止普通刷新请求；使用 refresh，它禁止普通刷新处理并在 heartbeat-interval 过期时发送刷新请求；以及使用 passive，它仅关闭普通刷新处理等来进行更细的控制。

dialup mode	normal refresh	heart-beat refresh	heart-beat notify
no (default)	yes	no	no
yes	no	yes	yes
notify	yes	no	yes
refresh	no	yes	no
passive	no	no	no
notify-passive	no	no	yes

注意，普通的 NOTIFY 进程不受 dialup 的影响。

flush-zones-on-shutdown 当服务器由于收到 SIGTERM 信号而退出时，刷新或不刷新未完成的区信息写磁盘操作。缺省为 flush-zones-on-shutdown no 。

geoip-use-ecs 这个选项曾经是一个针对权威服务器的 EDNS CLIENT-SUBNET 的试验性实现的一部份。但是现在已被废弃。

root-key-sentinel 如果为 yes ，按照 draft-ietf-dnsop-kskroll-sentinel-08 所描述的响应对根密钥标记的探测。缺省是 yes 。

message-compression 如果为 yes ，DNS 名字压缩被用在给普通请求（不包含 AXFR 或 IXFR，这两个总是使用压缩）的响应中。设置这个选项为 no 能减少服务器上的 CPU 使用并可能提高吞吐量。然而，它增加了响应的大小，这可能导致更多的请求使用 TCP 处理；一个关闭压缩的服务器是与 RFC 1123 的 6.1.3.2. 节不兼容的。缺省是 yes 。

minimal-responses 这个选项控制对响应的权威和附加部份添加记录。这类记录可以被添加到响应中以帮助客户端；例如，NS 或 MX 记录可以附带相应的地址记录在附加部份，避免了一次额外的地址查询。然而，向响应中添加这些记录不是必须的，且需要额外的数据库查找，在安排响应时带来额外的延迟。minimal-responses 的取值为下列四种之一：

- no： 服务器在生成响应时将尽可能完整。
- yes： 服务器将只添加 DNS 协议所要求的记录到权威和附加部份（例如，在返回授权或否定响应时）。这将提供最好的服务器性能，但是可能导致更多的客户端请求。
- no-auth： 服务器省略权威部份的记录，除非它们恰是被请求的记录。但是仍然会添加记录到附加部份。
- no-auth-recursive： 当请求中要求递归时（RD=1），与 no-auth 时相同，或者在没有要求递归时，与 no 时相同。

在回复存根客户端时，no-auth 和 no-auth-recursive 很有用，因为存根客户端通常忽略权威部份。no-auth-recursive 是设计用于混合模式服务器中，它会处理权威和递归请求。

缺省是 no-auth-recursive 。

glue-cache 当设置为 yes 时，在添加地址类型（A 和 AAAA）的粘合记录到授权到一个子区的 DNS 响应信息的附加部分时，会使用一个缓存来提高请求性能。

粘合缓存使用的内存与中授权数目成比例。缺省设置是 yes ，它提高了性能，代价是增加了这个区的内存使用量。要避免这样，将其设置为 no 。

minimal-any 如果设置为 yes ，当生成一个针对 ANY 类型且通过 UDP 传输的请求的肯定响应时，服务器将会仅仅回复请求名的一个资源记录集，以及覆盖它的 RRSIG 记录，如果存在 RRSIG 记录的情况，而不是回复这个名字的所有已知的 RRSIG。类似地，对类型 RRSIG 的请求将使用覆盖仅一个类型的 RRSIG 记录响应。这能够减少某些类型攻击流量的影响，而不伤害合法的客户端。（注意，然而，返回的资源记录集是在数据库中首先找到的；它不需要是最小可用的资源记录集。）此外，minimal-responses 对这些请求是打开的，这样不必要的记录就不会被添加到权威或附加部份。缺省是 no 。

notify 如果为 yes （缺省情况），服务器所负责的区发生改变时发出 DNS NOTIFY 消息，参见通知（Notify） 。消息发向区的 NS 记录（除 SOA 记录的 MNAME 字段所指示的主服务器之外）中所列出的服务器，以及在 also-notify 选项中列出的任何服务器。

如果为 master-only ，通知仅发给主区。如果为 explicit ，通知仅发给使用 also-notify 显式列出的服务器。如果为 no ，不发出通知。

notify 选项也可在 zone 语句中指定，在这种情况下，它将覆盖 options notify 语句。仅仅在它导致辅服务器崩溃时才需要关掉这个选项。

notify-to-soa 如果是 yes ，就不针对 SOA MNAME 检查 NS 资源记录集中的名字服务器。通常一个 NOTIFY 消息不会发送给 SOA MNAME（SOA 源），因为假设它包含终极主服务器的名字。然而，有时候在隐藏主服务器的配置中一个辅服务器被作为 SOA MNAME 列出，在这种情况下，应该设置终极主服务器仍然发送 NOTIFY 消息给 NS 资源记录集中列出的所有名字服务器。

recursion 如果为 yes ，并且一个 DNS 请求也要求递归，服务器就试图完成所有要求的工作以回答请求。如果递归关闭并且服务器不知道答案，它将会返回一个参考响应。缺省为 yes 。注意，设置 recursion no 不会阻止客户端从服务器的缓存获取数据；它仅仅阻止作为客户端请求结果的新的数据被缓存。缓存仍然会作为服务器内部操作的结果而被产生，例如 NOTIFY 地址查找。

request-nsid 如果为 yes ，在迭代解析时，所有发向权威服务器的请求都会带上一个空的 EDNS(0) NSID（名字服务器标识符）。如果权威服务器在其响应中返回了一个 NSID 选项，其内容将被记录到日志的 nsid 分类，级别为 info 。缺省是 no 。

request-sit 这个试验性选项已被废弃。

require-server-cookie 如果为 yes , 在发送一个完整响应给一个来自 cookie 感知的客户端的 UDP 请求之前，要求一个正确的服务器 cookie。如果存在一个错误的或不存在服务器 cookie，将发送 BADCOOKIE。

缺省为 no 。

那些希望测试 DNS COOKIE 客户端是否正确处理 BADCOOKIE 的用户，或者那些收到许多带有 DNS COOKIES 的伪造 DNS 请求的人应该将这个设置为 yes 。将这个选项设置为 yes ，将在遇到反射攻击时有效降低放大效果，因为 BADCOOKIE 响应将比一个完整的响应更小，同时也会要求一个合法的客户端随后的第二个请求使用新的、有效的 cookie。

answer-cookie 当设置为缺省值 yes 时，在合适的时候，将在回复客户端请求时发送 COOKIE EDNS 选项。如果设置为 no ，应答中将不会发送 COOKIE EDNS 选项。这个仅能在全局选项中设置，不能用于视图中。

answer-cookie no 的目的是作为一个临时测量手段，用于 named 与还不支持 DNS COOKIE 的其它服务器共享一个 IP 地址时。监听在同一个地址上的服务器不一致不会导致运行问题，但是出于更加谨慎，还是提供了一个关闭 COOKIE 响应的选项以使所有的服务器有同样的行为表现。DNS COOKIE 是一个重要的安全机制，不应该被关闭，除非绝对必要。

send-cookie 如果设置为 yes ，伴随请求会发送一个 COOKIE EDNS。如果解析器先前与服务器有过通信，将会发送先前交易中返回的 COOKIE。这是服务器用于决定解析器是否曾与之对话。一个发送正确 COOKIE 的解析器会被假设不是发送伪造源地址请求的旁路攻击者；因而，请求不太可能是一个反射/放大攻击的一部分，所以发送一个正确 COOKIE 选项的解析器就不会遭遇响应率限制（RRL）。不发送一个正确 COOKIE 选项的解析器可能被服务器通过 nocookie-udp-size 选项限制成只能收到更小的响应。

缺省为 yes 。

stale-answer-enable 如果为 yes , 开启当一个区的名字服务器没有答复且 stale-cache-enable 选项也开启时时返回“旧的”缓存答复。缺省是不返回旧答复。

旧答复也可以在运行时通过 rndc serve-stale on 或 rndc serve-stale off 开启或关闭；这些覆盖已配置的设置。rndc serve-stale reset 将设置恢复成 named.conf 中的设置。注意，如果旧答复被 rndc 关闭，它们不能通过重新装载或重新配置 named 来重新打开；它们只能通过 rndc serve-stale on 或者重新启动服务器来重新打开。

关于旧答复的信息是记录在 serve-stale 日志类别下。

stale-cache-enable 如果为 yes ，开启对“旧的”缓存答复的保留。缺省是 yes 。

nocookie-udp-size 这个设置发送给不带合法服务器 COOKIE 的请求的最大 UDP 响应大小。小于 128 的值将被静默地提高到 128。缺省值是 1232，但是 max-udp-size 选项可以更多地限

制响应大小。

sit-secret 这个试验性选项已被废弃。

cookie-algorithm 这个设置用于生成服务器 cookie 的算法。选项为 “aes”, “sha1” 或 “sha256”。如果加密库支持，缺省是 “aes”，否则是 “sha256”。

cookie-secret 如果设置，这是一个用于在一个任播集群内生成和验证 EDNS COOKIE 选项的共享密钥。如果未设置，系统将在启动时生成一个随机密钥。共享密钥被编码成一个十六进制字符串，对于 AES128 为 128 位，对于 SHA1 为 160 位，对于 SHA256 为 256 位。

如果指定了多个密钥，在 named.conf 中列出的第一个将用于生成新的服务器 cookie。其它的将仅用于验证返回的 cookie。

response-padding EDNS 填充选项的目的是当 DNS 请求通过一个加密通道发送时，通过减少包大小的变化来提升保密性。如果一个请求：

1. 包含一个 EDNS 填充选项，
2. 包含一个有效的服务器 cookie 或者使用 TCP，
3. 没有被 TSIG 或 SIG(0) 签名，以及
4. 来自一个地址与指定 ACL 匹配的客户端，

响应就使用一个 EDNS 填充选项被填充到 block-size 字节的倍数。如果这些条件不满足，响应就不会被填充。

如果 block-size 是 0 或者 ACL 是 none; ，这个特性会被关闭，不进行填充；这是缺省行为。如果 block-size 大于 512，将在日志中记录一条告警，并且这个值会被截断到 512。块大小通常期望是 2 的幂（例如，128），但这不是必须的。

trust-anchor-telemetry 这使 named 每天发送一次特定格式的请求给配置了信任锚的域，例如，通过 trust-anchors 或者 dnssec-validation auto 。

这些请求的请求名具有这样的格式 _ta-xxxx(-xxxx)(...).<domain> ，其中每个 “xxxx” 是一组四位十六进制数字，表示一个信任的 DNSSEC 密钥的密钥 ID。每个域的密钥在编码前以从最小到最大的顺序排序。请求类型为 NULL。

通过监控这些请求，区操作员能够发现哪些解析器被更新以信任一个新密钥；这会帮助他们决定何时能够安全地删除一个旧密钥。

缺省值是 yes 。

use-ixfr **这个选项被废除了**。要关掉到一个或多个服务器的 IXFR，参见server 语句定义和用法 中 provide-ixfr 选项的相关信息。也可参见增量区传送（IXFR） 。

provide-ixfr 参见server 语句定义和用法 中对 provide-ixfr 的描述。

request-ixfr 参见server 语句定义和用法 中对 request-ixfr 的描述。

request-expire 参见server 语句定义和用法 中对 request-expire 的描述。

match-mapped-addresses 如果为 yes ，一个映射到 IPv4 的 IPv6 地址将会与相关 IPv4 地址匹配的任何地址匹配表条目匹配。

引入这个选项是工作在某些操作系统中内核特有的行为，即导致例如区传送中的 IPv4 的 TCP 的连接被使用地址映射的 IPv6 套接字所接受。这会导致为 IPv4 所设计的地址匹配表匹配失败。然而，named 现在在内部解决这个问题。不鼓励使用这个选项。

ixfr-from-differences 如果为 yes 并且服务器从其区文件装载一个新版本的主区或者通过区传送方式接收一个新版本的辅区文件时，它将会比较新版本与先前版本并计算一个差集。差别就写入到区的日志文件中，这样变化就可以用增量区传送的方式向下游的辅服务器传送。

通过允许在非动态区中使用增量区传送，这个选项以增加主服务器的 CPU 和内存消耗的代价节约了带宽。在特殊情况下，如果一个区的新版本与前一个版本完全不一样，差别集将会有相当于旧版本和新版本之和的大小，并且服务器会临时申请内存以保持这个完整的差别集。

ixfr-from-differences 在 view 和 options 级别也接受 master （或 primary ）和 slave （或 secondary ），这将导致 ixfr-from-differences 分别在所有的主区或辅区被打开。缺省时对所有区都是关闭的。

注意：如果一个区打开了行内签名（译注：inline signing），用户为这个区所提供的 ixfr-from-differences 设置将被忽略。

multi-master 当一个区有多个主服务器并且其地址指向不同的机器时，应该设置此选项。如果为 yes ，named 在主服务器的序列号小于当前 named 的序列号时将不会记录。缺省为 no 。

auto-dnssec 为动态 DNS 配置的区可以使用这个选项来允许自动 DNSSEC 密钥管理变化级别。有三个可能的设置：

auto-dnssec allow; 允许无论何时用户发送 rndc sign zonename 命令时更新密钥和重签整个区。

auto-dnssec maintain; 包含上述功能，但也根据密钥时间元数据，按时自动调整区的 DNSSEC 密钥。（参见dnssec-keygen: DNSSEC 密钥生成工具 和dnssec-settime: 为一个 DNSSEC 密钥设置密钥定时元数据 ）。命令 rndc sign zonename 使得 named 从密钥仓库加载密钥并使用所有活跃密钥对区签名。rndc loadkeys zonename 使得 named 从密钥仓库加载密钥并调度未来发生的密钥维护事件，但它不会立即对整个区签名。注意：一旦密钥首次从一个区加载，就会定期搜索仓库以查找变化，而不考虑是否使用了 rndc loadkeys 。重新检查的间隔由 dnssec-loadkeys-interval 定义。

缺省设置为 auto-dnssec off 。

dnssec-enable 这个选项被废弃，已无效。

dnssec-validation 这个选项在 named 中打开 DNSSEC 验证。

如果设置为 auto ，DNSSEC 验证被开启，并使用一个缺省的 DNS 根区信任锚。

如果设置为 yes ，DNSSEC 验证被开启，但是必须使用一个 trust-anchors 语句（或者 managed-keys 或 trusted-keys 语句，这两个都已被废弃）手工配置一个信任锚；如果没有配置信任锚，将不会进行验证。

如果设置为 no ，DNSSEC 验证将被关闭。

缺省为 auto，除非 BIND 使用 configure --disable-auto-validation 构建，在此情况下缺省为 yes 。

缺省的根信任锚被存储在文件 bind.keys 中。如果 dnssec-validation 被设置为 auto , named 将在启动时装载这个密钥。这个文件的一份拷贝随着 BIND 9 被安装，是发布日期时的版本。如果根密钥过期，新的 bind.keys 拷贝可以从 https://www.isc.org/bind-keys 下载。

（为防止因未找到 bind.keys 而带来的问题，当前信任锚也被编译进 named 。不推荐依赖这个措施，无论如何，因为这在根密钥过期时，要求带一个新密钥重新编译 named。）

注解：named **只**从 bind.keys 中装载根密钥。这个文件不能用于存放其它区的密钥。如果未使用 dnssec-validation auto ，bind.keys 中的根密钥将被忽略。

无论何时解析器发出请求到 EDNS 兼容的服务器，它总是设置 DO 位表示它可以支持 DNSSEC 响应，即使 dnssec-validation 被关闭。

---

validate-except 这指定一个域名列表，在这些域名或之下 **不应**执行 DNSSEC 验证，无论这些名字或之上是否提供了信任锚。这个可以被用于，例如，当配置一个顶级域仅仅用于内部，这样缺乏在根区中对这个域的安全授权不会导致验证失败。（这类似设置一个否定信任锚，除了它是永久配置，而否定信任锚会在一段时间之后过期并被删除之外。）

dnssec-accept-expired 这在校验 DNSSEC 签名时接受过期的签名。缺省为 no 。把这个选项设置为 yes 有可能使 named 遭受重发攻击。

querylog 请求日志提供一个所有进入的请求和所有请求错误的完整日志。这提供了对服务器活动更多了解，但是对高负载服务器会有更显著的性能消耗。

querylog 选项指定是否在 named 启动的同时启动请求日志。如果未指定 querylog ，则是否记录请求日志由 logging 类别中 queries 中是否出现来决定。请求日志也可以在运行时使

用命令 rndc querylog on 来激活，或者使用 rndc querylog off 来关闭。

check-names 这个选项是用于限制主服务器文件和/或从网络中接收到的 DNS 响应中的域名的字符集和语法。根据使用场合的缺省值是不一样的。对于 primary 区，缺省为 fail 。对于 secondary 区，缺省是 warn 。对于从网络接收的回答（response ），缺省是 ignore 。

合法主机名和邮件域名的规则是由 RFC 952 和 RFC 821 派生出来的，并由 RFC 1123 所修改。

check-names 应用到 A，AAAA 和 MX 记录的属主名字段。它也应用到 NS，SOA，MX 和 SRV 记录 RDATA 字段的域名中。它也应用到 PTR 记录的 RDATA 字段中，在这里属主名表示其是一个主机的反向查找（属主名以 IN-ADDR.ARPA，IP6.ARPA 或者 IP6.INT 结尾）。

check-dup-records 这检查主区中，在 DNSSEC 中被当作不同的，但是在普通 DNS 中语义上相等的记录。缺省为 warn ，其它可能值为 fail 和 ignore 。

check-mx 这检查 MX 记录是否是指向一个 IP 地址。缺省值为 warn 。其它可能值为 fail 和 ignore 。

check-wildcard 这个选项用于检查非终结的通配符。使用非终结的通配符几乎总是由于缺乏对通配符匹配算法（RFC 1034 ）的理解。这个选项影响主区。缺省（yes ）是检查非终结的通配符并发出一个警告。

check-integrity 这对主区执行加载后的区完整性检查。这将检查 MX 和 SRV 记录指向地址（A 或 AAAA ）记录和因授权区而存在的粘着地址记录。对 MX 和 SRV 记录，仅对本区的主机名进行检查（对区外的主机名使用 named-checkzone ）。对 NS 记录，仅对区的顶端下的名字进行检查（对区外名字和粘着一致性检查使用 named-checkzone ）。缺省是 yes 。

使用 SPF 记录用于发送者策略框架的发布已被弃用，如同从使用 TXT 记录到 SPF 记录的迁移被放弃一样。如果已存在一个 SPF 记录，打开这个选项会检查一个 TXT 类型的发送者策略框架记录（以”v=spf1”开头）是否已经存在。如果 TXT 记录不存在，将发出警告，也可通过 check-spf 拟制。

check-mx-cname 如果设置了 check-integrity ，遇到指向 CNAME 记录的 MX 记录就会失败，警告或忽略。缺省是 warn 。

check-srv-cname 如果设置了 check-integrity ，遇到指向 CNAME 记录的 SRV 记录就会失败，警告或忽略。缺省是 warn 。

check-sibling 在执行完整性检查的同时，也检查兄弟线索的存在。缺省是 yes 。

check-spf 如果设置了 check-integrity 并且存在一条 SPF 记录，就检查是否存在一条 TXT 类型的发送者策略框架记录（以”v=spf1”开头）。缺省是 warn 。

zero-no-soa-ttl 如果为 yes ，当为 SOA 请求返回权威的否定响应时，将所返回的权威部份中的 SOA 记录的 TTL 值设置为零。缺省是 yes 。

zero-no-soa-ttl-cache 如果为 yes ，当缓存一个 SOA 请求的否定响应时，将其中 TTL 值设置为零。缺省是 no。

update-check-ksk 在设置为缺省值 yes 时，检查每个密钥的 KSK 位以决定为一个安全区生成 RRSIG 时如何使用密钥。

正常情况下，区签名密钥（即未置 KSK 位的密钥）用于对整个区签名，而密钥签名密钥（设置了 KSK 位的密钥）仅用于对区顶点的 DNSKEY 资源记录集签名。然而，如果这个选项被设置为 no ，KSK 位被忽略；KSK 被当成 ZSK，用于对整个区签名。这与 dnssec-signzone -z 命令行选项相似。

当这个选项被设置为 yes 时，在 DNSKEY 资源记录集中的每个算法至少有两个激活的密钥：每个算法至少有一个 KSK 和一个 ZSK。如果有任何一个算法的要求未被满足，这个选项对于这种算法将被忽略。

dnssec-dnskey-kskonly 当这个选项和 update-check-ksk 都被设置为 yes 时，只有密钥签名密钥（即设置了 KSK 位的密钥）会被用于对区顶点的 DNSKEY，CDNSKEY 和 CDS 资源记录集签名。区签名密钥（未置 KSK 位的密钥）会被用于对区剩下部份签名，不包括 DNSKEY 资源记录集。这与 dnssec-signzone -x 命令行选项相似。

缺省为 no 。如果 update-check-ksk 被设置为 no ，这个选项将被忽略。

try-tcp-refresh 如果为 yes ，如果 UDP 请求失败，试图使用 TCP 刷新区。缺省是 yes 。

dnssec-secure-to-insecure 这允许一个动态区从安全状态迁移到不安全状态（即，从签名到不签名），通过删除所有 DNSKEY 记录。缺省为 no 。如果设置为 yes ，并且如果区顶点处的 DNSKEY 资源记录集被删除了，则所有的 RRSIG 和 NSEC 记录也会从区中删除。

如果区使用了 NSEC3，也需要从区顶点处删除 NSEC3PARAM 资源记录集；这将导致所有相关的 NSEC3 记录被删除。（预计这一要求将在未来的版本中被去掉。）

注意如果一个区配置了 auto-dnssec maintain 并且在密钥仓库中仍然能够访问私钥，则在 named 下次启动时，区将会再次自动签名。

synth-from-dnssec 这个选项从 NSEC，NSEC3 和其它被使用 DNSSEC 证明是正确的资源记录集合成答复。缺省是 no ，但是它可能在将来的版本中变为 yes 。

注解：要使这个选项生效，必须开启 DNSSEC 验证。这个首次实现仅仅覆盖了从 NSEC 记录合成答复。从 NSEC3 合成是将来的计划。这个也由 synth-from-dnssec 控制。

转发

转发设施可以被用于在一些服务器上建立一个大的缓存，并减少到外部名字服务器的流量。它也允许这样的请求，服务器不能直接访问互联网，但也希望通过某种方式查找外部的名字。转发只发生在那些服务器不是其权威服务器并且其缓存没有答案的请求上。

forward 这个选项仅在转发服务器列表非空的情况下有意义。一个为 first 的值是其缺省值，将使服务器首先请求转发服务器；并且如果它不回答问题时，服务器再自行查找答案。如果指定为 only ，服务器只请求转发服务器。

forwarders 这个指定一个请求会被转发到的 IP 地址列表。缺省是空列表（不转发）。列表中的每个地址可以附带一个可选的端口号和/或 DSCP 值，还可以为整个列表设置一个缺省的端口号和 DSCP 值。

转发也可以按域来配置，允许全局转发选项被多种方式覆盖。可以设置某个特定的域使用不同的转发服务器，或者使用不同的 forward only/first 行为，或者全都不转发，参见zone 语句语法 。

双栈服务器

双栈服务器是服务器用于解决由于主机缺乏对 IPv4 或 IPv6 的支持而导致的可达性问题的最后手段。

dual-stack-servers 这指定机器的主机名或者地址，它可以通过 IPv4 和 IPv6 传输协议都能被访问到。如果使用主机名，服务器必须在其所使用的传输协议上能够被解析到。如果机器是双栈的，dual-stack-servers 参数就无效，除非对某个传输协议的访问被命令行所关闭（如 named -4 ）。

访问控制

对服务器的访问可以基于发出请求的系统的 IP 地址进行限制。参见地址匹配表 以获得如何指定 IP 地址列表的详细内容。

allow-notify 这个 ACL 指定哪些主机可以发送 NOTIFY 消息通知本服务器关于本服务器作为辅服务器的区的变化。这个仅适用于辅区（即类型 secondary 或 slave ）。

如果这个选项设置在 view 或 options 中，它全局应用于所有的辅区。如果设置在 zone 语句，会覆盖全局值。

如果未指定，缺省是仅处理来自区中配置在 masters 中的服务器的 NOTIFY 消息。allow-notify 可以用于扩展所允许的主机的名单，但不能缩小它。

allow-query 这指定允许哪些主机可以进行普通的 DNS 查询。allow-query 也可以在 zone 语句中指定，这时它会覆盖 options allow-query 语句。如果未指定，缺省是允许所有主机请求。

注解：allow-query-cache 用于指定对缓存的访问。

---

allow-query-on 这指定哪些本机地址可以接受普通的 DNS 问题。这使得这样的情况成为可能，例如，允许面向内部的接口接受请求而不允许面向外部的接口接受请求，而不需要知道内部的网络地址。

注意 allow-query-on 仅仅针对被 allow-query 所允许的请求才被检查。请求必须被两个 ACL 都允许，否则将被拒绝。

allow-query-on 也可以在 zone 语句中指定，这时，它会覆盖 options allow-query-on 语句。

如果未指定，缺省是允许在所有地址上的请求。

注解：allow-query-cache 用于指定对缓存的访问。

---

allow-query-cache 这指定允许哪些主机可以从缓存中获取答案。如果未设置 allow-recursion，Bind 顺序检查下列参数是否设置：allow-query-cache 和 allow-query （除非设置了 recursion no; ）。如果这些参数都没有设置，就使用缺省的（localnets; localhost;）。

allow-query-cache-on 这指定哪些本机地址可以发送来自缓存的响应，如果未设置 allow-query-cache-on ，就使用 allow-recursion-on ，如果设置了后者。否则，缺省允许缓存响应从任何地址发送。注意：在一个缓存响应被发送之前，allow-query-cache 和 allow-query-cache-on 都必须被满足；一个被其中之一阻止的客户端也会被另一个阻止。

allow-recursion 这指定允许哪些主机可以通过本服务器进行递归查询。BIND 以顺序：allow-query-cache 和 and allow-query 检查这些参数是否设置。如果这些参数都没有设置，就使用缺省的（localnets;localhost;）。

allow-recursion-on 这指定哪些本机地址可以接受递归请求。如果 allow-recursion-on 未设置，就使用 allow-query-cache-on ，如果后者被设置了；否则，缺省是允许在所有地址上的递归请求：任何允许发送递归请求的客户端可以将其发送到 named 监听的任何地址。注意：在递归被允许之前，allow-recursion 和 allow-recursion-on 都必须被满足；被其中之一阻止的客户端也会被另一个阻止。

allow-update 当为了一个主区而设置在 zone 语句中时，这指定允许哪些主机可以对这个区提交动态 DNS 更新。缺省是拒绝所有主机的动态更新。

注意，允许基于请求者 IP 地址的动态更新是不安全的；更详细的内容参见动态更新的安全 。

通常情况这个选项只应该设置在 zone 级。由于一个缺省值可以设置在 options 或 view 级并被区所继承，这可能导致某些区无意间允许更新。

allow-update-forwarding 当为了一个辅区而设置在 zone 语句中时，这指定允许哪些主机可以提交动态 DNS 更新，并将其转发给主区。缺省是 { none; } ，表示不执行更新转发操作。

要允许更新转发，在 zone 语句中指定 allow-update-forwarding { any; }; 。指定 { none; } 或 { any; } 之外的值通常是事与愿违的；对更新的访问控制应该是主服务器的责任，而不是辅服务器的责任。

注意打开一个辅服务器上的更新转发特性可能将主服务器暴露给攻击者，如果它们依赖不安全的基于 IP 地址的访问控制；更详细的内容参见动态更新的安全 。

通常情况这个选项只应该设置在 zone 级。由于一个缺省值可以设置在 options 或 view 级并被区所继承，这可能导致某些区无意间允许转发更新。

allow-v6-synthesis 本选项引入从 AAAA 到 A6 以及从 “半字节标记” 到二进制标记的平滑转换功能。然而，由于 A6 和二进制标记都废弃了，本选项也被废弃了。现在这个选项被忽略，并带有一些警告信息。

allow-transfer 这指定哪些主机允许从服务器接收区传送。allow-transfer 也可以在 zone 语句中指定，在这种情况下它会覆盖设置在 options 或 view 中的 allow-transfer 语句。如果未指定，缺省是允许所有主机进行传送。

blackhole 这指定一个地址列表，服务器将不会接受来自其中的请求或者用其解析一个请求。从这些地址来的请求将不会有任何响应。缺省是 none 。

keep-response-order 这指定一个地址清单，服务器向这些地址的 TCP 请求发送响应时将会以接收它们时同样的顺序。这关闭了对 TCP 请求的并行处理。缺省是 none 。

no-case-compress 这指定一个地址列表，其中地址要求响应使用大小写无关的压缩。当 named 需要与那些不遵守 RFC 1034 所要求的在匹配域名时使用大小写无关的名字压缩的客户端的通信时，可以使用这个 ACL。

如果未定义，ACL 缺省为 none： 大小写无关的压缩将用于所有客户端。如果定义了 ACL 并且与一个客户端匹配，在发送给那个客户端的响应中压缩域名时将忽略大小写。

这将导致稍稍小一点的响应：如果一个响应包含名字” example.com” 和 “example.COM”，大小写无关的压缩将把第二个当成一个副本。它也确保请求名的大小写与返回记录的属主名精确匹配，而不是匹配其文件中记录的大小写。这允许响应精确匹配请求，这是一些不正确使用大小写相关压缩的客户端所要求的。

大小写无关压缩 **总是**用在 AXFR 和 IXFR 响应上，而不管客户端是否匹配这个 ACL。

在有些环境中 named 不保持记录属主名的大小写：如果一个区文件中定义同一个名字的不同类型记录，但名字的大写字母使用不同（例如， “www.example.com/A” 和” WWW.EXAMPLE.COM/AAAA” ），那么所有对那个名字的响应将使用在区文件中用到的那个名字的 **第一个**版本。这个限制可能在一个将来的版本中被处理。然而，在资源记录的 rdata 部份所指定的域名（例如，NS，MX，CNAME 等记录类型）将总是保持其大小写，除非客户端匹配这个 ACL。

resolver-query-timeout 解析器在尝试解析一个递归请求时，在失败之前将花费的以毫秒计的时间。缺省值和最小值是 10000 ，最大值是 30000 。将其设置为 0 将会导致使用缺省值。

这个值最初以秒指定。小于或等于 300 的值将被当成秒，并在应用到上述限制之前被转换为毫秒。

接口

服务器回答请求的接口和端口可以由 listen-on 选项指定。listen-on 接受作为可选项的端口号和一个 IPv4 地址的 address_match_list 。（IPv6 地址被忽略，并在日志中记录一条警告。）服务器将监听在地址匹配表所允许的所有接口上。如果未指定端口，将使用 53 端口。

允许使用多个 listen-on 语句。例如，

```
listen-on { 5.6.7.8; };
listen-on port 1234 { !1.2.3.4; 1.2/16; };
```

使服务器监听在 IP 地址 5.6.7.8 的 53 端口，以及本机在网络 1.2 上但不是 1.2.3.4 的地址的 1234 端口上。

如果没有指定 listen-on ，服务器将会监听在所有 IPv4 接口的 53 端口。

listen-on-v6 选项指定服务器用来监听 IPv6 发来的请求所用的接口和端口。如果未指定，服务器将监听在所有 IPv6 接口的 53 端口。

可以使用多个 listen-on-v6 选项。例如，

```
listen-on-v6 { any; };
listen-on-v6 port 1234 { !2001:db8::/32; any; };
```

使服务器监听在任何 IPv6 地址（带有单个通配套接字）的 53 端口，以及任何不在 2001:db8::/32 前缀（对每个匹配的地址有单独的套接字）的 IPv6 地址的 1234 端口上。

要使服务器不监听在任何 IPv6 地址，使用

```
listen-on-v6 { none; };
```

请求地址

如果服务器不知道一个问题答案，它就会请求其它的名字服务器。query-source 指定这样的请求所使用的地址和端口。对通过 IPv6 发出的请求，有一个单独的 query-source-v6 选项。如果 address 为 * （星号）或被省略，就使用通配符 IP 地址（INADDR_ANY ）。

如果 port 为 * 或被省略，就从一个预先配置的范围中随机取出一个端口号，并用在每个请求中。端口范围是在 use-v4-udp-ports （对 IPv4）和 use-v6-udp-ports （对 IPv6）选项中指定，并分别排除 avoid-v4-udp-ports 和 avoid-v6-udp-ports 选项所指定的范围。

query-source 和 query-source-v6 的缺省选项为：

```
query-source address * port *;
query-source-v6 address * port *;
```

如果未指定 use-v4-udp-ports 或者 use-v6-udp-ports ， named 检查操作系统是否提供一个程序接口来提取系统缺省的临时端口的范围。如果有这样的一个接口，named 将使用相应的系统缺省范围；否则，它将使用自己的缺省值：

```
use-v4-udp-ports { range 1024 65535; };
use-v6-udp-ports { range 1024 65535; };
```

注解：确保这个范围足够大，以保证安全。一个合理的大小依赖于多个参数，但是我们通常推荐它至少包含 16384 个端口（14 位熵）。还要注意系统的缺省范围在使用时可能太小而不满足这个目的，这个范围甚至可能在 named 运行时发生变化；当 named 重新装载时，将会自动应用新的范围。鼓励显式地配置 use-v4-udp-ports 和 use-v6-udp-ports，这样范围就足够大并适度地独立于其它应用所使用的范围。

注解：named 所用的运行配置可能禁止使用某些端口。例如，UNIX 系统将不允许以一个非 root 权限运行的 named 使用小于 1024 的端口。如果这样的端口被包含在所指定的（或被检测的）请求端口集合中，相应的请求企图将会失败，从而导致解析失败或延期。因此，配置端口集合，使其能够安全地使用在期望的运行环境中是很重要的。

---

avoid-v4-udp-ports 和 avoid-v6-udp-ports 的缺省选项为：

```
avoid-v4-udp-ports {};
avoid-v6-udp-ports {};
```

注解： BIND 9.5.0 引入了 use-queryport-pool 来支持随机端口池，但是这个选项现在被废弃了，因为重复使用池中的同样端口可能不是足够安全。由于这个原因，通常强烈不鼓励为 query-source 或 query-source-v6 选项指定一个特定的端口；它隐含地关闭了使用随机端口号。

use-queryport-pool 这个选项被废弃。

queryport-pool-ports 这个选项被废弃。

queryport-pool-updateinterval 这个选项被废弃。

注解： query-source 选项中所指定的地址是同时用于 UDP 和 TCP 请求，但是端口只用于 UDP 请求。TCP 请求总是使用一个随机的非特权端口。

注解：Solaris 2.5.1 及之前的版本不支持设置 TCP 套接字的源地址。

注解：参见 transfer-source 和 notify-source 。

区传送

BIND 有机制来做区传送，并在系统进行区传送的地方对负载量设置限制。下列选项应用于区传送。

also-notify 这个选项定义一个全局的名字服务器的 IP 地址表，无论何时只要有更新的区被加载，都会向这个表中的服务器发出 NOTIFY 消息，还包括在区的 NS 记录中列出的服务器。这将有助于确保区拷贝尽快地同步到隐藏服务器。作为可选项，可以在每个 also-notify 地址指定一个端口，用于将通知消息发到缺省的 53 之外的端口。每个地址也可指定一个可选的 TSIG 密钥，使通知信息也被签名；这在向多个视图发送通知时很有用。作为显式地址的替代，也可以使用一个或多个命名的 masters 列表。

如果在 zone 语句中有一个 also-notify 表，它将会覆盖 options also-notify 语句。当一个

zone notify 语句被设置为 no，对于这个区，将不会向全局 also-notify 表中的 IP 地址发出 NOTIFY 消息。缺省是空表（无全局通知表）。

max-transfer-time-in 入向区传送如果运行超过这个分钟数将被终止。缺省为 120 分钟（2 小时）。最大值为 28 天（40320 分钟）。

max-transfer-idle-in 入向区传送在这个分钟数之内没有进展将被终止。缺省为 60 分钟（1 小时）。最大值为 28 天（40320 分钟）。

max-transfer-time-out 出向区传送如果运行超过这个分钟数将被终止。缺省为 120 分钟 (2 小时)。最大值为 28 天（40320 分钟）。

max-transfer-idle-out 出向区传送在这个分钟数之内没有进展将被终止。缺省为 60 分钟 (1 小时)。最大值为 28 天（40320 分钟）。

notify-rate 这指定在通常的区维护操作中，NOTIFY 请求被发送的频率。(NOTIFY 请求由于初始化区加载而受限于一个独立的速率限制；参见以下。) 缺省是每秒 20。可能的最低速率是 1 每秒；当设置为 0 时，它会被静默地提升为 1。

startup-notify-rate 这是当名字服务器初次启动，或者当区是新增到名字服务器中时，NOTIFY 请求被发送的频率。缺省是每秒 20。可能的最低速率是 1 每秒；当设置为 0 时，它会被静默地提升为 1。

serial-query-rate 辅服务器将会定期请求主服务器以发现区的序列号是否改变。每个这样的请求占用一定量的辅服务器网络带宽。为限制所使用的带宽数量，BIND 9 限制请求发送的速率。serial-query-rate 选项的值为一个整数，表示每秒发送的最大请求数。缺省为 20 每秒。可能的最低速率是 1 每秒；当设置为 0 时，它会被静默地提升为 1。

transfer-format 区传送可以使用两种不同的格式发送，one-answer 和 many-answers 。transfer-format 选项用于在主服务器上决定使用那种格式发送。one-answer 使用一个 DNS 消息传送一个资源记录。many-answers 将多个资源记录尽可能地打包在一个消息中。many-answers 更有效率；缺省是 many-answers 。通过使用 server 语句，transfer-format 可以被覆盖成以每个服务器为基础。

transfer-message-size 这是一个用于使用 TCP 进行区传送的 DNS 消息的未压缩大小的上限。如果一个消息增长超过这个大小，会增加额外的消息以完成区传送。(注意，无论如何，这是一个提示，不是一个硬限制；如果一个消息包含单一资源记录并且其 RDATA 超过了这个大小限制，将允许一个更大的消息以完成这个记录的传送。)

有效值介于 512 和 65535 字节之间，任何超过这个范围的值将被调整为范围内最接近的值。缺省是 20480 ，选择这个值是为了提高消息压缩：大多数这个大小的 DNS 消息将被压缩到 16536 字节以内。更大的消息不能被有效地压缩，因为 16536 是一个 DNS 消息内最大允许压缩偏移量指针。

这个选项主要用于服务器测试；设置为缺省值之外的值几乎没有收益。

transfers-in 这是可以同时运行的入向区传送的最大数目。缺省值是 10 。增加 transfers-in 可以加速辅区的同步，但是它也会增加本地系统的负载。

transfers-out 这是可以同时运行的出向区传送的最大数目。超过限制的区传送请求将会被拒绝。缺省值是 10 。

transfers-per-ns 这是可以同时运行的来自某个给定远程名字服务器的入向区传送的最大数目。缺省值是 2 。增加 transfers-per-ns 可以加速辅区的同步，但是它也会增加远程名字服务器的负载。通过在 server 语句中使用 transfers 子句，transfers-per-ns 可以被覆盖成基于每个服务器。

transfer-source transfer-source 决定哪个本地地址将会被绑定到 IPv4 的 TCP 连接，是用于服务器获取传入的区。它也决定更新区的请求和转发动态更新所使用的源 IPv4 地址，及可选的 UDP 端口，如果未指定，它缺省是一个系统控制的值，通常是“最接近”远端的接口的地址。这个地址必须出现在远端被传送区的 allow-transfer 选项中，如果指定了这个选项的话。这个语句为所有区设置 transfer-source ，但是可以通过在配置文件的 view 或 zone 块中包含 transfer-source 语句而被覆盖成以每个视图或每个区为基础。

注解: Solaris 2.5.1 及更早期的版本不支持设置 TCP 套接字的源地址。

---

transfer-source-v6 这个选项与 transfer-source 相似，除了区传送是使用 IPv6 之外。

alt-transfer-source 这个指示了一个可替换的传送源，如果在 transfer-source 中列出的源失效并且设置了 use-alt-transfer-source 。

注解: 要避免使用替换的传送源，正确设置 use-alt-transfer-source 并且不要依赖从第一个更新请求获得回复。

---

alt-transfer-source-v6 这个指示了一个可替换的传送源，如果在 transfer-source-v6 中列出的源失效并且设置了 use-alt-transfer-source 。

use-alt-transfer-source 这个指示是否使用可替换的传送源，如果使用视图，这个选项的缺省为 no ，否则其缺省为 yes 。

notify-source notify-source 决定哪个本地源地址，及可选的 UDP 端口，将用于发送 NOTIFY 消息。这个地址必须出现在辅服务器的 masters 区子句或 allow-notify 子句中。本语句为全部区设置 notify-source ，但是可以通过在配置文件的 zone 或 view 块中包含 notify-source 语

句而被覆盖成以每个区或每个视图为基础。

注解：Solaris 2.5.1 及更早期的版本不支持设置 TCP 套接字的源地址。

---

notify-source-v6 这个选项与 notify-source 相似，但是以 IPv6 地址发送通知消息。

UDP 端口列表

use-v4-udp-ports, avoid-v4-udp-ports, use-v6-udp-ports 和 avoid-v6-udp-ports 指定能或不能用于 UDP 消息源端口的 IPv4 和 IPv6UDP 端口的列表。关于如何决定可利用的端口，参见请求地址 。例如，使用下列配置：

```
use-v6-udp-ports { range 32768 65535; };
avoid-v6-udp-ports { 40000; range 50000 60000; };
```

从 named 发送 IPv6 消息的 UDP 端口将是下列范围中的一个：32768 到 39999，40001 到 49999，以及 60001 到 65535。

avoid-v4-udp-ports 和 avoid-v6-udp-ports 可以用来防止 named 随机选择到被防火墙阻止或被其它应用使用的端口；如果一个请求带有被防火墙所阻止的源端口，回答就不会经过防火墙，名字服务器不得不重新请求。注意：期望的范围也可以只使用 use-v4-udp-ports 和 use-v6-udp-ports 来表示，在这个意义上 avoid- 选项是冗余的；提供它们是为了向后兼容性和有可能简化端口设定。

操作系统资源限制

服务器使用的大多数系统资源是可以限制的。在指定资源限制时的值是可变的。例如，1G 可以使用 1073741824 来替代，以指定一吉字节的限制。unlimited 要求不限制使用，或可利用的最大数量。default 使用服务器启动时强制设定的限制。参见配置文件元素 中的关于 size_spec 的描述。

下列选项为名字服务器进程设置操作系统限制。某些操作系统不支持某些或任何限制；在这样的系统上，如果使用一个不支持的限制，将会发出一条警告。

coresize 这个设置内核转储的最大大小。缺省为 default 。

datasize 这个设置服务器可以使用的数据内存的最大数量。缺省为 default 。这是服务器内存用量的硬限制。如果服务器企图申请的内存超过这个限制，申请将会失败，将会导致服务器不能执行 DNS 服务。所以，这个选项很少作为一种限制内存总量的方式由服务器所使用，但是它可以用于在缺省值太小的情况下提升操作系统数据大小的限制。如果想限制服务器所使用的内存总量，使用 max-cache-size 和 recursive-clients 选项来替代。

files 这个设置服务器可以同时打开的文件的最大数目。缺省为 unlimited 。

stacksize 这个设置服务器可以使用的栈内存的最大数量。缺省为 default 。

服务器资源限制

下列选项设置了服务器对资源消耗的限额，这些限额是在服务器内部实现而不是由操作系统来实现的。

max-journal-size 这个设置每个日志文件（参见日志文件 ）的最大大小，表示成字节，或者如果后跟一个可选的单位后缀（‘k’，‘m’ 或 ‘g’），成千字节，兆字节或吉字节。当日志文件达到指定的大小时，其中一些最旧的事务会被自动删除。允许的最大值为 2G 字节。非常小的值会取整到 4096 字节。可以指定为 unlimited ，表示 2 吉字节。如果设置限制为 default 或者不设置值，日志可以增长到两倍区的大小。（存储较大的日志有点微小的好处。）

这个选项也可以基于每个区来设置。

max-records 这个设置一个区内允许的最大记录条目数。缺省是零，表示没有限制。

recursive-clients 这个设置服务器代表客户端执行的并行递归查询的最大数目（“硬限额”），缺省值是 1000 。因为每个递归查询使用相当大小的内存（大致是 20k 字节），在内存有限的主机上可以降低 recursive-clients 选项的值。

recursive-clients 为等待的递归客户端定义一个 “硬限额”：当超过这个数目的客户端在等待时，新进入的请求将不被接受，对每个新进入的请求，先前等待的请求将被丢弃。

也可以设置一个 “软限额”。当这个更低的限额被超过时，新进入的请求被接受，但是对每个新进入的请求，一个等待的请求将被丢弃。如果 recursive-clients 大于 1000，软限额被设置为 recursive-clients 减 100；否则，它被设置为 recursive-clients 的 90%。

tcp-clients 这个是服务器所能接受的并发的 TCP 客户端连接的最大数目。缺省是 150 。

clients-per-query; max-clients-per-query 这些设置允许同时进行的针对任何给定请求（<qname，qtype，qclass>）的递归客户端的初始数目（最小值）和最大数目，对任何超过这个数目的额外请求，服务器都将扔掉。named 会尝试自己调整这个值，并将改变记入日志。缺省值为 10 和 100。

这个值应该反映自它开始解析某个名字开始的一段时间内进来了多少个针对给定名字的请求。如果请求的数目超过这个值，named 将会假定它正在与一个无响应的区打交道，就会扔掉多余的请求。如果它在扔掉请求后收到响应，它会调高估值。如果没有变化，20 分钟后估值会被降低。

如果 clients-per-query 设为零，表示每个请求的客户端没有限制，没有请求被扔掉。

如果 max-clients-per-query 设为零，表示除了 recursive-clients 所设定的值外没有上限。

fetches-per-zone 这个设置对任何一个域的最大并发迭代请求数目，在这个数目内服务器会允许，超过这个数目，服务器或阻塞对这个区及之下数据的请求。这个值反映了在它所承担的解析时间内，对任意一个区，可以发送多少个正常的解析请求。它应该小于 recursive-clients 。

当多个客户端并发请求同样的名字和类型，客户端将被归并为同一次解析，直到达到 max-clients-per-query 限制，并且只发出一个迭代请求。然而，当不同的客户端并发请求不同的名字或类型，将会发出多个请求，而 max-clients-per-query 没有限制效果。

做为可选项，这个值可以跟随关键字 drop 或 fail ，指示针对一个区超过解析限额的请求是毫无响应地丢弃还是回应 SERVFAIL。缺省是 drop。

如果 fetches-per-zone 被设置为 0，对每个请求的解析数没有限制，不会丢弃请求。缺省是 0.

当前活跃解析的名单可以通过运行 rndc recursing 导出。名单包括对每个域活跃解析的数目和通过的请求数目以及作为 fetches-per-zone 限制结果而丢弃的请求数目。（注意：这些计数器不随时间而累积；无论何时一个域的活跃解析数目下降为 0，这个域的计数器将被删除，下一次一个解析发到这个域，就会重建计数器并被设置为 0.）

fetches-per-server 这个设置服务器允许发送给单一上游服务器的最大并发迭代请求数目，超过这个数目，服务器阻塞额外的请求。这个值反映了在它所承担的解析时间内，对任意一个服务器，可以发送多少个正常的解析请求。它应该小于 recursive-clients 。

做为可选项，这个值可以跟随关键字 drop 或 fail ，指示在一个区的所有权威服务器被发现超过每个服务器的限额时，请求是应该毫无响应地丢弃还是回应 SERVFAIL。缺省是 fail 。

如果 fetches-per-server 被设置为 0，对每个请求的解析数没有限制，不会丢弃请求。缺省是 0.

响应中的 fetches-per-server 限额是动态调整以检测拥塞。发给服务器的请求要么被响应，要么超时，会计算一个响应超时比率的指数加权移动平均值。如果当前平均超时比率上升到超过“高”限，那台服务器的 fetches-per-server 被减小。如果超时比率下降到“低”限以下，fetches-per-server 被增加。fetch-quota-params 选项可以用来调整这个计算的参数。

fetch-quota-params 这个设置用于动态增减 fetches-per-server 限额的参数。上述限额是在响应中检测拥塞。

第一个参数是一个整数，指示重新计算对每个服务器响应超时比率的移动平均值的频率。缺省值是 100，意谓着每 100 个请求被回答或超时之后我们重新计算平均比率。

剩余三个参数表示“低”限（缺省是 0.1 的超时比率），“高”限（缺省是 0.3 的比率）和移动平均值的折扣率（缺省是 0.7）。越大的折扣率在计算移动平均值时使最近的事件权重更大；

越小的折扣率使更远的事件权重更大，平滑掉超时比率中的短期噪点。这些参数都是定点数，精度为 1/100：小数点后至少 2 位是要保留的。

reserved-sockets 这个为 TCP，标准输入输出等设置保留的文件描述符个数。这个数需要足够大，以覆盖 named 监听的接口数加上用于提供出向 TCP 请求和入向区传送的 tcp-clients 。缺省是 512 。最小值是 128 ，最大值是 maxsockets (-S) 减 128 。这个选项在将来可能被去掉。

这个选项在 Windows 上几乎没有效果。

max-cache-size 这个设置用于服务器缓存的最大内存量，以字节或物理内存总量的百分比计。当缓存中的数据量达到这个限制时，服务器将使用一个基于 LRU 的策略使记录提前作废以免超过限制。关键字 unlimited ，或值 0，将使缓存大小没有限制；记录将在 TTL 过期时才将其从缓存中清除。任何小于 2MB 的正数都被忽略并重置为 2MB。在一个带多个视图的服务器上，这个限制分别应用到各个视图的缓存上。缺省为 90% 。在不支持检测物理内存的系统上，以% 表示的值回退为无限制。注意对物理内存的检测仅在启动时做一次，所以如果运行时发生物理内存数量变化，named 将不会调整缓存大小。

tcp-listen-queue 这个设置监听队列长度。缺省值和最小值都是 10。如果内核支持接受过滤器 “dataready” ，这个也可以控制有多少 TCP 连接可以在内核空间中排队以等待数据被接收。小于 10 的非零值将被静默地提升。也可以使用 0；在大多数平台上这将会将监听队列长度设置为一个系统定义的缺省值。

tcp-initial-timeout 这个设置服务器等待一个新的 TCP 连接上客户端的第一个消息的总时间（以 100 毫秒为单位）。缺省是 300（30 秒），最小值是 25（2.5 秒），最大值是 1200（2 分钟）。超过最大值或者低于最小值将被调整，并在日志中记录告警。（注意：这个值必须大于期望的往返延迟时间；否则没有客户端能够有足够的时间提交一个消息。）这个值可以在运行时使用 rndc tcp-timeouts 更改。

tcp-idle-timeout 这个设置当客户端不使用 EDNS TCP keepalive 选项时，服务器在关闭一个空闲的 TCP 连接之前等待的总时间（以 100 毫秒为单位）。缺省是 300（30 秒），最大值是 1200（2 分钟），最小值是 1（十分之一秒）。超过最大值或者低于最小值将被调整，并在日志中记录告警。参见 tcp-keepalive-timeout 查看客户端使用 EDNS TCP keepalive 选项。这个值可以在运行时使用 rndc tcp-timeouts 更改。

tcp-keepalive-timeout 这个设置当客户端使用 EDNS TCP keepalive 选项时，服务器在关闭一个空闲的 TCP 连接之前等待的总时间（以 100 毫秒为单位）。缺省是 300（30 秒），最大值是 65535（大约 1.8 小时），最小值是 1（十分之一秒）。超过最大值或者低于最小值将被调整，并在日志中记录告警。这个值可以大于 tcp-idle-timeout ，因为使用 EDNS TCP keepalive 选项的客户端期望使用 TCP 连接传送多个消息。这个值可以在运行时使用 rndc tcp-timeouts 更改。

tcp-advertised-timeout 这个设置服务器在包含 EDNS TCP keepalive 选项的响应中发送的超时

值(100 毫秒为单位)。它通知一个客户端应该保持会话打开的总时间。缺省是 300(30 秒), 最大值是 65535（大约 1.8 小时), 最小值是 0, 这表示客户端必须立即关闭 TCP 连接。通常这应该被设置为与 tcp-keepalive-timeout 相同的值。这个值可以在运行时使用 rndc tcp-timeouts 更改。

周期性任务的间隔

cleaning-interval 这个选项已被废弃。

heartbeat-interval 服务器将对所有标记为 dialup 的区执行区维护任务，无论本时间间隔是否过期。缺省是 60 分钟。1 天（1440 分钟）以内的值都是合理的。最大值是 28 天（40320 分钟)。如果设置为 0，就不会对这些区进行区维护。

interface-interval 服务器将每 interface-interval 分钟对网络接口列表进行扫描。缺省是 60 分钟。最大值是 28 天（40320 分钟） 。如果设置为 0，仅在配置文件装载后，或者当 automatic-interface-scan 被开启并且操作系统支持，才会进行接口扫描。扫描后，服务器将开始在新发现的端口监听请求（假定这些端口在 listen-on 配置中被允许)，并会停止在已经不存在的端口上的监听。为方便起见，可以用 TTL 风格的时间单位后缀来指定这个值。它也接受 ISO 8601 的持续时间格式。

sortlist 语句

一个 DNS 请求的响应由多个资源记录（Resource Record，RR）构成，它们形成了一个资源记录集（RRset)。名字服务器的通常返回中，资源记录集中的资源记录的顺序是不确定的（但可参见资源记录集排序 中的 rrset-order 语句)。客户端解析器代码应该适当地重新对资源记录排序，即，优先使用本地网络中的任何地址，然后是其它地址。然而，不是所有的解析器可以完成这项工作，或者被错误配置。当一个客户端使用一个本地服务器时, 排序可以基于客户端的地址在服务器上完成。这仅需要配置名字服务器，完全不需要配置客户端。

sortlist 语句(见下)接受一个 address_match_list , 并以一个特殊的方式解释它。每个 sortlist 中的顶层语句必须使用一个或两个元素在 address_match_list 中显式地指定其自身。每个顶层列表的第一个元素 (必须是一个 IP 地址, 一个 IP 前缀, 一个 ACL 名或者一个嵌套的 address_match_list ）与请求的源地址比较，直到匹配。当第一个元素中的地址一致时，第一个匹配的规则就会被选择。

一旦请求的源地址匹配，如果顶层语句只包含一个元素，实际与源地址匹配的元素用于选择响应中的地址，通过放在响应的开始处来实现。如果语句是两个元素的列表，第二个元素被解释成一个拓扑优先列表。每个顶层元素被指定一个距离，响应中具有最小距离的地址被放在响应的开始。

在下面的例子中,从主机的任何地址所收到的任何请求将会优先得到任何本地连接网络地址的响应。

下一个优先的是在 192.168.1/24 网络上的地址，然后是 192.168.2/24 或 192.168.3/24 网络，这两个网络之间没有优先关系。来自 192.168.1/24 网络上的主机的请求比 192.168.2/24 和 192.168.3/24 网络的地址优先。来自 192.168.4/24 或 192.168.5/24 网络的主机的请求将仅仅比主机直接连接的其它地址优先。

```
sortlist {
  // IF the local host
  // THEN first fit on the following nets
  { localhost;
  { localnets;
    192.168.1/24;
    { 192.168.2/24; 192.168.3/24; }; }; };
  // IF on class C 192.168.1 THEN use .1, or .2 or .3
  { 192.168.1/24;
  { 192.168.1/24;
    { 192.168.2/24; 192.168.3/24; }; }; };
  // IF on class C 192.168.2 THEN use .2, or .1 or .3
  { 192.168.2/24;
  { 192.168.2/24;
    { 192.168.1/24; 192.168.3/24; }; }; };
  // IF on class C 192.168.3 THEN use .3, or .1 or .2
  { 192.168.3/24;
  { 192.168.3/24;
    { 192.168.1/24; 192.168.2/24; }; }; };
  // IF .4 or .5 THEN prefer that net
  { { 192.168.4/24; 192.168.5/24; };
  };
};
```

下列例子给出本地主机和本地主机直接连接网络下的合理行为。发给来自本地主机请求的响应优先选用任何直接连接的网络。发给来自直接连接网络的其它主机请求的响应优先选用同一网络的地址。发给其它请求的响应不排序。

```
sortlist {
    { localhost; localnets; };
    { localnets; };
};
```

资源记录集排序

注解：在随后的请求中变换一个 DNS 响应中记录的顺序是一个众所周知的负载分担技术，某些警告（主要来自缓存）适用于这种情况，当单独使用时，通常使其成为负载平衡的次优选择。

rrset-order 语句允许在一个多记录响应中配置记录的顺序。参见：sortlist 语句.

一条 rrset-order 语句中的每条规则是如下定义的：

```
[class <class_name>] [type <type_name>] [name "<domain_name>"] order <ordering>
```

每条规则的缺省修饰符为：

- 如果没有指定 class，缺省是 ANY 。
- 如果没有指定 type，缺省是 ANY 。
- 如果没有指定 name，缺省是 * （星号）。

<domain_name> 只匹配名字自身，不匹配其任何子域。要使一条规则匹配一个给定名字的所有子域，必须使用一个通配名（*.<domain_name> ）。注意 *.<domain_name> **不会**匹配 <domain_name> 自身；要指定一个名字和其所有子域的资源记录集顺序，必须定义两条独立的规则：一条用于 <domain_name> 而另一条用于 *.<domain_name> 。

合法的 ordering 值是：

fixed 以区文件中定义的顺序返回记录。

注解：fixed 选项仅在 BIND 编译时带有 --enable-fixed-rrset 配置时才可用。

random 以随机的顺序返回记录。

cyclic 以循环轮转的顺序返回记录，每次请求轮转一条记录。

none 以从数据库中提取到的顺序返回记录。这个顺序是不确定的，但是会保持一致，只要数据库没有被修改。

使用的缺省资源记录集顺序取决于 named 所使用的配置文件中是否存在任何 rrset-order 语句：

- 如果配置文件中没有出现 rrset-order 语句，隐式的缺省是以 random 顺序返回所有记录。
- 如果配置文件中有任何 rrset-order 语句，但是与给定的资源记录集匹配的这些语句中没有顺序规则，对那些资源记录集而言，缺省的顺序是 none 。

注意如果在配置文件（在 options 和 view 两级中）出现了多条 rrset-order 语句，它们 **不会**组合；而是更具体的一个（view ）取代了不太具体的一个（options ）。

如果同一条 rrset-order 语句内的多条规则匹配了一个给定的资源记录集，将会应用第一条匹配的规则。

例如：

```
rrset-order {
  type A name "foo.isc.org" order random;
  type AAAA name "foo.isc.org" order cyclic;
  name "bar.isc.org" order fixed;
  name "*.bar.isc.org" order random;
  name "*.baz.isc.org" order cyclic;
};
```

由于上述的配置，将使用下列资源记录集顺序：

QNAME	QTYPE	RRset Order
foo.isc.org	A	random
foo.isc.org	AAAA	cyclic
foo.isc.org	TXT	none
sub.foo.isc.org	all	none
bar.isc.org	all	fixed
sub.bar.isc.org	all	random
baz.isc.org	all	none
sub.baz.isc.org	all	cyclic

调优

lame-ttl 这个设置为缓存一个跛服务器指示的秒数。0 关掉缓存。（这是 **不**推荐的。）缺省值为 600（10 分钟），最大值为 1800 （30 分钟）。

servfail-ttl 这个设置由于 DNSSEC 验证失败或者其它通用的服务器失败而导致 SERVFAIL 响应被缓存的秒数。如果设置为 0 ，关闭 SERVFAIL 缓存。如果一个请求带有 CD（Checking Disabled）位，SERVFAIL 缓存不被查询；这允许由于 DNSSEC 验证而失败的请求能够重试，而不是等待 SERVFAIL 的 TTL 过期。

最大值是 30 秒；更高的值将被静默地减小。缺省是 1 秒。

min-ncache-ttl 为了减少网络流量和提高性能，服务器存储否定响应。min-ncache-ttl 用于设置这些响应在服务器中最小持时间，单位为秒。为方便起见，TTL 风格的时间单位后缀也可用于指定这个值。它也接受 ISO 8601 的持续时间格式。

缺省的 min-ncache-ttl 值是 0 秒。min-ncache-ttl 不能超过 90 秒，如果设置为更大的值将被截断成 90 秒。

min-cache-ttl 这个设置服务器缓存普通（肯定）响应的最小时间，单位为秒。为方便起见，TTL 风格的时间单位后缀也可用于指定这个值。它也接受 ISO 8601 的持续时间格式。

缺省的 min-cache-ttl 值是 0 秒。min-cache-ttl 不能超过 90 秒，如果设置为更大的值将被截断成 90 秒。

max-ncache-ttl 为了减少网络流量和提高性能，服务器会存储否定响应。max-ncache-ttl 用于设置这些响应在服务器中最大保持时间，单位为秒。为方便起见，TTL 风格的时间单位后缀也可用于指定这个值。它也接受 ISO 8601 的持续时间格式。

缺省的 max-ncache-ttl 值为 10800 （3 小时）。max-ncache-ttl 不能超过 7 天，如果设置为更大的值，将会静默地被截为 7 天。

max-cache-ttl 这个设置服务器缓存普通（肯定）响应的最大时间，单位为秒。为方便起见，TTL 风格的时间单位后缀也可用于指定这个值。它也接受 ISO 8601 的持续时间格式。

缺省的 max-cache-ttl 值是 604800 （一周）。值为 0 可能导致所有的请求都返回 SERVFAIL，因为在解析过程中会丢失中间资源记录集（如 NS 和粘着 AAAA/A 记录）的缓存。

max-stale-ttl 如果在缓存中保留旧资源记录集被开启，并且返回旧缓存答复也被开启，max-stale-ttl 设置服务器保留那些超期记录的最大时间。这些超期记录在其服务器不可达时将作为旧记录响应。缺省是 12 小时周。允许的最小值是 1 秒；一个为 0 的值将被静默地更新为 1 秒。

要返回旧记录，必须通过配置选项 stale-cache-enable 将其保留在缓存中，并且必须开启返回所缓存的答复，要么在配置文件中使用 stale-answer-enable 选项，或者通过调用 rndc serve-stale on 。

当 stale-cache-enable 被设置为 no 时，设置 max-stale-ttl 没有效果，在这种情况下，max-stale-ttl 的值将为 0 。

resolver-nonbackoff-tries 这个指定在指数后退进行之前重试的次数。缺省是 3 。

resolver-retry-interval 这个设置以毫秒计的基本重试间隔。缺省是 800 。

sig-validity-interval 这个指定 DNSSEC 签名在未来失效前的天数，这个签名是作为动态更新（*动态更新（Dynamic Update）*）的结果自动生成的。有一个可选的第二字段指定在失效之前多长时间重新生成签名。如果未指定，就在基本间隔的 1/4 时重新生成签名。如果基本间隔

大于 7 天，第二字段以天数指定，否则就以小时数指定。缺省的基本间隔是 30 天，对应的重签名间隔是 7 又 1/2 天。最大值是 10 年（3660 天）。

签名的开始时间被无条件地设为当前时间之前一小时，以允许有限的时钟误差。

sig-validity-interval 可以为了 DNSKEY 记录被覆盖，通过设置 dnskey-sig-validity 。

sig-validity-interval 应该至少是 SOA 过期间隔的几倍，以允许在不同计时器和过期日期之间合理的互操作。

dnskey-sig-validity 这个指定在未来的一个天数，在这个时候，为 DNSKEY 资源记录集而自动生成的，且作为动态更新（动态更新（Dynamic Update））的结果的 DNSSEC 签名将会过期。如果设置为一个非零值，这将覆盖由 sig-validity-interval 所设置的值。缺省是零，表示使用 sig-validity-interval 。最大值是 3660 天（10 年），更大的值将被拒绝。

sig-signing-nodes 这个指定在使用一个新 DNSKEY 签名一个区时，在每个量中检查的节点的最大数目。缺省是 100 。

sig-signing-signatures 这个指定在使用一个新 DNSKEY 签名一个区时，签名数目的上限，如果超过将会终止对一个量的处理。缺省是 10 。

sig-signing-type 这个指定一个在生成签名状态记录时会用到的私有 RDATA 类型。缺省是 65534 。

一旦有了一个标准类型，预料这个参数在将来的版本中会被去掉。

签名状态记录是用于 named ，用来跟踪一个区签名进程的当前状态。即，它是否还存活或者已经完成。这些记录可以使用命令 rndc signing -list zone 检查。一旦 named 使用一个特定密钥完成对一个区的签名，与那个密钥相关的签名状态记录就可以通过执行 rndc signing -clear keyid/algorithm zone 删除。要删除一个区的所有已完成签名状态记录，使用 rndc signing -clear all zone 。

min-refresh-time; max-refresh-time; min-retry-time; max-retry-time 这些选项控制服务器在刷新一个区（请求 SOA 看是否有变化）或者重试失败的传送时的行为。通常是使用区的 SOA 值，直到一个硬编码的最大 24 周的过期值。然而，这些值是由主服务器设置的，只给辅服务器管理员一点点对内容的控制权。

这些选项允许管理员针对每个区，每个视图或者全局设置一个以秒计的最小和最大的刷新和重试时间。这些选项对辅区和存根区有效，将 SOA 的刷新和重试时间指定到特定的值上。

下列为相应的缺省值。min-refresh-time 300 秒，max-refresh-time 2419200 秒（4 周），min-retry-time 500 秒，max-retry-time 1209600 秒（2 周）。

edns-udp-size 这个设置被广告的以字节计的 EDNS UDP 缓存的最大大小，用以控制从权威服务器接收的作为递归响应的包的大小。有效值为从 512 到 4096；超出这个范围的值将被静默地

调整为范围内最接近的值。缺省值是 1232。

设置 edns-udp-size 为非缺省值的通常的原因是为了让 UDP 回答穿过坏的防火墙，这样的防火墙阻止碎包和（或）阻止大于 512 字节的 UDP DNS 包通过。

当 named 首次请求一个远程服务器时，它会广告一个 512 字节的 UDP 缓冲区，这样在首次试探中有最大的成功机会。

如果初始的请求，其带有 512 字节缓存的 EDNS 广告，成功后，named 将逐步在后继的请求中广告一个更大的缓存，直到响应超时或者达到 edns-udp-size 。

named 使用的缺省缓存大小为 512，1232，1432 和 4096，但不超过 edns-udp-size 。(选择值 1232 和 1432 是为了让一个 IPv4/IPv6 封装的 UDP 消息能够不被分片地在最小 MTU 的以太网和 IPv6 网络中发送。)

max-udp-size 这个设置 named 所发送的 EDNS UDP 消息的最大字节数。有效值为从 512 到 4096（超出这个范围的值将被静默地调整为范围内最接近的值）。缺省值是 1232。

这个值应用在一个服务器发出的响应；要设置请求中广告的缓存大小，参见 edns-udp-size 。

设置 max-udp-size 为非缺省值的通常的原因是为了让 UDP 回答穿过坏的防火墙，这样的防火墙阻止碎包和（或）阻止大于 512 字节的 UDP 包通过。这是独立于所广播的接收缓存（edns-udp-size ）。

这个值设置得越小，就会鼓励越多的 TCP 流量到达名字服务器。

masterfile-format 这个指定区文件的格式（参见附加文件格式 ）。缺省值是 text ，即标准的文本表示，除非对辅区，对后者缺省值为 raw 。text 之外的文件格式一般是用 named-compilezone 工具生成，或由 named 导出。

需要注意的是，如果一个非 text 格式的区文件被装载，named 将省略某些只对 text 格式文件才做的检查。特别的，check-names 检查不会应用到 raw 格式。这就意谓着 raw 格式的区文件必须以 named 配置文件中所指定的同样检查级别来生成。而且，map 格式的文件通过内存映射直接装入内存，只有最低限度的检查。

本语句为所有区设置 masterfile-format ，但是可以通过在配置文件中的 zone 或 view 块中包含 masterfile-format 语句而被覆盖成以每个区或每个视图为基础。

masterfile-style 这指定当 masterfile-format 为 text ，在导出时指定区文件的格式。在带有任何其它 masterfile-format 时，这个选项被忽略。

当设置为 relative 时，记录以多行格式输出，其属主名被表示成相对于一个共享的起点。当设置为 full ，记录以单行格式输出，并带有绝对的属主名。当一个区文件需要被一个脚本自动化处理时，full 格式是最适合的。relative 格式更适合阅读，更适合手工编辑一个区。缺省是 relative 。

max-recursion-depth 这个设置在提供递归请求服务时任意一个时刻允许的递归层级的最大数目。解析一个名字时可能要求查找一个名字服务器的地址，这会导致要求解析另外的名字，等等；如果间接查询的次数超过了这个值，递归请求会被终止并返回 SERVFAIL。缺省是 7.

max-recursion-queries 这个设置提供递归请求服务时可以发出的迭代查询的最大数目。如果需要发出更多的请求，递归请求会被终止并返回 SERVFAIL。缺省是 75。

notify-delay 这个设置在发送一个区的通知消息集之间的延迟，单位为秒。缺省是 5 秒。

所有区所发出的 NOTIFY 消息的整体速率是由 serial-query-rate 来控制。

max-rsa-exponent-size 这个设置在验证时能够接受的最大 RSA 指数的位数。有效值是从 35 到 4096 位。缺省值 0 也可以接受的，并且等效于 4096.

prefetch 当收到一个有缓存且即将过期的请求，named 可以立即从权威服务器刷新数据，确保缓存总有答复可用。

prefetch 指定”触发”TTL 值，在这个值将进行当前请求的预取：当一个低 TTL 值的缓存记录在查询过程中遇到，它将被刷新。有效的触发 TTL 值是 1 到 10 秒。超过 10 秒的值将被静默地减到 10。将一个触发 TTL 设置为零将关闭预取功能。缺省的触发 TTL 是 2 。

一个可选的第二参数指定”合格”TTL：对于一个适合预取的记录可接受的最小 **初始** TTL 值。合格 TTL 必须至少比触发 TTL 大六秒：如果不是这样，named 将静默地向大调整。缺省的合格 TTL 是 9 。

v6-bias 在决定要试探的下一个名字服务器时，这个指示 IPv6 的名字服务器会优先的毫秒数。缺省是 50 毫秒。

服务器内置信息区

服务器通过在 CHAOS 类中的伪顶级域 bind 下面的一些内置区提供了一些有用的诊断信息。这些区是 CHAOS 类中内置视图（参见view 语句语法 ）的一部份，与缺省的 IN 类视图是分开的。大多数全局配置选项（allow-query 等等）将会应用到这个视图，但是有一些是局部被覆盖的：notify，recursion 和 allow-new-zones 总是被设置为 no ，并且 rate-limit 被设置为允许每秒三个响应。

要关掉这些区，使用下面的选项，或者通过定义一个 CHAOS 类下的匹配所有客户端的显式视图来隐藏内置的 CHAOS 视图。

version 这是服务器在收到一个类型为 TXT ，类为 CHAOS ，名为 version.bind 的请求时应该报告的版本。缺省是这个服务器的真实版本号。指定 version none 关掉对这个请求的处理。

将 version 设置为任何值（包括 none ）将同时关闭对 authors.bind TXT CH 的请求。

hostname 这是服务器在收到一个类型为 TXT，类为 CHAOS，名为 hostname.bind 的请求时应该报告的主机名。缺省为运行名字服务器的主机名，是由 gethostname() 函数所查到的。这个请求的主要目的是判定在一组 anycast 服务器中是哪一台在实际回应请求。指定 hostname none; 关掉对这个请求的处理。

server-id 这是服务器在收到一个名字服务器标识符（Name Server Identifier, NSID）请求，或一个类型为 TXT，类为 CHAOS，名为 ID.SERVER 的请求时应该报告的 ID。这类请求的主要目的是判定在一组 anycast 服务器中是哪一台在实际回应请求。指定 server-id none; 关掉对这个请求的处理。指定 server-id hostname; 将使 named 使用由 gethostname() 函数所查到的主机名。缺省 server-id 是 none 。

内置空区

named 服务器有一些内置空区（仅含有 SOA 和 NS 记录）。这是通常情况下只应在本地响应而不应该发到互联网的根服务器上的请求。覆盖这些名字空间的官方服务器会对这些请求响应 NXDOMAIN。特别地，这些覆盖了来自于 RFC 1918，RFC 4193，RFC 5737 和 RFC 6598 。其中也包含 IPv6 本地地址（本地分配的），IPv6 链路本地地址，IPv6 环回地址和 IPv6 未知地址的反向名字空间。

服务器将试图决定是否一个内置区已经存在或者是激活的（被一个 forward-only 的转发定义所覆盖），并且在此情况不会创建一个空区。

当前的空区列表为：

- 10.IN-ADDR.ARPA
- 16.172.IN-ADDR.ARPA
- 17.172.IN-ADDR.ARPA
- 18.172.IN-ADDR.ARPA
- 19.172.IN-ADDR.ARPA
- 20.172.IN-ADDR.ARPA
- 21.172.IN-ADDR.ARPA
- 22.172.IN-ADDR.ARPA
- 23.172.IN-ADDR.ARPA
- 24.172.IN-ADDR.ARPA
- 25.172.IN-ADDR.ARPA

- 26.172.IN-ADDR.ARPA
- 27.172.IN-ADDR.ARPA
- 28.172.IN-ADDR.ARPA
- 29.172.IN-ADDR.ARPA
- 30.172.IN-ADDR.ARPA
- 31.172.IN-ADDR.ARPA
- 168.192.IN-ADDR.ARPA
- 64.100.IN-ADDR.ARPA
- 65.100.IN-ADDR.ARPA
- 66.100.IN-ADDR.ARPA
- 67.100.IN-ADDR.ARPA
- 68.100.IN-ADDR.ARPA
- 69.100.IN-ADDR.ARPA
- 70.100.IN-ADDR.ARPA
- 71.100.IN-ADDR.ARPA
- 72.100.IN-ADDR.ARPA
- 73.100.IN-ADDR.ARPA
- 74.100.IN-ADDR.ARPA
- 75.100.IN-ADDR.ARPA
- 76.100.IN-ADDR.ARPA
- 77.100.IN-ADDR.ARPA
- 78.100.IN-ADDR.ARPA
- 79.100.IN-ADDR.ARPA
- 80.100.IN-ADDR.ARPA
- 81.100.IN-ADDR.ARPA
- 82.100.IN-ADDR.ARPA
- 83.100.IN-ADDR.ARPA

- 84.100.IN-ADDR.ARPA
- 85.100.IN-ADDR.ARPA
- 86.100.IN-ADDR.ARPA
- 87.100.IN-ADDR.ARPA
- 88.100.IN-ADDR.ARPA
- 89.100.IN-ADDR.ARPA
- 90.100.IN-ADDR.ARPA
- 91.100.IN-ADDR.ARPA
- 92.100.IN-ADDR.ARPA
- 93.100.IN-ADDR.ARPA
- 94.100.IN-ADDR.ARPA
- 95.100.IN-ADDR.ARPA
- 96.100.IN-ADDR.ARPA
- 97.100.IN-ADDR.ARPA
- 98.100.IN-ADDR.ARPA
- 99.100.IN-ADDR.ARPA
- 100.100.IN-ADDR.ARPA
- 101.100.IN-ADDR.ARPA
- 102.100.IN-ADDR.ARPA
- 103.100.IN-ADDR.ARPA
- 104.100.IN-ADDR.ARPA
- 105.100.IN-ADDR.ARPA
- 106.100.IN-ADDR.ARPA
- 107.100.IN-ADDR.ARPA
- 108.100.IN-ADDR.ARPA
- 109.100.IN-ADDR.ARPA
- 110.100.IN-ADDR.ARPA

- 111.100.IN-ADDR.ARPA
- 112.100.IN-ADDR.ARPA
- 113.100.IN-ADDR.ARPA
- 114.100.IN-ADDR.ARPA
- 115.100.IN-ADDR.ARPA
- 116.100.IN-ADDR.ARPA
- 117.100.IN-ADDR.ARPA
- 118.100.IN-ADDR.ARPA
- 119.100.IN-ADDR.ARPA
- 120.100.IN-ADDR.ARPA
- 121.100.IN-ADDR.ARPA
- 122.100.IN-ADDR.ARPA
- 123.100.IN-ADDR.ARPA
- 124.100.IN-ADDR.ARPA
- 125.100.IN-ADDR.ARPA
- 126.100.IN-ADDR.ARPA
- 127.100.IN-ADDR.ARPA
- 0.IN-ADDR.ARPA
- 127.IN-ADDR.ARPA
- 254.169.IN-ADDR.ARPA
- 2.0.192.IN-ADDR.ARPA
- 100.51.198.IN-ADDR.ARPA
- 113.0.203.IN-ADDR.ARPA
- 255.255.255.255.IN-ADDR.ARPA
- 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA
- 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA
- 8.B.D.0.1.0.0.2.IP6.ARPA

- D.F.IP6.ARPA
- 8.E.F.IP6.ARPA
- 9.E.F.IP6.ARPA
- A.E.F.IP6.ARPA
- B.E.F.IP6.ARPA
- EMPTY.AS112.ARPA
- HOME.ARPA

可以在视图一级设置空区，并且只应用到 IN 类的视图。仅当没有在视图一级指定被禁止的空区时，被禁止的空区才从全局选项继承。要重载被禁止区的全局选项列表，需要在视图一级禁止根区，例如：

```
disable-empty-zone ".";
```

如果使用这里所列的地址范围，应该已经有反向区来覆盖所用的地址。在实际中，不会出现这样的情况，即许多请求发向基础的服务器来查找这个空间中的名字。在实际情况中有太多这样的情况，需要部署牺牲服务器来将查询负载从基础服务器引导出来。

注解：在这些区的真实的上级服务器中，应该禁止其下的所有空区。对真正的根服务器，这都是内建空区。这使它们能够返回树中对更深一级的引用。

---

empty-server 这个指定出现在空区的返回的 SOA 记录中的服务器名字。如果未指定，就使用区的名字。

empty-contact 这个指定出现在空区的返回的 SOA 记录中的联系人名字。如果未指定，就使用 “.”。

empty-zones-enable 这个打开或关闭所有的空区。缺省是打开的。

disable-empty-zone 这个关闭单独的空区。缺省没有空区是关闭的。这个选项可以多次指定。

内容过滤

BIND 9 提供了过滤来自外部 DNS 服务器的，在回答部份包含某种类型数据的 DNS 响应的能力。特别地，如果相关的 IPv4 或 IPv6 地址与 deny-answer-addresses 选项中的 address_match_list 相匹配，它可以拒绝地址（A 或者 AAAA）记录。它也可以拒绝 CNAME 或 DNAME 记录，如果 “别名”（即 CNAME 别名或由于 DNAME 而被替换的请求名）与 deny-answer-aliases 选项中的

namelist 匹配时，这里所指的“匹配”表示别名是 name_list 中一个元素的一个子域。如果伴随 except-from 指定了可选的 namelist ，请求名与列表匹配的记录也会被接受，而不考虑过滤设置。同样地，如果别名是相应区的一个子域，deny-answer-aliases 过滤器也不会生效；例如，即使给 deny-answer-aliases 指定了“example.com”，

```
www.example.com. CNAME xxx.example.com.
```

来自一个“example.com”服务器的响应也会被接受。

在 deny-answer-addresses 选项的 address_match_list 中，只有 ip_addr 和 ip_prefix 有意义；任何 key_id 将会被静默地忽略。

如果一个响应消息因为过滤而被拒绝，整个消息将会被丢弃并且不会缓存，还会返回一个 SERVFAIL 错误给客户端。

这个过滤器是为了阻止“DNS 重绑定攻击”，在这种攻击中，攻击者对查询一个攻击者所控制的域名的请求，返回一个用户自己网络内的 IP 地址或者一个用户自己域内的一个别名。一个幼稚的网页浏览器或者脚本可能无意会充当代理，会让攻击者访问局域网络中的内部节点，而这本不是外部所能够访问到的。关于这个攻击更详细的内容参见 https://dl.acm.org/doi/10.1145/1315245.1315298 处的论文。

例如，拥有一个域名“example.net”并且内部网络使用 192.0.2.0/24 的 IPv4 前缀，一个管理员可能设定如下规则：

```
deny-answer-addresses { 192.0.2.0/24; } except-from { "example.net"; };
deny-answer-aliases { "example.net"; };
```

如果一个外部攻击者让局域网内的一个网页浏览器查找“attacker.example.com”的 IPv4 地址，攻击者的 DNS 服务器会返回一个在回答部份中有像这样记录的响应：

```
attacker.example.com. A 192.0.2.1
```

由于这个记录（IPv4 地址）的 rdata 与所设定的前缀 192.0.2.0/24 匹配，这个响应将会被忽略。

另一方面，如果浏览器查找一个合法的内部 web 服务器“www.example.net”并且下列响应被返回给 BIND 9 服务器：

```
www.example.net. A 192.0.2.2
```

这个响应会被接受，因为属主名“www.example.net”与 except-from 元素的设置“example.net”匹配。

注意这个本身并非真正是对 DNS 的攻击。实际上，一个“外部的”名字映射到“内部的”IP 地址

或者来自一个视图点上的域名没有任何错误；它实际上可能被用作合法目的，例如调试。由于这个映射表是由正确的属主所提供的，在 DNS 中检测这个映射的目的是否合法是不可能的或者是没有意义的。“重绑定” 攻击首先必须由使用 DNS 的应用进行防护。然而，对一个大的站点，立即保护所有可能的应用是很困难的。提供这个过滤特征仅仅是为了对这种运行环境有所帮助；通常是不鼓励打开这个功能，除非没有其它的选择，并且攻击是一个现实的威胁。

如果在地址段 127.0.0.0/8 中使用这个选项，就需要特别小心。这些地址显然是 “内部的” 地址，但是许多应用传统上依赖 DNS 将某些名字映射到这样的地址。过滤掉虚假地包含这类地址的 DNS 记录可能会破坏这类应用。

响应策略区（RPZ）重写

BIND 9 包含了一个限制机制，它修改请求的 DNS 响应，类似于电子邮件中的反垃圾邮件 DNS 拒绝名单。响应可以被修改成某个域名不存在（NXDOMAIN），某个域名的 IP 地址不存在（NODATA），或者包含其它的 IP 地址或数据。

响应策略区在视图中的 response-policy 选项中命名，或者当视图中没有 response-policy 选项时在全局选项中命名。响应策略区是普通的 DNS 区，包含资源记录集，在允许时可以进行普通的查询。通常最好使用某些规则限制那些请求，如：allow-query { localhost; }; 。注意，使用了 masterfile-format map 的区不能用作策略区。

response-policy 选项可以支持多个策略区。为最大化性能，使用一个基数树来快速识别包含与当前查询匹配的触发器的响应策略区。这强制一个 response-policy 选项中策略区的数量上限为 64；超过那个数就是一个配置错误。

在响应策略区中编码的规则当其在访问控制 中被定义之后处理。来自不被允许访问本解析器的客户端的所有请求都将得到带一个 REFUSED 状态码的回复，而不管所配置的 RPZ 规则。

在 RPZ 记录中可以编码五种策略触发器。

RPZ-CLIENT-IP IP 记录由 DNS 客户端的 IP 地址触发。客户端 IP 地址触发器被编码到的记录中，其属主名是 rpz-client-ip 的子域名，它相对于策略区原点名字，并编码一个地址或地址块。IPv4 地址被表示为 prefixlength.B4.B3.B2.B1.rpz-client-ip 。IPv4 前缀长度必须在 1 和 32 之间。所有四个字节 - B4，B3，B2 和 B1 - 都必须出现。B4 是 IPv4 地址中的最低字节的十进制数值，如同在 IN-ADDR.ARPA 中一样。

IPv6 被编码成类似于标准的 IPv6 文本表示格式，prefixlength.W8.W7.W6.W5.W4.W3.W2.W1.rpz-client-ip 。W8，…，W1 中的每个都是一个一到四位的十六进制数，表示 IPv6 地址中的 16 位，如同 IPv6 地址的标准文本表示法，但是是反向表示的，如同 IN6.ARPA 中一样。（注意，这个 IPv6 地址表示是与 IP6.ARPA 中的每个 16 进制数字占据一个标记是不同

的。）所有的 8 个双字节都必须出现，除非一个连续值为零的双字节集合，后者被替换为 .zz. ，类似于标准 IPv6 文本编码中的双冒号（::）。IPv6 前缀长度必须在 1 和 128 之间。

QNAME QNAME 策略记录由请求中的请求名所触发，并且 CNAME 记录所指向的目标被解析用以生成响应。一个 QNAME 策略记录的属主名是与策略区相对的请求名。

RPZ-IP IP 触发器是一个响应中 ANSWER 部份中的一个 A 或 AAAA 记录中的 IP 地址。它们被编码成类似客户端 IP 触发器，除了作为 rpz-ip 的子域名之外。

RPZ-NSDNAME NSDNAME 触发器用权威服务器的名字去匹配请求名，一个请求名的父域，一个请求名的 CNAME 或者一个 CNAME 的父域。它们被编码成相对于 RPZ 原点的 rpz-nsdname 的子域名。NSIP 触发器匹配域名的 A 和 AAAA 资源记录集中的 IP 地址，而域名可以根据 NSDNAME 策略记录进行检查。nsdname-enable 子句为单一策略区或所有策略区关闭或打开 NSDNAME 触发器。

如果请求的名字的权威服务器还是未知，named 在应用一条 RPZ-NSDNAME 规则之前会为这个请求名递归地查找权威服务器。这会导致一个处理延迟。为了以精度为代价加快处理速度，可以使用 nsdname-wait-recurse 选项；当设置为 no ，RPZ-NSDNAME 规则仅在请求名的权威服务器已经被查找过并且被缓存时才会应用。如果请求名的权威服务器不在缓存中，RPZ-NSDNAME 规则会被忽略，但是请求名的权威服务器会在后端被查找，并且对随后的请求，这条规则会被应用。缺省是 yes ，意谓着 RPZ-NSDNAME 规则总是被应用，即使需要首先查找请求名的权威服务器。

RPZ-NSIP NSIP 触发器匹配权威服务器的 IP 地址。他们被编码成类似 IP 触发器，除了作为 rpz-nsip 的子域名之外。NSDNAME 和 NSIP 触发器仅在名字至少带有 min-ns-dots 个点时才被检查。min-ns-dots 的缺省值是 1，以排除顶级域。nsip-enable 子句为单一策略区或所有策略区关闭或打开 NSIP 触发器。

如果一个名字服务器的 IP 地址还是未知，named 将在应用一条 RPZ-NSIP 规则之前递归查找这个 IP 地址。这会导致处理延迟。要以精确的成本来加速处理，可以使用 nsip-wait-recurse 选项：当设置为 no 时，RPZ-NSIP 规则仅应用在一个名字服务器的 IP 地址已经被查找并缓存的情况。如果一个名字服务器的 IP 地址不在缓存中，RPZ-NSIP 规则将被忽略，但地址将在后台被查找，这条规则将应用到后续的请求上。缺省是 yes ，表示 RPZ-NSIP 规则应该总是应用，即使一个地址需要先查找。

查询响应将与所有响应策略区进行比对，所以两条或多条策略记录可以被一条响应所触发。由于 DNS 响应根据至多一条策略记录被重写，必须选择一条策略编码一个动作（除 DISABLED 策略之外）。以下列顺序为重写选择编码它们的触发器或记录：

1. 选择区中最先在 response-policy 选项中出现的触发器记录。
2. 在单个区中，按 CLIENT-IP，QNAME，IP，NSDNAME，NSIP 的顺序使用触发器。

3. 在 NSDNAME 触发器中，优先使用匹配 DNSSEC 顺序下最小名字的触发器。

4. 在 IP 或者 NSIP 触发器之间，优先选择具有最长前缀的触发器。

5. 在带有相同前缀长度的触发器之间，优先选择匹配最小 IP 地址的 IP 或 NSIP 触发器。

当一个响应的处理被重新开始解析 DNAME 或 CNAME 记录并且没有策略记录被触发，所有的响应策略区被重新针对 DNAME 或者 CNAME 名字和地址查找。

RPZ 记录集是除了 DNAME 或 DNSSEC 之外的任何类型的 DNS 记录，它编码了针对请求的动作或响应。任何策略可以于任何触发器一起使用。例如，TCP-only 策略通常与 client-IP 触发器一起使用，但它也可以与任意其它类型的触发器一起使用，并强制一个区中对属主名的响应使用 TCP。

PASSTHRU 策略由一个目标为 rpz-passthru 的 CNAME 所指定。它使响应不被重写，通常用于在 CIDR 块的策略中”挖出小洞”。

DROP 策略由一个目标为 rpz-drop 的 CNAME 所指定。它使响应被丢弃。不发送任何响应给 DNS 客户端。

TCP-Only “滑动”策略由一个目标为 rpz-tcp-only 的 CNAME 所指定。它修改 UDP 响应为短小的、被截断的 DNS 响应，要求 DNS 客户端使用 TCP 重试。它用于缓解分布式 DNS 反射攻击。

NXDOMAIN 未定义域名响应由一个目标为根域（.）的 CNAME 编码。

NODATA 空白资源记录集由目标为通配顶级域（*.）的 CNAME 指定。它重写响应为 NODATA 或 ANCOUNT=0。

Local Data 一个由普通 DNS 记录所组成的集合可以用于回答请求。请求不在集合中的记录类型将会得到 NODATA 应答。

本地数据的一个特殊形式是一条目标为一个通配符，如 *.example.com，的 CNAME。它通常被用作一个普通的 CNAME，在星号（*）被请求名替换之后。这个特殊形式被用于在围墙花园（原文：walled garden）内的权威 DNS 服务器上作请求日志。

在一个策略区中所有由全部单个记录指定的动作可以被 response-policy 选项中的 policy 字句所覆盖。一个使用由另一个组织所提供的策略区的组织可以使用这个机制来重定向域名到它自己的围墙花园。

GIVEN 占位符策略表示”不覆盖但是执行区中所指定的动作。”

DISABLED 测试覆盖策略导致策略区记录不做任何事情，但是记录在策略区未被关闭时可能做的事情。对 DNS 请求的应答会根据任何被触发的未被关闭的策略记录被重写（或不被重写）。关闭的策略区应放在最前面，因为如果一个更高优先级的触发器在前面时，它们通常都不会被记录。

PASSTHRU; DROP; TCP-Only; NXDOMAIN; NODATA 这些设置都覆盖相应的每记录策略。

CNAME domain 这个导致所有 RPZ 策略记录表现为好像它们是”cname domain”记录。

缺省时，在一个响应策略区中编码的动作仅应用到要求递归（RD=1）的请求。这个缺省可以在一个视图中使用一条 recursive-only no 子句为一个策略区或全部响应策略区而被改变。这个特性在下面的情况很有用，在一个 RFC 1918 云的内部和外部有同样的区文件，并使用 RPZ 来删除外部可见的名字服务器或视图上可能包含的 RFC 1918 值。

同样缺省的是，RPZ 动作仅应用于那些要么不要求 DNSSEC 元数据（DO=0），要么在原始区中没有请求名的 DNSSEC 记录的 DNS 请求中。这个缺省可以在一个视图中使用 break-dnssec yes 子句而对所有响应策略区被改变。在这种情况下，RPZ 动作被应用而不考虑 DNSSEC。子句选项的名字反映了被 RPZ 动作所重写的结果不能被验证的事实。

对一个 QNAME 或 Client-IP 触发器，不需要 DNS 记录；名字或者 IP 地址自身就已足够，所以原则上请求名不需要被递归解析。但是，不解析请求名可能泄漏使用了策略区重写和名字被列入一个策略区的事实给所列名字的服务器的操作者。为阻止这个信息泄漏，缺省时一个请求所需的任何递归查询都在任何策略触发器被考虑之前完成。因为列出的域名通常有较慢的权威服务器，这个行为可能花费大量时间。当递归查询不能改变一个非错误响应时，qname-wait-recurse yes 选项覆盖缺省并开启这个行为。这个选项不影响列在其它包含 IP，NSIP 和 NSDNAME 触发器的区之后的策略区中的 QNAME 或 client-IP 触发器，因为那些（触发器）可能依赖在递归解析中会被发现的 A，AAAA 和 NS 记录。它也不影响 DNSSEC 请求（DO=1），除非使用了 break-dnssec yes ，因为响应会依赖解析中是否发现了 RRSIG 记录。使用这个选项可能导致诸如出现 SERVFAIL 的错误响应被重写，由于没有递归查询被完成，以发现权威服务器中的问题。

dnsrps-enable yes 选项打开 DNS 响应策略服务（DNSRPS）接口，如果它已经使用 configure --enable-dnsrps 编译进了 named 。

dnsrps-options 块提供额外的 RPZ 配置设置，它会被传递给 DNSRPS 提供者库。在一个 dnsrps-options 串中的多个 DNSRPS 设置应当使用分号分开。DNSRPS 提供者库，librpz，被传递一个由 dnsrps-options 文本组成的配置字符串，连接到从 response-policy 语句所派生的设置。

注意: dnsrps-options 文本只能包含与具体 DNSRPS 提供者相关的配置设置。例如，来自 Farsight Security 的 DNSRPS 提供者使用的选项有 dnsrpzd-conf ， dnsrpzd-sock 和 dnzrpzd-args （关于这些选项的详细信息，参见 librpz 文档）。其它 RPZ 配置设置也可以被包含进 dnsrps-options ，但是如果 named 通过设置 dnsrps-enable 为“no”而切换到传统的 RPZ 时，这些选项都会被忽略。

一个被 RPZ 策略所修改的记录的 TTL 根据策略区中相关记录的 TTL 来设置。它被限制为一个最大值。max-policy-ttl 子句从其缺省值 5 改变最大值秒数。为方便起见，TTL 风格的时间单位后缀也可用于指定这个值。它也接受 ISO 8601 的持续时间格式。

例如，一个管理员可能使用这个选项语句：

```
response-policy { zone "badlist"; };
```

和这个区语句：

```
zone "badlist" {type master; file "master/badlist"; allow-query {none;}; };
```

以及这个区文件：

```
$TTL 1H
@               SOA LOCALHOST. named-mgr.example.com (1 1h 15m 30d 2h)
        NS  LOCALHOST.

; QNAME policy records.  There are no periods (.) after the owner names.
nxdomain.domain.com    CNAME   .           ; NXDOMAIN policy
*.nxdomain.domain.com   CNAME   .           ; NXDOMAIN policy
nodata.domain.com      CNAME   *.          ; NODATA policy
*.nodata.domain.com    CNAME   *.          ; NODATA policy
bad.domain.com         A       10.0.0.1    ; redirect to a walled garden
        AAAA    2001:2::1
bzone.domain.com       CNAME   garden.example.com.

; do not rewrite (PASSTHRU) OK.DOMAIN.COM
ok.domain.com          CNAME   rpz-passthru.

; redirect x.bzone.domain.com to x.bzone.domain.com.garden.example.com
*.bzone.domain.com     CNAME   *.garden.example.com.

; IP policy records that rewrite all responses containing A records in 127/8
;       except 127.0.0.1
8.0.0.0.127.rpz-ip     CNAME   .
32.1.0.0.127.rpz-ip    CNAME   rpz-passthru.

; NSDNAME and NSIP policy records
ns.domain.com.rpz-nsdname   CNAME   .
48.zz.2.2001.rpz-nsip       CNAME   .

; disapprove and approve some DNS clients
112.zz.2001.rpz-client-ip   CNAME   rpz-drop.
8.0.0.0.127.rpz-client-ip   CNAME   rpz-drop.
```

（下页继续）

(续上页)

```
; force some DNS clients and responses in the example.com zone to TCP
16.0.0.1.10.rpz-client-ip  CNAME  rpz-tcp-only.
example.com            CNAME  rpz-tcp-only.
*.example.com          CNAME  rpz-tcp-only.
```

RPZ 会影响服务器性能。每个已配置的响应策略区都会使一个请求被回答前要求服务器执行一到四次额外的数据库查找。例如，一个 DNS 服务器有四个策略区，每个有四种类型的响应触发器（QNAME，IP，NSIP 和 NSDNAME），会要求进行相当于一个没有响应策略区的普通 DNS 服务器 17 倍的数据库查找。一个 BIND9 服务器，带有足够内存和一个带 QNAME 和 IP 触发器的响应策略区可能比最大每秒请求数下降约 20%。一个有四个带 QNAME 和 IP 触发器的响应策略区的服务器可能比最大 QPS 下降约 50%。

由 RPZ 所重写的响应在 RPZRewrites 统计中被计数。

可以使用 log 子句来选择性关闭对某个特定响应策略区的重写日志。缺省时，所有重写都被记录到日志。

add-soa 选项控制是否将 RPZ 的 SOA 记录添加到用于从该区追踪变化的部份中。这个可以在单独的策略区级别或者在响应策略级别设置。缺省是 yes 。

对 RPZ 区的更新是异步处理的；如果存在多个更新，它们会被捆绑在一起。如果一个对 RPZ 区的更新（例如，通过 IXFR）在最近的更新之后还不到 min-update-interval 秒，这个改变就不会被执行，直到这个间隔时间之后。缺省是 60 秒。为方便起见，TTL 风格的时间单位后缀也可用于指定这个值。它也接受 ISO 8601 的持续时间格式。

响应比率限制

过多的几乎同样的 UDP **响应**可以通过在一个 options 或 view 语句中配置 rate-limit 子句来控制。这个机制使权威的 BIND 9 免于被用于放大反射拒绝服务（DoS）攻击。可以通过发送短小的、截断的（TC=1）响应来提供按比率限制的响应给位于一个伪造的、受攻击的 IP 地址范围中的合法客户端。合法的客户端对丢弃或截断响应的反应分别是使用 UDP 或 TCP 重试。

这个机制目的是用于权威 DNS 服务器。它也可以用于递归服务器，但会减慢诸如 SMTP 服务器（邮件接收者）和 HTTP 客户端（web 浏览器）这些反复请求同一个域的应用。在可能情况下，关闭“公开”递归服务器是更好的选择。

响应比率限制使用一个“信用（credit）”或“代价桶（token bucket）”方案。每个同一响应和客户端的组合有一个概念上的帐号，会每秒挣得一个特定数目的信用值。一个预期的的响应将会从其帐号上减一。当帐号是负值时，响应将被丢弃或者被截断。响应在一个缺省值为 15 秒的时间轮转窗

口中被跟踪，窗口大小可以使用 window 选项配置为从 1 到 3600 秒（1 小时）之间的任意值。帐号不能大于每秒限制值或者小于 window 乘以每秒限制值。当一类响应的指定信用值被设为 0，这些响应没有比率限制。

对比率限制中的 “同一响应（identical response）” 和 “DNS 客户端” 的概念是不能简单化理解的。所有到一个地址块的响应被统计为如同到一个单一地址。地址块的前缀长度由 ipv4-prefix-length（缺省 24）和 ipv6-prefix-length （缺省 56）指定。

所有针对一个有效域名（qname）和记录类型（qtype）的非空响应都是相同的，有一个由 responses-per-second 指定的限制 (缺省为 0 或没有限制)。所有针对一个有效域名，不管其请求类型的空（NODATA）响应也是相同的。NODATA 这一类响应被 nodata-per-second 所限制（缺省是 responses-per-second ）。对一个给定的有效域名的任何及所有未定义的子域的请求而产生的 NXDOMAIN 错误，不考虑请求类型，都是同一的。它们由 nxdomains-per-second 限制（缺省是 responses-per-second ）。这控制某些使用随机域名的攻击，但能够在期望许多合法 NXDOMAIN 响应的服务器上被放松或关闭（设置为 0），例如来自反垃圾拒绝名单。指向或授权到一个给定域名的服务器都是同一的并可以被 referrals-per-second 所限制（缺省是 responses-per-second ）。

由本地通配符产生的响应被计数并限制，如同对于其父域名一样。这使用 random.wild.example.com 控制泛洪。

所有导致 DNS 错误，例如 SERVFAIL 和 FORMERR，而不是 NXDOMAIN 的请求都是相同的，而不考虑请求名（qname）或记录类型（qtype）。这控制使用无效请求或使用陌生、缺陷权威服务器的攻击。缺省时，对错误的限制与 responses-per-second 值一致，但它也可以被独立地使用 errors-per-second 设置。

许多利用 DNS 的攻击涉及伪造源地址的 UDP 请求。比率限制阻止利用 BIND 9 由于对伪造源地址请求的响应而对一个网络进行泛洪攻击，但是可以让一个第三方阻塞对合法请求响应。有一个机制可以回复一些来自一个其地址被伪造而用于泛洪攻击的客户端的合法请求。设置 slip 为 2（其缺省值）将使每隔一个 UDP 请求在回复时带有一个短小的、截断的（TC=1）响应。“滑动的” 响应的小尺寸和低频率，就缺乏放大效应，使其对反射 DoS 攻击没有吸引力。slip 必须介于 0 到 10 之间。值为 0 表示不 “滑动”；由于比率限制而不发出被截断的响应，所有响应都被丢弃。值为 1 使每个响应都滑动；值为 2 到 10 之间使每 N 个响应一次。一些错误响应，包括 REFUSE 和 SERVFAIL，不能被替代成截断的响应，而是以 slip 比率返回响应。

（注意：从一个权威服务器丢弃响应可能减小一个第三方成功伪造响应到一个递归解析器的难度。针对伪造响应的最安全方法，对权威服务器的操作者而言是使用 DNSSEC 签名自己的区，对递归服务器的操作者而言是验证响应。当这不是一个可选项时，对更关心响应完整性而不是缓解泛洪攻击的操作者，可以考虑设置 slip 为 1，将使所有比率限制响应都被截断而不是被丢弃。这就减小了针对反射攻击的比率限制的效果。）

当大致的每秒请求率超过了 qps-scale 值，responses-per-second ，errors-per-second ，

nxdomains-per-second 和 all-per-second 的值就减小一个比率，即当前比率比 qps-scale 的值。这个特性可以在攻击时增强抵抗能力。例如，设置 qps-scale 250; responses-per-second 20; 并且一个包含 TCP 的所有 DNS 客户端的所有请求的总请求率为 1000 次请求/秒，这时的’有效响应/秒’限制变为 (250/1000)*20 或 5。通过 TCP 发送的响应不会被限制，但是会被计数，以计算每秒请求率。

DNS 客户端社区可以给出其自身的参数，或完全没有比率限制，即将 rate-limit 语句放入 view 语句中而不是 option 语句中。一个视图中的 rate-limit 语句是替代而不是增补了主配置中的 rate-limit 语句。在一个视图中的 DNS 客户端可以通过 exempt-clients 子句免除比率限制。

所有类型的 UDP 响应可以由 all-per-second 短句限制。这个比率限制不同于在一个 DNS 服务器上由 responses-per-second，errors-per-second 和 nxdomains-per-second 所提供的比率限制，后者通常都是 DNS 反射攻击的受害者所看不到的。除非攻击者伪造的请求与受害者的合法请求完全一致，受害者的请求是不受影响的。受一个 all-per-second 限制影响的响应总是被丢弃；slip 值没有任何效果。一个 all-per-second 限制至少是其它限制的 4 倍大，因为单一的 DNS 客户端通常爆发出合法的请求。例如，在考虑到进入的 SMTP/TCP/IP 连接时，单个邮件消息的接收者可能产生来自一个 SMTP 服务器的对 NS，PTR，A 和 AAAA 记录的请求。这个 SMTP 服务器在它考虑 SMTP 的 Mail From 命令时，可能需要额外的 NS，A，AAAA，MX，TXT 和 SPF 记录。Web 浏览器通常重复地解析在一页的 HTML <IMG> 标签中重复出现的同样名字。all-per-second 类似于防火墙所提供的比率限制，但通常不如后者。忽略 DNS 响应内容的攻击更像是对服务器本身的攻击。它们通常应该在 DNS 服务器花费资源建立 TCP 连接或分析 DNS 请求之前被丢弃，但是比率限制必须在 DNS 服务器见到请求之前完成。

用于跟踪请求和比率限制响应的表的最大大小，通过 max-table-size 设置。表中每个条目介于 40 到 80 字节之间。表中需要的条目数大致是每秒收到的请求数。缺省是 20,000。为减小冷启动时的表增长，min-table-size （缺省是 500）可以设置最小的表大小。打开 rate-limit 类日志以监控表的增长并获取关于选择表大小初始值和最大值的信息。

使用 log-only yes 测试比率限制参数而不会实际丢掉任何请求。

被比率限制所丢弃的响应被包含在 RateDropped 和 QryDropped 统计中。被比率限制所截断的响应被包含在 RateSlipped 和 RespTruncated 统计中。

named 通过两个方法支持 NXDOMAIN 重定向：

- 重定向区 (zone 语句语法)
- 重定向名字空间

在这两种方法中，当 named 收到一条 NXDOMAIN 响应，它检查一个独立的名字空间以查看 NXDOMAIN 响应是否应该被替换为替代的响应。

在一个重定向区（zone "." { type redirect; }; ），用于替换 NXDOMAIN 的数据保存在一个单独的

区中，它不是通常的名字空间的一部分。所有重定向信息都存放在这个区中；其中没有授权。

在一个重定向名字空间（option { nxdomain-redirect <suffix> }; ），用于替换 NXDOMAIN 的数据是普通名字空间的一部份，通过在原始请求名后添加指定的后缀来查找。这大致使所需的缓存增加一倍，以处理 NXDOMAIN 响应，因为原始的 NXDOMAIN 响应和替代数据（或者一个 NXDOMAIN 指示不存在替代者）都必须存储。

如果一个重定向区和一个重定向名字空间都被配置了，首先试探重定向区。

4.2.15 server 语句语法

```
server <netprefix> {
    bogus <boolean>;
    edns <boolean>;
    edns-udp-size <integer>;
    edns-version <integer>;
    keys <server_key>;
    max-udp-size <integer>;
    notify-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [
      dscp <integer> ];
    notify-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ]
      [ dscp <integer> ];
    padding <integer>;
    provide-ixfr <boolean>;
    query-source ( ( [ address ] ( <ipv4_address> | * ) [ port (
      <integer> | * ) ] ) | ( [ [ address ] ( <ipv4_address> | * ) ]
      port ( <integer> | * ) ) ) [ dscp <integer> ];
    query-source-v6 ( ( [ address ] ( <ipv6_address> | * ) [ port (
      <integer> | * ) ] ) | ( [ [ address ] ( <ipv6_address> | * ) ]
      port ( <integer> | * ) ) ) [ dscp <integer> ];
    request-expire <boolean>;
    request-ixfr <boolean>;
    request-nsid <boolean>;
    send-cookie <boolean>;
    tcp-keepalive <boolean>;
    tcp-only <boolean>;
    transfer-format ( many-answers | one-answer );
    transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [
      dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * )
```

（下页继续）

(续上页)

```
    ] [ dscp <integer> ];
  transfers <integer>;
};
```

4.2.16 server 语句定义和用法

server 语句定义与一个远程名字服务器相关的特性。如果指定一个前缀长度，就覆盖一个范围的服务器。只有最特定的 server 子句应用，不考虑在 named.conf 中的顺序。

server 语句可以出现在配置文件的顶级，也可以在一个 view 语句中。如果一个 view 语句中包含一个或多个 server 语句时，只有这些会应用到视图中，而任何顶级的都被忽略。如果一个视图不包含 server 语句，任何顶级 server 语句都作为缺省使用。

如果一个远程服务器发出错误的数据，将其标志为 bogus 而阻止再发请求给它。bogus 的缺省值是 no 。

provide-ixfr 子句决定本地服务器在作为主服务器时，当远程服务器，一个辅服务器请求时是否使用一个增量区传送进行应答。如果设为 yes ，无论何时都提供增量区传送。如果设为 no ，所有到远程服务器的区传送都不会使用增量方式。如果未设置，就使用视图或者全局选项块中的 provide-ixfr 选项值作为缺省值。

request-ixfr 子句决定本地服务器，作为一个辅服务器时，将向给定的远程服务器，即主服务器，请求增量区传送。如果未设置，就使用视图或者全局选项块中的 request-ixfr 选项值作为缺省值。它也可在区块中设置，如果在那里设置，它会对那个区覆盖全局和视图中的设置。

IXFR 请求如果发送到不支持 IXFR 的服务器上，将会自动回退为 AXFR。所以，不需要手工列出哪个服务器支持 IXFR 和哪个服务器不支持；全局缺省值 yes 将总能够工作。provide-ixfr 和 request-ixfr 子句的目的是即使主服务器和辅服务器都支持 IXFR 时也可以关掉它：例如，如果一个服务器在使用 IXFR 时，有许多 bug 并宕掉或者数据损坏。

request-expire 子句决定本地服务器在作为辅服务器时，是否请求 EDNS EXPIRE 值。EDNS EXPIRE 指示在区数据将要过期并需要被刷新之前剩余的时间。这用于一个辅服务器从另一个辅服务器传送一个区；当从主服务器传送时，过期计数器是使用 SOA 记录中的 EXPIRE 字段来设置。缺省是 yes 。

edns 子句决定在与远程服务器通信时，是否试图使用 EDNS。缺省是 yes 。

edns-udp-size 选项设置 EDNS UDP 的大小，这个值在 named 向远程服务器发出请求时广播出去。有效值是从 512 到 4096 字节；在这个范围之外的值将被静默地调整到范围内最接近的值。这个选项是用于你想要广播一个与你全局广播值不同的值到这个服务器时：例如，当远程站点有一个

防火墙阻止较大的响应时。（注意：当前，这对所有发送给这个服务器的包设置了一个 UDP 大小；named 将不会偏离这个值。这与 options 或者 view 语句中的 edns-udp-size 的行为方式有所不同，后者指定一个最大值。在将来的版本中，server 语句的行为可能与 options/view 的行为一致。）

edns-version 选项设置解析器能够发送给服务器的最大 EDNS VERSION。实际发送的 EDNS 版本仍然服从通常的 EDNS 版本协商规则（参见 RFC 6891 ），服务器所支持的最大 EDNS 版本，以及其它线索指示的一个更低版本应被发送。这个选项的意图是用于一个远程服务器对一个给定的 EDNS 版本或更高版本反应错误时；它应该被设置为远程服务器为外界所知的能够支持的最高版本。有效值是 0 到 255；更大的值将被静默地调整。这个选项在比 0 更高的 EDNS 版本被使用之前将不需要。

max-udp-size 选项设置 named 发出的 EDNS UDP 消息的最大大小。有效值是从 512 到 4096 字节；在这个范围之外的值将被静默地调整。这个选项是用于当有防火墙会阻止来自 named 的大回复包时。

padding 选项为发出的消息增加 EDNS 填充选项，增加包的大小为一批特定的块大小。有效块大小范围从 0（缺省值，它关闭使用 ENS 填充）到 512 字节。更大的值将被缩减成 512，并在日志中记录告警。注意：这个选项当前不兼容于 TSIG 或 SIG(0)，因为包含填充的 EDNS OPT 记录必须被添加到已经被签名的包中。

tcp-only 选项设置传输协议为 TCP。缺省是使用 UDP 传输，并仅在收到一个截断响应时回退到 TCP。

tcp-keepalive 选项添加 EDNS TCP keepalive 到经过 TCP 发送的消息。注意当前响应中的空闲超时被忽略。

服务器支持两种区传送方法。第一种，one-answer ，使用一个 DNS 消息传送资源记录。many-answers 将尽可能多的资源记录封装在一个消息中，它更有效率，可以通过 transfer-format 选项为服务器指定使用哪种方法；如果未指定，使用 options 选项中所指定的 transfer-format 。

transfers 用于限制从指定的服务器进来的并发区传送的数目。如果没有设定 transfers 子句，就根据 transfers-per-ns 选项设定限制。

keys 子句标识一个由 key 语句所定义的 key_id , 在与远程服务器通讯时，用于事务安全（参见TSIG ）。当一个请求发向远程服务器时，就使用这里所指定的 key 来生成一个请求签名并将其添加到消息尾部。一个源自远程服务器的请求不要求被这个 key 签名。

当前只支持每个服务器一个 key。

transfer-source 和 transfer-source-v6 子句分别指定用于同远程服务器进行区传送的 IPv4 和 IPv6 源地址。对一个 IPv4 远程服务器，仅能指定 transfer-source 。类似的，对一个 IPv6 远程服务器，仅能指定 transfer-source-v6 。更多细节，参见区传送 中对 transfer-source 和 transfer-source-v6 的描述。

notify-source 和 notify-source-v6 子句分别指定用于向远程服务器发送通知消息的 IPv4 和 IPv6 源地址。对一个 IPv4 远程服务器，仅能指定 notify-source 。类似的，对一个 IPv6 远程服务器，仅能指定 notify-source-v6 。

query-source 和 query-source-v6 子句分别指定用于向远程服务器发送请求的 IPv4 和 IPv6 源地址。对一个 IPv4 远程服务器，仅能指定 query-source 。类似的，对一个 IPv6 远程服务器，仅能指定 query-source-v6 。

request-nsid 子句决定本地服务器在发送请求给这个服务器时是否需要添加一个 NSID EDNS 选项。这个覆盖在视图级和 option 级的 request-nsid 设置。

send-cookie 子句决定本地服务器在发送到其它服务器的请求中是否要添加一个 COOKIE EDNS 选项。这覆盖视图或全局选项中的 request-cookie 设置。named 服务器可以决定远程服务器不支持 COOKIE 并且不在请求中添加一个 COOKIE EDNS 选项。

4.2.17 statistics-channels 语句语法

```
statistics-channels {
    inet ( <ipv4_address> | <ipv6_address> |
        * ) [ port ( <integer> | * ) ] [
        allow { <address_match_element>; ...
        } ];
};
```

4.2.18 statistics-channels 语句定义和用法

statistics-channels 语句声明了系统管理员用来获取名字服务器统计信息的通信通道。

这个语句是打算在将来能够灵活地支持多个通信协议，但是当前只支持 HTTP 访问。它要求带有 libxml2 和/或 json-c（也被称为 libjson0）编译 BIND 9；即使在编译时没有提供库，statistics-channels 语句也被接受，但是任何 HTTP 访问都会失败并返回一个错误。

一个 inet 控制通道是一个监听在特定 ip_addr ，可以是一个 IPv4 或 IPv6 地址，上的特定 ip_port 的 TCP 套接字，一个为 * （星号）的 ip_addr 将被解释为 IPv4 通配地址；到系统的任何 IPv4 地址的连接都会被接受。要监听在 IPv6 的通配地址，使用为 :: 的 ip_addr 。

如果没有指定端口，端口 80 用于 HTTP 通道。星号（* ）不能用作 ip_port 。

打开一个统计通道的企图被可选的 allow 子句所限制。到统计通道的连接基于 address_match_list 而允许。如果没有提供 allow 子句，named 接受任何地址来的连接企图；由于统计可能包含敏感

的内部信息，强烈推荐正确地限制连接源。

如果没有提供 statistics-channels 语句，named 将不会打开任何通信通道。

统计以不同格式和视图提供，依赖于访问它们所使用的 URI。例如，如果配置了统计通道并监听在 127.0.0.1 的 8888 端口，就可以以 XML 格式在 http://127.0.0.1:8888/或 http://127.0.0.1:8888/xml 访问统计。引入一个 CSS 文件，在使用一个样式表兼容的浏览器查看时，可以将 XML 统计转换格式为表格，在使用一个支持 javascript 的浏览器查看时，可以使用 Google Charts API 将其转换格式为图表和图形。

统计的单独子项可以在下列位置查看 http://127.0.0.1:8888/xml/v3/status（服务器运行时间和上次修改配置时间），http://127.0.0.1:8888/xml/v3/server（服务器和解析器统计），http://127.0.0.1:8888/xml/v3/zones（区统计），http://127.0.0.1:8888/xml/v3/net（网络状态和套接字统计），http://127.0.0.1:8888/xml/v3/mem（内存管理统计），http://127.0.0.1:8888/xml/v3/tasks（任务管理统计）和 http://127.0.0.1:8888/xml/v3/traffic（流量大小）。

也可以 JSON 格式读取完整的统计集合 http://127.0.0.1:8888/json，在下列位置查看单独子项 http://127.0.0.1:8888/json/v1/status （服务器运行时间和上次修改配置时间），http://127.0.0.1:8888/json/v1/server（服务器和解析器统计），http://127.0.0.1:8888/json/v1/zones（区统计），http://127.0.0.1:8888/json/v1/net（网络状态和套接字统计），http://127.0.0.1:8888/json/v1/mem（内存管理统计），http://127.0.0.1:8888/json/v1/tasks（任务管理统计）和 http://127.0.0.1:8888/json/v1/traffic（流量大小）。

4.2.19 trust-anchors 语句语法

```
trust-anchors { <string> ( static-key |
  initial-key | static-ds | initial-ds )
  <integer> <integer> <integer>
  <quoted_string>; ... };
```

4.2.20 dnssec-keys 语句定义和用法

trust-anchors 语句定义 DNSSEC 信任锚。DNSSEC 在DNSSEC 中描述。

当一个非权威区的公钥或公钥摘需要公开，但又不能通过 DNS 安全地获得，要么因为它是 DNS 根区，要么因为它的父区未签名，这时，就定义一个信任锚。一旦一个密钥或一个摘要被配置成一个信任锚，它就被当作已验证并被证实为安全。

解析器试图对一个已配置信任锚的子域中的所有 DNS 数据进行 DNSSEC 验证。在指定名字之下的验证可以使用 rndc nta 暂时关闭，或使用 validate-except 选项永久关闭。

所有 trust-anchors 中列出的密钥，和对应的区，都被认为是存在的，并与父区怎么说无关。只有配置在信任锚中的密钥被用于验证对应名字的 DNSKEY 资源记录集。而不会用到父区的 DS 资源记录集。

trust-anchors 可以被设置在 named.conf 的顶层或者在一个视图内。如果它在两个地方都被设置了，它们都会被添加：在顶层定义的密钥被所有的视图所继承，但是在一个视图内部定义的密钥只能用于那个视图。

trust-anchors 语句可以包含多个信任锚条目，每个条目由一个域名，一个“锚类型”关键字用来指示信任锚的格式，以及密钥或摘要数据所组成。

如果锚类型是 static-key 或者 initial-key，那它后面会跟随密钥标志，协议，算法和 Base64 表示的公钥数据。这与 DNSKEY 记录的文本表示相同。密钥数据中的空格、制表符、换行和回车字符都被忽略，所以配置可以被分割为多行。

如果锚类型是 static-ds 或者 initial-ds，那它后面会跟随密钥标签，算法，摘要类型和十六进制表示的密钥摘要。这与 DS 记录的文本表示相同。空格、制表符、换行和回车字符都被忽略。

使用 static-key 或 static-ds 配置的信任锚是不可变的，而使用 initial-key 或 initial-ds 配置的密钥可以保持自动更新，而无需解析器操作员的干预。（static-key 中的密钥与使用已废弃的 trusted-keys 语句配置的密钥是一致的。）

例如，假设一个区的密钥签名密钥被攻破了，区的所有者必须撤销并替代这个密钥。拥有由 static-key 或者 static-ds 所配置的初始密钥的解析器就不能再验证这个区的数据；它的回答会带有一个 SERVFAIL 响应码。这会一直持续到解析器管理员使用新的密钥更新 trust-anchors 语句。

然而，如果信任锚是以 initial-key 或 initial-ds 配置的，区的所有者可以预先给他的区增加一个“备用的”密钥。named 将会存储这个备用的密钥，并在原始的密钥被撤销时，named 将能够平滑地过渡到新的密钥。它也能够识别旧密钥已经被撤销，并停止使用旧密钥验证回答，最小化被攻破的密钥所能够带来的损害。这是用于保持 ICANN 根 DNSSEC 密钥为最新的流程。

鉴于 static-key 和 static-ds 信任锚持续受信任直到从 named.conf 中删除，一个 initial-key 或 initial-ds 只被信任 **一次**：只要它一被装载到被管理密钥数据库中就开始 RFC 5011 密钥维护进程。

对同一个域名，不能将静态与初始信任锚混合使用。

当 named 第一次在 named.conf 中配置有 initial-key 或 initial-ds 的情况下运行时，它直接从区顶点取得 DNSKEY 资源记录集，并使用 trust-anchors 中所指定的信任锚验证它。如果这个 DNSKEY 资源记录集被一个与这个信任锚相匹配的密钥验证是有效签名的，它就被用作一个新的被管理密钥数据库的基础。

从那个点开始，无论 named 何时运行，只要它发现了出现在 trust-anchors 中的 initial-key 或 initial-ds，检查并确认针对指定域的 RFC 5011 密钥维护已经被初始化，如果是，它只是简单地继

续。在 trust-anchors 语句中所指定的密钥并不用于校验回答；它被储存在被管理的密钥数据库中的一个或多个密钥所替代。

在一个 initial-key 或 initial-ds 信任锚从 trust-anchors 语句中 **删除**（或者改变成一个 static-key 或 static-ds ）之后下一次运行 named 时，对应的区将会从被管理密钥数据库中删除，并且 RFC 5011 密钥维护将不再用于这个域。

在当前实现中，被管理密钥数据库是以一个 master-format （主区格式）区文件存放的。

在没有使用视图的服务器上，这个文件被命名为 managed-keys.bind 。当使用了视图时，对每个视图有一个独立的被管理密钥数据库；文件名是视图名（或者，如果一个视图名包含了使其成为非法文件名的字符，则是视图名的一个 hash），后跟后缀 .mkeys 。

当密钥数据库被改变时，区就被更新。与其它任何动态区一样，改变会被写入到一个日志文件中，例如，managed-keys.bind.jnl 或 internal.mkeys.jnl 。随后改变会尽快被提交到主文件；这个通常发生在 30 秒之内。所以，无论何时 named 采用了自动密钥管理，区文件和日志文件将会存在在工作目录中。（因为这个以及其它原因，工作目录应该总是 named 可写的。）

如果 dnssec-validation 选项被设为 auto ，named 会自动为根区初始化一个 initial-key 。用于初始化密钥维护进程的密钥存储在 bind.keys ；这个文件的位置可以使用 bindkeys-file 选项覆盖。如果没有找到 bind.keys ，作为回退，初始化密钥也被直接编译进 named 。

```
dnssec-policy <string> {
    dnskey-ttl <duration>;
    keys { ( csk | ksk | zsk ) [ ( key-directory ) ] lifetime
      <duration_or_unlimited> algorithm <string> [ <integer> ]; ... };
    max-zone-ttl <duration>;
    parent-ds-ttl <duration>;
    parent-propagation-delay <duration>;
    publish-safety <duration>;
    retire-safety <duration>;
    signatures-refresh <duration>;
    signatures-validity <duration>;
    signatures-validity-dnskey <duration>;
    zone-propagation-delay <duration>;
};
```

dnssec-policy 语句为区定义了一个密钥和签名策略（KASP，key and signing policy）。

一个 KASP 决定一个或多个区将怎样使用 DNSSEC 签名。例如，它指定密钥多长时间应当轮转，使用哪种加密算法，以及 RRSIG 记录多久需要刷新。

不同的区之间不共享密钥，这意谓着每个区将生成一套密钥，即使条目有同样的策略。如果多个视

图配置了同一个区的不同版本，则每个单独的版本都将使用同一套签名密钥。

可以配置多个密钥和签名策略。要将一个策略绑定到一个区，在 zone 语句中增加一个 dnssec-policy 选项，指定要使用的策略名字。

每个密钥的轮转时间是根据 KASP 中定义的密钥生存期来计算的。生存期可以被区的 TTL 和广播延迟所修改，目的是防止验证失败。当一个密钥到达其生存期末尾时，named 将自动生成并发布一个新密钥，然后失效旧密钥并激活新的，最终根据计算的时间表将旧密钥废弃。

区签名密钥（ZSK，Zone-signing key）的轮转不需要操作员输入。密钥签名密钥（KSK，Key-signing key）和组合签名密钥（CSK，combined signing key）的轮转需要进行一个向父区提交一条 DS 记录的动作。对 KSK 和 CSK 的轮转时间进行调整，以考虑在处理和传播 DS 更新中的延迟。

有两个预定义的 dnssec-policy 名字：none 和 default 。将一个区的策略设置为 none 与完全不设置 dnssec-policy 等效；区将不会被签名。策略 default 导致区被一个使用算法 ECDSAP256SHA256 的单一组合签名密钥（CSK，combined signing key）所签名；这个密钥将有一个无限的生存期。（可以在源码树种找到这个策略的拷贝，在文件 doc/misc/dnssec-policy.default.conf 中。）

注解：在未来的版本中，缺省的签名策略可能会变化。这可能会在升级到新版本的 BIND 时要求签名策略的变化。在升级时仔细检查发行说明以了解这些变化。要防止在升级时的策略变化，使用一个显式定义的 dnssec-policy 而不是 default 。

如果一条 dnssec-policy 语句被修改并且服务器重启或重读配置，named 会试图平滑地从旧的策略改变为新的策略。例如，如果密钥算法改变了，就会使用新算法生成一个新密钥，在当前密钥的生存期结束之后，旧算法就会退役了。

注解：在一个密钥轮转正在进行时，不支持轮转到一个新策略，这可能导致无法预期的行为。

在一个 dnssec-policy 语句中，可以指定下列选项：

dnskey-ttl 这指定在生成 DNSKEY 资源记录时使用的 TTL。缺省是 1 小时（3600 秒）。

keys 这是一个指定用于生成密钥和签名区的算法和角色的列表。这个列表中的条目不代表特定的 DNSSEC 密钥，密钥可能会定期更改，而是密钥将在签名策略中扮演的角色。例如，配置一个使用算法 RSASHA256 的密钥确保 DNSKEY 资源记录集总是包含一个使用此算法的密钥签名密钥。

这里是在一个 keys 中可能出现的一些条目的一个例子（仅作说明用途）。

```
keys {
    ksk key-directory lifetime unlimited algorithm rsasha1 2048;
    zsk lifetime P30D algorithm 8;
    csk lifetime P6MT12H3M15S algorithm ecdsa256;
};
```

这个例子指定了三个应当用在区中的密钥。第一个符号决定了密钥在签名资源记录集中所扮演的角色。如果设为 ksk，这就是一个密钥签名密钥；它会设置有 KSK 标志并且只能用于对 DNSKEY，CDS 和 CDNSKEY 资源记录集签名。如果设为 zsk，这就是一个区签名密钥；其 KSK 标志位为复位状态，这个密钥可以用于对 **除了** DNSKEY，CDS 和 CDNSKEY 之外的其它所有资源记录集签名。如果设为 csk，这个密钥会设置 KSK 标志位并且可以用于给所有资源记录集签名。

一个可选的第二个符号决定密钥的存储位置。当前，密钥只能存放在 key-directory 配置的目录。这个符号可以在将来用于将密钥存放在硬件服务模块或分离的目录中。

lifetime 参数指定一个密钥在轮转前使用多长时间。在上面的例子中，第一个密钥有无限的生存期，第二个密钥可以使用 30 天，第三个密钥则具有一个相当特别地生存期，6 个月，12 小时，3 分又 15 秒。一个为 0 秒的生存期与 unlimited 相同。

注意，如果一个密钥退役太快可能引起验证失败时，其生存期可以延长。例如，如果密钥配置成轮转频繁超过了它的 TTL，它的生存期将会自动延长，以考虑到这一点。

algorithm 参数指定密钥的算法，表示成一个字符串（“rsasha256”，“ecdsa384”等等）或者一个十进制数。一个可选的第二个参数指定以位计算的密钥的大小。如果省略，如同例子中第二和第三个密钥，就会使用这种算法的一个合适的缺省大小。

publish-safety 这是一段在计算轮转时间时增加到发布前间隔的空白，以留出一些额外的时间来覆盖未预见的事件。这增加了密钥在发布和激活之间的时间。缺省值是 PT1H （1 小时）。

retire-safety 这是一段在计算轮转时间时增加到发布后间隔的空白，以留出一些额外的时间来覆盖未预见的事件。这增加了密钥在失活之后维持被发布状态的时间。缺省值是 PT1H （1 小时）。

signatures-refresh 这个决定一个 RRSIG 记录需要刷新的频繁程度。签名在距离过期时间小于指定的间隔时被更新。缺省值是 P5D （5 天），意谓着将在 5 天或更短时间内过期的签名会被刷新。

signatures-validity 这指示一个 RRSIG 记录的有效期（取决于起始偏移和抖动）。缺省值是 P2W （2 周）。

signatures-validity-dnskey 这个类似于 signatures-validity ，但仅对 DNSKEY 记录。缺省值是 P2W （2 周）。

max-zone-ttl 与 max-zone-ttl 区选项相似，这为区指定了以秒为单位的最大允许的 TTL 值。当使用值为 text 或者 raw 的 masterfile-format 装载一个区文件时，任何 TTL 大于 max-zone-ttl 的记录都会被限制为最大允许的 TTL 值。

这在 DNSSEC 维护的区中是需要的，因为当轮转到一个新的 DNSKEY，旧密钥需要保持可用，直到在缓存中的 RRSIG 记录过期。max-zone-ttl 选项确保区中最大的 TTL 不会比所设置的值更大。

注解: 因为 map 格式的文件直接装载到内存，这个选项不能与之共同使用。

缺省值是 PT24H （24 小时）。一个为零的 max-zone-ttl 将被视作如同使用了缺省值。

zone-propagation-delay 这是从一个区首次更新到新版本的区在所有辅服务器上提供服务的期望传播延迟。缺省值是 PT5M （5 分钟）。

parent-ds-ttl 这是父区使用的 DS 资源记录集的 TTL。缺省值是 P1D （1 天）。

parent-propagation-delay 这是从父区被更新到新版本在所有父区的名字服务器中提供服务的期望的传播延迟。缺省值是 PT1H （1 小时）。

4.2.21 managed-keys 语句语法

```
managed-keys { <string> ( static-key
  | initial-key | static-ds |
  initial-ds ) <integer> <integer>
  <integer> <quoted_string>; ... };, deprecated
```

4.2.22 managed-keys 语句定义和用法

managed-keys 语句已被废弃，建议使用带 initial-key 关键字的trust-anchors 语句语法 。

4.2.23 trusted-keys 语句语法

```
trusted-keys { <string> <integer>
  <integer> <integer>
  <quoted_string>; ... };, deprecated
```

4.2.24 trusted-keys 语句定义和用法

trusted-keys 语句已被废弃，建议使用带 static-key 关键字的trust-anchors 语句语法 。

4.2.25 view 语句语法

```
view view_name [ class ] {
  match-clients { address_match_list } ;
  match-destinations { address_match_list } ;
  match-recursive-only yes_or_no ;
 [ view_option ; ... ]
 [ zone_statement ; ... ]
} ;
```

4.2.26 view 语句定义和用法

view 语句是 BIND 9 的一个强大功能，它使一个名字服务器在回答一个 DNS 请求时，可以依据是谁在请求而有所不同。这在实现分割的 DNS 设置却不需要运行多个服务器时特别有用。

每个 view 语句定义了一个被一些客户端子集所看到的 DNS 名字空间的视图。一个客户匹配一个视图是指它的源地址与视图的 match-clients 子句中的 address_match_list 相匹配并且它的目的地址与视图的 match-destinations 子句中的 address_match_list 相匹配。如果未指定，match-clients 和 match-destinations 缺省都是匹配所有地址。除了检查 IP 地址，match-clients 和 match-destinations 也可以使用 keys，它给客户端提供了一个选择视图的机制。一个视图也可以指定成为 match-recursive-only，它表示仅仅来自相匹配客户的递归请求才匹配此视图。view 语句的顺序非常重要——一个客户端请求将在它所匹配的第一个 view 的上下文中被解析。

在一个 view 语句中定义的区只能被与这个 view 相匹配的客户端访问。通过在多个视图中定义同样名字的区，可以为不同的客户端提供不同的区数据，例如，在一个分割的 DNS 上建立 “internal” 和 “external” 客户端。

许多在 options 语句中给出的选项也可以用在 view 语句中，并且仅仅在解析这个视图之内的请求时生效。当没有给出按视图指定的值时，缺省使用 options 语句中的值。同样，区选项也使用在 view 语句中指定的值做缺省值；这些按视图指定的缺省值优先于其在 options 语句中的值。

视图是按类划分的。如果没有给出类，假设为 IN 类。注意所有的非 IN 视图必须包含一个暗示区，因为只有 IN 类有一个预编译的缺省暗示区。

如果配置文件中没有 view 语句，就自动在 IN 类中建立一个匹配所有客户端的缺省视图。然后在配置文件的顶层所指定的 zone 语句都是这个缺省视图的一部份，options 语句将会应用到这个缺省视图。如果提供任何显式的 view 语句，所有 zone 语句都必须出现在 view 语句之内。

这里是使用 view 语句实现分割 DNS 的一个典型设置：

```
view "internal" {
    // This should match our internal networks.
    match-clients { 10.0.0.0/8; };

    // Provide recursive service to internal
    // clients only.
    recursion yes;

    // Provide a complete view of the example.com
    // zone including addresses of internal hosts.
    zone "example.com" {
     type master;
     file "example-internal.db";
    };
};

view "external" {
    // Match all clients not matched by the
    // previous view.
    match-clients { any; };

    // Refuse recursive service to external clients.
    recursion no;

    // Provide a restricted view of the example.com
    // zone containing only publicly accessible hosts.
    zone "example.com" {
     type master;
```

（下页继续）

(续上页)

```
    file "example-external.db";
  };
};
```

4.2.27 zone 语句语法

```
zone <string> [ <class> ] {
  type ( master | primary );
  allow-query { <address_match_element>; ... };
  allow-query-on { <address_match_element>; ... };
  allow-transfer { <address_match_element>; ... };
  allow-update { <address_match_element>; ... };
  also-notify [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [ port <integer> ] |
↪<ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
  alt-transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
  alt-transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
  auto-dnssec ( allow | maintain | off );
  check-dup-records ( fail | warn | ignore );
  check-integrity <boolean>;
  check-mx ( fail | warn | ignore );
  check-mx-cname ( fail | warn | ignore );
  check-names ( fail | warn | ignore );
  check-sibling <boolean>;
  check-spf ( warn | ignore );
  check-srv-cname ( fail | warn | ignore );
  check-wildcard <boolean>;
  database <string>;
  dialup ( notify | notify-passive | passive | refresh | <boolean> );
  dlz <string>;
  dnskey-sig-validity <integer>;
  dnssec-dnskey-kskonly <boolean>;
  dnssec-loadkeys-interval <integer>;
  dnssec-policy <string>;
  dnssec-secure-to-insecure <boolean>;
  dnssec-update-mode ( maintain | no-resign );
  file <quoted_string>;
  forward ( first | only );
  forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_address> ) [ port
↪<integer> ] [ dscp <integer> ]; ... };
```

(下页继续)

(续上页)

```
    inline-signing <boolean>;
    ixfr-from-differences <boolean>;
    journal <quoted_string>;
    key-directory <quoted_string>;
    masterfile-format ( map | raw | text );
    masterfile-style ( full | relative );
    max-journal-size ( default | unlimited | <sizeval> );
    max-records <integer>;
    max-transfer-idle-out <integer>;
    max-transfer-time-out <integer>;
    max-zone-ttl ( unlimited | <duration> );
    notify ( explicit | master-only | <boolean> );
    notify-delay <integer>;
    notify-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    notify-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    notify-to-soa <boolean>;
    serial-update-method ( date | increment | unixtime );
    sig-signing-nodes <integer>;
    sig-signing-signatures <integer>;
    sig-signing-type <integer>;
    sig-validity-interval <integer> [ <integer> ];
    update-check-ksk <boolean>;
    update-policy ( local | { ( deny | grant ) <string> ( 6to4-self | external | krb5-self | krb5-selfsub | krb5-
↪subdomain | ms-self | ms-selfsub | ms-subdomain | name | self | selfsub | selfwild | subdomain | tcp-
↪self | wildcard | zonesub ) [ <string> ] <rrtypelist>; ... };
    zero-no-soa-ttl <boolean>;
    zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
    type ( slave | secondary );
    allow-notify { <address_match_element>; ... };
    allow-query { <address_match_element>; ... };
    allow-query-on { <address_match_element>; ... };
    allow-transfer { <address_match_element>; ... };
    allow-update-forwarding { <address_match_element>; ... };
    also-notify [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [ port <integer> ] |
↪<ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
    alt-transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
```

(下页继续)

(续上页)

```
    alt-transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    auto-dnssec ( allow | maintain | off );
    check-names ( fail | warn | ignore );
    database <string>;
    dialup ( notify | notify-passive | passive | refresh | <boolean> );
    dlz <string>;
    dnskey-sig-validity <integer>;
    dnssec-dnskey-kskonly <boolean>;
    dnssec-loadkeys-interval <integer>;
    dnssec-policy <string>;
    dnssec-update-mode ( maintain | no-resign );
    file <quoted_string>;
    forward ( first | only );
    forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_address> ) [ port
↪<integer> ] [ dscp <integer> ]; ... };
    inline-signing <boolean>;
    ixfr-from-differences <boolean>;
    journal <quoted_string>;
    key-directory <quoted_string>;
    masterfile-format ( map | raw | text );
    masterfile-style ( full | relative );
    masters [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [ port <integer> ] |
↪<ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
    max-journal-size ( default | unlimited | <sizeval> );
    max-records <integer>;
    max-refresh-time <integer>;
    max-retry-time <integer>;
    max-transfer-idle-in <integer>;
    max-transfer-idle-out <integer>;
    max-transfer-time-in <integer>;
    max-transfer-time-out <integer>;
    min-refresh-time <integer>;
    min-retry-time <integer>;
    multi-master <boolean>;
    notify ( explicit | master-only | <boolean> );
    notify-delay <integer>;
    notify-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    notify-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    notify-to-soa <boolean>;
```

(下页继续)

(续上页)

```
    request-expire <boolean>;
    request-ixfr <boolean>;
    sig-signing-nodes <integer>;
    sig-signing-signatures <integer>;
    sig-signing-type <integer>;
    sig-validity-interval <integer> [ <integer> ];
    transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    try-tcp-refresh <boolean>;
    update-check-ksk <boolean>;
    use-alt-transfer-source <boolean>;
    zero-no-soa-ttl <boolean>;
    zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
    type mirror;
    allow-notify { <address_match_element>; ... };
    allow-query { <address_match_element>; ... };
    allow-query-on { <address_match_element>; ... };
    allow-transfer { <address_match_element>; ... };
    allow-update-forwarding { <address_match_element>; ... };
    also-notify [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [ port <integer> ] |
↪<ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
    alt-transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    alt-transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    check-names ( fail | warn | ignore );
    database <string>;
    file <quoted_string>;
    ixfr-from-differences <boolean>;
    journal <quoted_string>;
    masterfile-format ( map | raw | text );
    masterfile-style ( full | relative );
    masters [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [ port <integer> ] |
↪<ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
    max-journal-size ( default | unlimited | <sizeval> );
    max-records <integer>;
    max-refresh-time <integer>;
    max-retry-time <integer>;
```

(下页继续)

(续上页)

```
    max-transfer-idle-in <integer>;
    max-transfer-idle-out <integer>;
    max-transfer-time-in <integer>;
    max-transfer-time-out <integer>;
    min-refresh-time <integer>;
    min-retry-time <integer>;
    multi-master <boolean>;
    notify ( explicit | master-only | <boolean> );
    notify-delay <integer>;
    notify-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    notify-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    request-expire <boolean>;
    request-ixfr <boolean>;
    transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    try-tcp-refresh <boolean>;
    use-alt-transfer-source <boolean>;
    zero-no-soa-ttl <boolean>;
    zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
    type hint;
    check-names ( fail | warn | ignore );
    delegation-only <boolean>;
    file <quoted_string>;
};
```

```
zone <string> [ <class> ] {
    type stub;
    allow-query { <address_match_element>; ... };
    allow-query-on { <address_match_element>; ... };
    check-names ( fail | warn | ignore );
    database <string>;
    delegation-only <boolean>;
    dialup ( notify | notify-passive | passive | refresh | <boolean> );
    file <quoted_string>;
    forward ( first | only );
    forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_address> ) [ port
↪<integer> ] [ dscp <integer> ]; ... };
```

(下页继续)

(续上页)

```
    masterfile-format ( map | raw | text );
    masterfile-style ( full | relative );
    masters [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [ port <integer> ] |
↪<ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
    max-records <integer>;
    max-refresh-time <integer>;
    max-retry-time <integer>;
    max-transfer-idle-in <integer>;
    max-transfer-time-in <integer>;
    min-refresh-time <integer>;
    min-retry-time <integer>;
    multi-master <boolean>;
    transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
    use-alt-transfer-source <boolean>;
    zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
    type static-stub;
    allow-query { <address_match_element>; ... };
    allow-query-on { <address_match_element>; ... };
    forward ( first | only );
    forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_address> ) [ port
↪<integer> ] [ dscp <integer> ]; ... };
    max-records <integer>;
    server-addresses { ( <ipv4_address> | <ipv6_address> ); ... };
    server-names { <string>; ... };
    zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
    type forward;
    delegation-only <boolean>;
    forward ( first | only );
    forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_address> ) [ port
↪<integer> ] [ dscp <integer> ]; ... };
};
```

```
zone <string> [ <class> ] {
    type redirect;
    allow-query { <address_match_element>; ... };
    allow-query-on { <address_match_element>; ... };
    dlz <string>;
    file <quoted_string>;
    masterfile-format ( map | raw | text );
    masterfile-style ( full | relative );
    masters [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [ port <integer> ] |
↪<ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
    max-records <integer>;
    max-zone-ttl ( unlimited | <duration> );
    zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
    type delegation-only;
};
```

```
zone <string> [ <class> ] {
    in-view <string>;
};
```

4.2.28 zone 语句定义和用法

区类型

zone 配置要求 type 关键字，除非它时一个 in-view 配置。可以接受的值包括：primary（或 master），secondary（或 slave），mirror，delegation-only，forward，hint，redirect，static-stub 和 stub 。

primary 一个主区拥有区数据的主拷贝，并能够提供关于这个区的权威回答。类型 master 与 primary 是同义词。

secondary 一个辅区是一个主区的一份复制。类型 slave 与 secondary 是同义词。masters 列表指定一个或多个主服务器，辅服务器从其更新自己的区拷贝。主服务器列表元素也可以是其它主服务器列表的名字。缺省时，自 53 端口发出区传送；这个也可以通过在 IP 地址列表之前指定一个端口号修改，这对所有服务器都适用；或者在 IP 地址之后指定端口，这个就依每个服务器而不同。对主服务器的认证也可以基于每个服务器的 TSIG 密钥来做。如果指定了一

个文件，无论何时区有了改变，就将复制来的数据写到这个文件中，并在服务重新启动时重新装载这个文件。推荐使用一个文件，因为这常常会加速服务的启动并消除不必要的带宽浪费。注意，对一个服务器上大量的区（数万或数十万），最好对区文件名使用两级命名机制。例如，区 example.com 的辅服务器可能将区的内容放进名为 ex/example.com 的文件中，在这里 ex/ 仅仅是区名的头两个字母。（如果将 100000 个文件放在一个目录中，大多数操作系统操作都非常缓慢。）

stub 存根（stub）区类似于一个辅区，差别仅仅是它只复制一个主区中的 NS 记录而不是整个区。存根区不是 DNS 的标准部份；它只是 BIND 实现的一个特性。

存根区可以用来消除父区中粘着 NS 记录的需求，代价是维护一个存根区条目和 named.conf 中名字服务器地址的集合。在新配置中不推荐这种用法，并且 BIND 9 仅仅以有限的方式支持它。如果一个 BIND 9 主服务器配置成一个父区，并配置了子区的存根区，那么所有父区的辅服务器也都需要配置同样的子区的存根区。

存根区也可以用于强制使用某个特定权威服务器集合来解析某一给定域名。例如，使用 RFC 1918 地址的某个私有网络上的缓存服务器可能为 10.in-addr.arpa 配置存根区，并使用内部名字服务器的一个集合作为这个区的权威服务器。

mirror 一个镜像区类似于一个类型为 secondary 的区，除了其数据在用于响应之前会被 DNSSEC 验证。在区传送过程中，验证被应用到整个区，并且当 named 重启时，从磁盘装载区文件时又会再次进行验证。如果对一个新版本镜像区的验证失败，就调度一次重传，并使用最近正确验证过的这个区的版本，直到其过期或者一个更新的版本正确验证。如果一个镜像区完全没有可以使用的区数据，无论是因为区传送失败还是过期，作为替代，都会使用传统的 DNS 递归来查找答案。镜像区不能用在没有打开递归的视图中。

来自镜像区域的答复看起来几乎与来自 secondary 区的答复一模一样，除了这个显著的例外，即 AA 位（“权威答复”）未被置位，且 AD 位（“认证数据”）被置位。

镜像区的目的是用于设置一个快速的根区的本地拷贝，类似于在 RFC 7706 中所描述的。IANA 根区的一个缺省的主服务器列表内置在 named ，因而它的镜像可以使用下列配置来打开：

```
zone "." {
   type mirror;
};
```

镜像一个非根区需要使用 masters 选项提供一个显式的主服务器列表（参见masters 语句语法 获取详细信息），并且要将指定区的一个密钥签名密钥（KSK）显式配置成一个信任锚。

要使镜像区内容在两次 named 重启间持久化，使用file 选项。

当为一个镜像区配置 NOTIFY 时，仅能在区一级使用 notify no; 和 notify explicit; 。在区一

级使用 notify 的任何其它设置都是配置错误。在 options 或者 view 级使用 notify 的任何其它设置将导致这个设置被镜像区的 notify explicit; 所覆盖。notify 选项的全局缺省值是 yes，所以镜像区缺省是配置为 notify explicit; 。

镜像区的出向区传送缺省是被关闭的，但是可以使用allow-transfer 开启。

注解: 对根区之外的其它任何区使用这个区类型都应考虑为 **试验性的**，可能会有性能问题，特别是对特别大的区和/或被频繁更新的区。

---

static-stub 静态存根（static-stub）区类似于存根区，仅仅有下列例外：区数据是静态配置而不是从主服务器传送而来；当一个匹配某个静态存根区的请求需要递归时，总是使用本地配置的数据（名字服务器的名字和粘连地址），即使其缓存了不同的权威信息。

区数据使用 server-addresses 和 server-names 区选项来配置。

区数据在内部是以 NS 和（如果需要）粘连的 A 或者 AAAA 资源记录的形式维护，可以使用 rndc dumpdb -all 将区数据库导出为可读的格式。配置的资源记录集被认为是本地配置参数，而不是公共数据。所以发给一个静态存根区的非递归请求（即那些 RD 位复位的）是被禁止的，并且以 REFUSED 响应。

由于数据是静态配置的，对于一个静态存根区来说，不会发生区维护动作。例如，没有定期刷新，收到的通知消息将会被拒绝，并带有一个 NOTAUTH 的响应码（rcode）。

每个静态存根区都被配置为带有内部生成的 NS 和（如果需要）粘连的 A 或者 AAAA 资源记录。

forward 一个转发区是基于域名配置转发的方式。一个 forward 类型的 zone 语句可以包含 forward 和/或 forwarders 语句，这将对由区名所给出的域内的请求起作用。如果没有 forwarders 语句或者为 forwarders 配置一个空表，这个域就不会有转发动作，即取消了 options 语句中任何转发者的效果。这样，如果想使用这种类型的区来改变全局 forward 选项（即，先 “forward first”，然后 “forward only”，或者相反）的行为，但是想要使用同样的服务器作为全局设置，需要重新指定全局转发者。

hint 使用一个提示区来指定初始的根名字服务器集合。在服务器启动时，它使用根提示信息来找到一个根名字服务器并从后者获取最近的根名字服务器名单。如果没有为类 IN 指定提示区，服务器使用编译内建的根服务器提示集合。IN 之外的其它类没有内建缺省提示。

redirect 重定向区用于在普通解析将导致返回一个 NXDOMAIN 时，而给请求提供一个回答。每个视图只支持一个重定向区。可以使用 allow-query 限制哪些客户端可以看见这些回复。

如果客户端请求 DNSSEC 记录（DO=1）并且 NXDOMAIN 响应被签名，就不会发生任何替

换。

要重定向所有的 NXDOMAIN 响应到 100.100.100.2 和 2001:ffff:ffff::100.100.100.2，需要配置一个名为“.”的、类型为 redirect 的区，其区文件中包含指向期望地址的通配符记录："*. IN A 100.100.100.2" 和 "*. IN AAAA 2001:ffff:ffff::100.100.100.2" 。

另外一个例子，要重定向所有西班牙名字（在.ES 下面），需要使用类似的名字，但要用 *.ES. 替代 *. 。要重定向所有商业类西班牙名字（在.COM.ES 下面），需要使用通配符条目 *.COM.ES. 。

注意重定向区支持所有可能的类型；它不限于 A 和 AAAA 记录。

如果一个重定向区配置了一个 masters 选项，那么它就会像一个辅区一样从外面做区传送。否则，它就会如同一个主区一样从一个文件装载。

由于重定向区不会直接被名字引用，它们不保存在普通主区和辅区的区查找表中。因此，当前不可能使用要重新装载一个重定向区，使用 rndc reload -redirect ，要重新传输一个作为辅区配置的重定向区，使用 rndc retransfer -redirect 。当使用 rndc reload 而不指定一个区名时，重定向区将会随着其它区被重载。

delegation-only 这个区类型用于强制基础区（如 COM，NET，ORG）为只授权状态。任何收到的回答在权威部份没有一个显式或隐式的授权时都会被当做 NXDOMAIN。这个不应用在区顶点，也不应该应用在叶子区。

delegation-only 对从转发服务器来的回答没有效果。

参见root-delegation-only 中的说明。

类

作为可选项，区的名字后面可以跟一个类。如果未指定类，就设定为类 IN （表示 Internet ）。这在大多数情况下是正确的。

hesiod 类是表示来自 MIT 的雅典娜项目的一种信息服务。它用于在不同系统数据库之间共享信息，如用户、组、打印机等等。关键字 HS 是 hesiod 的符号。

另一个 MIT 开发的是 Chaosnet，一个创建于 1970 年代中期的局域网协议。它的区数据可以使用 CHAOS 类指定。

区选项

allow-notify 参见访问控制 中对 allow-notify 的描述。

allow-query 参见访问控制 中对 allow-query 的描述。

allow-query-on 参见访问控制 中对 allow-query-on 的描述。

allow-transfer 参见访问控制 中对 allow-transfer 的描述。

allow-update 参见访问控制 中对 allow-update 的描述。

update-policy 这个指定一个“简单安全更新”策略。参见动态更新策略 。

allow-update-forwarding 参见访问控制 中对 allow-update-forwarding 的描述。

also-notify 这个选项仅在这个区的 notify 是激活时有意义。会收到这个区的 DNS NOTIFY 消息的机器集由所有为这个区而列出的区名字服务器（主服务器除外）组成，外加 also-notify 所指定的任何 IP 地址。可以在每个 also-notify 地址指定通知消息要送达的端口，缺省为 53。也可以指定一个 TSIG 密钥，使 NOTIFY 被给出的密钥签名。also-notify 对存根区没有意义。缺省是空表。

check-names 这个选项用于限制某些域名的字符集和语法，这些域名包括来自主文件和/或来自网络的 DNS 响应。缺省值是依区类型的不同而不同的。对 primary 区缺省值是 fail 。对 secondary 区缺省值是 warn 。对于 hint 区没有实现。

check-mx 参见布尔选项 中对 check-mx 的描述。

check-spf 参见布尔选项 中对 check-spf 的描述。

check-wildcard 参见布尔选项 中对 check-wildcard 的描述。

check-integrity 参见布尔选项 中对 check-integrity 的描述。

check-sibling 参见布尔选项 中对 check-sibling 的描述。

zero-no-soa-ttl 参见布尔选项 中对 zero-no-soa-ttl 的描述。

update-check-ksk 参见布尔选项 中对 update-check-ksk 的描述。

dnssec-loadkeys-interval 参见options 语句定义和用法 中对 dnssec-loadkeys-interval 的描述。

dnssec-update-mode 参见options 语句定义和用法 中对 dnssec-update-mode 的描述。

dnssec-dnskey-kskonly 参见布尔选项 中对 dnssec-dnskey-kskonly 的描述。

try-tcp-refresh 参见布尔选项 中对 try-tcp-refresh 的描述。

database 这个指定用于存放区数据的数据库类型。database 关键字后跟的字符串被解释为一个空格分隔的单词列表。第一个单词标识数据库类型，接下来的单词作为参数传递给数据库，并按照数据库类型所指定的处理方式来解释。

缺省是 "rbt"，BIND 9 的原生内存红黑树数据库。这种数据库不需要参数。

如果有附加的数据库驱动被链接到服务器，也可能有其它值。在分发包中包含一些简单的驱动，但是缺省没有被链接进来。

dialup 参见布尔选项 中对 dialup 的描述。

delegation-only 这个标志只用于转发区，提示区和存根区。如果设置为 yes，区将被当成一个只授权类型的区。

参见root-delegation-only 中的说明。

file 这个设置区的文件名，在 primary，hint 和 redirect 这些没有定义 masters 的区中，区数据是从这个文件装载。在 secondary，mirror，stub 和 redirect 这些定义了 masters 的区中，区数据是来自另一个服务器并保存在这个文件中。这个选项对于其它区类型是不适用的。

forward 这个选项仅在区有一个转发者列表的时候有意义。only 值将会在试探转发者却没有获得回复之后导致查找失败，而 first 值将会允许进行一个普通查找的试探。

forwarders 这个用于重载全局转发者列表。如果没有在一个 forward 类型的区中指定，对此区就没有转发，也不使用全局选项。

journal 这个允许覆盖缺省的日志文件名。缺省为区文件后增“.jnl”。这个应用于 primary 和 secondary 区中。

max-ixfr-ratio 参见options 语句定义和用法 中对 max-ixfr-ratio 的描述。

max-journal-size 参见服务器资源限制 中对 max-journal-size 的描述。

max-records 参见服务器资源限制 中对 max-records 的描述。

max-transfer-time-in 参见区传送 中对 max-transfer-time-in 的描述。

max-transfer-idle-in 参见区传送 中对 max-transfer-idle-in 的描述。

max-transfer-time-out 参见区传送 中对 max-transfer-time-out 的描述。

max-transfer-idle-out 参见区传送 中对 max-transfer-idle-out 的描述。

notify 参见布尔选项 中对 notify 的描述。

notify-delay 参见调优 中对 notify-delay 的描述。

notify-to-soa 参见布尔选项 中对 notify-to-soa 的描述。

zone-statistics 参见options 语句定义和用法 中对 zone-statistics 的描述。

server-addresses 这个选项仅对静态存根区有意义。这是一个 IP 地址列表，在递归解析这个区时，请求将被发向这个列表。一个为这个选项配置的非空列表，会在内部配置顶点 NS 资源记录及附带的粘连 A 或者 AAAA 记录。

例如，如果 “example.com” 被配置成一个静态存根区，在一个 server-addresses 选项中配置 192.0.2.1 和 2001:db8::1234，内部将被配置为下列资源记录。

```
example.com. NS example.com.
example.com. A 192.0.2.1
example.com. AAAA 2001:db8::1234
```

这些记录将会被用于解析这个静态存根区下的名字。例如，如果服务器收到一个带 RD 位的 “www.example.com” 请求，服务器将发起一个递归解析，将请求发给 192.0.2.1 和/或 2001:db8::1234。

server-names 这个仅对静态存根区有意义。这是一个名字服务器域名的列表，充当这个静态存根区的权威服务器。在 named 需要发送请求到这些服务器时，这些名字将被解析成 IP 地址。为使这个补充解析能够成功，这些名字必须不能是静态存根区原点名的子域名。即，当一个静态存根区的原点是 “example.net” 时，“ns.example” 和 “master.example.com” 可以设定在 server-names 选项中，但是 “ns.example.net” 不能，它将被配置分析器拒绝。

为这个选项配置一个非空列表，将会使用指定的名字在内部配置顶点 NS 资源记录。例如，如果 “example.com” 被配置为一个静态存根区，并在一个 server-names 选项配置 “ns1.example.net” 和 “ns2.example.net”，内部将被配置为下列资源记录：

```
example.com. NS ns1.example.net.
example.com. NS ns2.example.net.
```

这些记录将会被用于解析这个静态存根区下的名字。例如，如果服务器收到一个带 RD 位的 “www.example.com” 请求，服务器将发起一个递归解析，先将 “ns1.example.net” 和/或 “ns2.example.net” 解析成 IP 地址，再将请求发给这些地址中的一个或多个。

sig-validity-interval 参见调优 中对 sig-validity-interval 的描述。

sig-signing-nodes 参见调优 中对 sig-signing-nodes 的描述。

sig-signing-signatures 参见调优 中对 sig-signing-signatures 的描述。

sig-signing-type 参见调优 中对 sig-signing-type 的描述。

transfer-source 参见区传送 中对 transfer-source 的描述。

transfer-source-v6 参见区传送 中对 transfer-source-v6 的描述。

alt-transfer-source 参见区传送 中对 alt-transfer-source 的描述。

alt-transfer-source-v6 参见区传送 中对 alt-transfer-source-v6 的描述。

use-alt-transfer-source 参见区传送 中对 use-alt-transfer-source 的描述。

notify-source 参见区传送 中对 notify-source 的描述。

notify-source-v6 参见区传送 中对 notify-source-v6 的描述。

min-refresh-time; max-refresh-time; min-retry-time; max-retry-time 参见调优 中的描述。

ixfr-from-differences 参见布尔选项 中对 ixfr-from-differences 的描述。（注意 ixfr-from-differences 选择 master 和 slave 不能用于区这一级。）

key-directory 参见options 语句定义和用法 中对 key-directory 的描述。

auto-dnssec 参见options 语句定义和用法 中对 auto-dnssec 的描述。

serial-update-method 参见options 语句定义和用法 中对 serial-update-method 的描述。

inline-signing 如果为 yes，将允许对一个区进行" bump in the wire" 签名，这时，一个未签名区通过区传送传入或从磁盘加载，将会生成区的签名版本，可能带有一个不同的序列号。此特性缺省是关闭的。

multi-master 参见布尔选项 中对 multi-master 的描述。

masterfile-format 参见调优 中对 masterfile-format 的描述。

max-zone-ttl 参见options 语句定义和用法 中对 max-zone-ttl 的描述。

dnssec-secure-to-insecure 参见布尔选项 中对 dnssec-secure-to-insecure 的描述。

动态更新策略

BIND 9 支持两种方法来授权客户端对一个区执行动态更新，它们是配置 allow-update 或 update-policy 选项。

allow-update 子句是一个简单的访问控制表，任何与其匹配的客户端被授权可以更改区中的任意记录。

update-policy 子句允许对所授权的更新作更细粒度的控制。它指定一个规则集，其中的每个规则是授权或禁止区中一个或多个名字被一个或多个身份所更新。身份是由密钥决定的，这个密钥使用 TSIG 或者 SIG(0) 签名更新请求。大多数情况下，update-policy 仅仅应用于基于密钥的身份。没法指定基于客户端源地址更新权限。

update-policy 规则仅对 primary 类型的区有意义，不能用于任何其它类型的区中。同时指定 allow-update 和 update-policy 会报出一个配置错误。

一个预定义的 update-policy 规则，可以使用命令 update-policy local; 打开。named 在启动时自动生成一个 TSIG 会话密钥，并将其存放在一个文件中；在 named 运行时这个密钥可以被本机客户

端用于更新区。缺省时，会话密钥存放在文件 /var/run/named/session.key 中，密钥名是 “local-ddns”，密钥算法是 HMAC-SHA256。这些值可以分别由 session-keyfile ， session-keyname 和 session-keyalg 选项配置。一个运行在本地系统上的客户端，如果以合适的授权运行，可以从密钥文件读取会话密钥并使用它对更新请求签名。区的更新策略被设置为允许那个密钥改变区中的任何记录。假设密钥名是 “local-ddns”，这个策略等效于：

```
update-policy { grant local-ddns zonesub any; };
```

带有附加限制条件，即只有连接来自本地系统的客户端允许发送更新。

注意，named 只生成一个会话密钥；所有配置成使用 update-policy local 的区将接受同一个密钥。

命令 nsupdate -l 实现了这个特性，发送请求到本机，并使用从会话密钥文件中取得的密钥对其签名。

其它的规则看起来像这个：

```
( grant | deny ) identity ruletype  name   types
```

每条规则授予或禁止权限。规则按其在 update-policy 语句中指定的顺序进行检查。一旦一条消息成功地匹配了一条规则，立即进行授予或禁止操作，而不再进行更多的规则检查。存在 13 种类型的规则；规则类型由 ruletype 字段指定，并且对其它字段的解释是变化的，这依赖于规则类型。

通常，一条规则匹配是指对一个更新请求签名的密钥与 identity 字段匹配，将被更新记录的名字与 name 字段匹配（由 ruletype 字段所指定的方式），并且将被更新记录的类型与 types 字段匹配。每种规则类型的详细情况将在下面描述。

identity 字段必须设置成一个完整域名。在大多数情况，表示必须被用于签名更新请求的 TSIG 或者 SIG(0) 密钥的名字。如果指定的名字是一个通配符，它服从 DNS 通配符扩展规范，并且规则可以应用到多个个体。当使用一个 TKEY 交换来创建一个共享密钥时，用于认证 TKEY 交换的密钥标识将被用作共享秘钥的标识。一些规则类型使用标识与客户端的 Kerberos principal（例如，"host/machine@REALM" ）或 Windows realm（machine$@REALM ）匹配。

name 字段也指定一个完整域名。这通常标识将被更新的记录的名字。对这个字段的解释依赖于规则类型。

如果没有显式指定类型，一条规则就匹配除了 RRSIG，NS，SOA，NSEC 和 NSEC3 之外的所有类型。类型可以由名字指定，包括 ANY ； ANY 匹配除了 NSEC 和 NSEC3 之外的所有类型，这两种永远不能被更新。注意当试图删除与一个名字相联系的所有记录时，会针对每种存在的记录检查规则。

ruletype 字段有 16 个值：name ，subdomain ，wildcard ，self ，selfsub ，selfwild , krb5-self ，ms-self ，krb5-selfsub ，ms-selfsub ，krb5-subdomain ，ms-subdomain ，tcp-self ，6to4-self

，zonesub 和 external 。

name 使用精确匹配语义，这个规则在被更新的名字与 name 字段内容一致时匹配。

subdomain 当被更新的名字是 name 字段内容的子域或与其一致时，这个规则匹配。

zonesub 这个规则与 subdomain 类似，除了当被更新的名字是配置了 update-policy 语句的区的子域时，它会匹配。这就排除了两次输入区名的需要，并且能够在多个区中使用标准的 update-policy 语句而不需要修改。在使用这条规则时，name 字段会被忽略。

wildcard name 字段服从 DNS 通配符扩展，当被更新的名字是一个通配符的有效扩展时，这条规则匹配。

self 当被更新记录的名字匹配 identity 字段的内容时，这条规则匹配。name 字段被忽略。为避免混淆，推荐将这个字段设置为与 identity 字段一致或为“.”。在允许为每个要更新的名字使用一个密钥时，self 规则类型是最有用的，这时密钥与被更新的记录有同样的名字。这种情况下，identity 字段可以被指定为 * （星号）。

selfsub 这条规则与 self 相似，除了 self 的子域也可以被更新之外。

selfwild 这条规则与 self 相似，除了仅仅只有 self 的子域才能被更新之外。

ms-self 当一个客户端使用一个 Windows 机器标识（例如，machine$@REALM ）发送一个 UPDATE，这条规则允许带有 machine.REALM 的绝对名字的记录被更新。

要匹配的域（译注：realm）在 identity 字段中指定。

name 字段在本规则中没有效果；作为占位符，它应被设置为“.”。

例如，grant EXAMPLE.COM ms-self . A AAAA 允许域 EXAMPLE.COM 中所有带有一个有效标记的机器更新它自己的地址记录。

ms-selfsub 这个与 ms-self 相似，除了它还允许更新在 Windows 机器标记下指定名字的任何子域，而不仅仅它自身。

ms-subdomain 当一个客户端使用一个 Windows 机器标识（例如，machine$@REALM ）发送一个 UPDATE 时，这条规则允许在指定域（译注：realm）中的任何机器更新区中或区的指定子域中的任何记录。

要匹配的域（译注：realm）在 identity 字段中指定。

name 字段指定可能被更新的子域。如果设置为“.”（或任何在区顶点及之上的任何其它名字），任何区内的名字都可以被更新。

例如，如果区“example.com”的 update-policy 包含 grant EXAMPLE.COM ms-subdomain hosts.example.com. A AAAA，任何在域（译注：realm）EXAMPLE.COM 内带有有效标识（译注：principal）的机器都能够更新在 hosts.example.com 或之下的地址记录。

krb5-self 当一个客户端使用一个 Kerberos 机器标识（例如，host/machine@REALM ）发送一个 UPDATE 时，这条规则允许带有完全 machine 名字的记录被更新，前提是它已被 REALM 认证了。这个类似于 ms-self ，但不完全与后者相同，这是由于 Kerberos 标识的 machine 部份是一个完全名字而不是一个不完全的名字。

要匹配的领域在 identity 字段中指定。

name 字段对这条规则没有影响；它应该被设置为 “.” 作为一个占位符。

例如，grant EXAMPLE.COM krb5-self . A AAAA 允许任何在域 EXAMPLE.COM 带有一个有效标识的机器更新其自身的地址记录。

krb5-selfsub 这条类似于 krb5-self ，除了它也允许更新在 Kerberos 标识的 machine 部份中指定的名字的任何子域，而不仅仅是名字自身。

krb5-subdomain 这条规则与 ms-subdomain 相同，除了它使用 Kerberos 机器标识（即，host/machine@REALM ）工作，而不是使用 Windows 机器标识。

tcp-self 这条规则允许通过 TCP 发送更新，并且对这些更新，从客户端的 IP 地址到 in-addr.arpa 和 ip6.arpa 名字空间的标准映射与被更新的名字进行匹配。identity 字段必须与名字匹配。name 字段应该被设置为 “.”。注意，由于 identity 是基于客户端的 IP 地址，不必对更新请求消息进行签名。

注解: 理论上有可能欺骗这些 TCP 会话。

6to4-self 这允许任何来自 6to4 网络或来自对应的 IPv4 地址的 TCP 连接更新与一个 6to4 IPv6 前缀匹配的名字，如同在 RFC 3056 中所描述的。这是刻意允许 NS 或 DNAME 资源记录集被添加到 ip6.arpa 反向树中。

identity 字段必须和 ip6.arpa 中的 6to4 前缀相匹配。name 字段应被设置为 “.”。注意，由于 identity 是基于客户端的 IP 地址，不必对更新请求消息进行签名。

另外，如果指定一个在 2.0.0.2.ip6.arpa 名字空间之外的 ip6.arpa 名字，对应的/48 反向名字可以被更新。例如，来自 2001:DB8:ED0C::/48 的 TCP/IPv6 连接可以更新 C.0.D.E.8.B.D.0.1.0.0.2.ip6.arpa 下的记录。

注解: 理论上有可能欺骗这些 TCP 会话。

---

external 这个规则允许 named 推迟决定是否允许一个给定的更新到一个外部的服务进程。

和服务进程通信的方法在 identity 字段中指定，其格式是 “local:path ”，这里 “path” 是一个 UNIX 域套接字的位置。（当前，“local” 是唯一支持的机制。）

到外部服务进程的请求通过 UNIX 域套接字以数据报发送，其格式为：

```
Protocol version number (4 bytes, network byte order, currently 1)
Request length (4 bytes, network byte order)
Signer (null-terminated string)
Name (null-terminated string)
TCP source address (null-terminated string)
Rdata type (null-terminated string)
Key (null-terminated string)
TKEY token length (4 bytes, network byte order)
TKEY token (remainder of packet)
```

服务进程以网络字节序回应一个四字节值，包含 0 或者 1；0 表示指定的更新不被允许，1 表示其被允许。

多视图

当使用了多个视图，一个区可能被在超过一个视图中引用。通常，这些视图会对同样的名字包含不同的区，即运行不同的客户端对同样请求收到不同的回复。然而，在许多时候，期望多个视图包含同样的区。in-view 区选项提供一个有效的方法来做这事：它允许一个视图引用在之前一个已配置视图中定义的一个区。例如：

```
view internal {
  match-clients { 10/8; };

  zone example.com {
  type master;
  file "example-external.db";
  };
};

view external {
  match-clients { any; };

  zone example.com {
  in-view internal;
  };
```

（下页继续）

(续上页)

```
};
```

一个 in-view 选项不能引用到在配置文件后面定义的视图。

一个使用了 in-view 选项的 zone 语句不能使用除 forward 和 forwarders 之外的任何其它选项。(这些选项控制包含视图的行为，而不是改变区对象自身。)

区级 ACL（例如：allow-query，allow-transfer）和区的其它详细配置是在定义区的视图中设置。需要足够仔细地确保这些 ACL 足够广泛能适应引用区的所有视图。

一个 in-view 区不能用作一个响应策略区。

一个 in-view 区不能引用一个 forward 区。

第 4.3 节　区文件

4.3.1　资源记录的类型及使用时机

本节很大部份是引用自 RFC 1034 ，描述资源记录（RR）的概念并解释了其使用时机。自从 RFC 1034 公布以来，几种新的 RR 被确定并在 DNS 中实现。这些 RR 也被包括进来。

资源记录

一个域名标识一个节点。每个节点有一个资源信息的集合，这个集合可以是空的。资源信息集合与一个由分离的资源记录所组成的特定名字相关。资源记录在集合中的顺序没有意义，也不会在名字服务器，解析器或 DNS 的其它部份中保持。但是，允许出于优化的目的对多个资源记录排序，例如，指定某个特定的附近的服务器第一个被尝试。参见sortlist 语句 和资源记录集排序 。

资源记录的组成部份为:

owner name　域名，用于确定资源记录的位置。

type　一个编码的 16 位值，指定资源记录的类型。

TTL　资源记录的生存时间。这个字段是以一个 32 位整数表示秒数，主要用于解析器缓存资源记录时。TTL 描述一个资源记录在其被丢弃前可以被缓存多长的时间。

class　一个编码的 16 位值，指定一个协议族或一个协议的实例。

RDATA　资源数据。数据的格式是与类型相关的，有时也是与类相关的。

有效资源记录的 **类型**的网站清单，包括那些已经被废弃的，请参考 https://en.wikipedia.org/wiki/List_of_DNS_record_types.

以下是当前 DNS 中有效的资源记录的 **类**：

IN 互联网（Internet）。

CH CHAOSnet，一个 MIT 在 1970 年代中期建立的局域网协议。由于其历史上的目的而使用非常少，但是被 BIND 重新用于服务器内建信息区，如，version.bind 。

HS Hesiod，一个由 MIT 的雅典娜（Athena）项目所开发的信息服务。它用于在不同的系统数据库之间共享信息，如用户，用户组，打印机等等。

主名字通常是隐含的，而不是形成资源记录的一个完整部份。例如，许多名字服务器在内部为名字空间形成树或 hash 结构，和资源记录链节点。其余的资源记录部份是固定的头部（类型，类，TTL），对所有的资源记录是一致的，还有一个变化的部份（RDATA）要适应其所描述的记录。

TTL 字段是一个资源记录可以被保留在缓存中的时间长度。这个限制不应用到区中的权威数据；它也会过期，但是遵从区的刷新策略。TTL 由数据源所在区的区管理员指定。短的 TTL 可以减少缓存，一个为零的 TTL 禁止缓存，互联网性能的现实建议对典型的主机，这些时间应该在天的数量级。如果有一个预期的变化，先在变化之前将 TTL 减小以在变化期间缩短不一致的持续时间，然后在变化之后增加回到其原先的值。

资源记录的 RDATA 部份中的数据作为二进制字符串和域名的一个组成部份被携带。域名频繁地被用作指向 DNS 中其它数据的“指针”。

文本形式的资源记录表示

资源记录在 DNS 协议包内被表示成二进制形式，而在存放到一个名字服务器或解析器中通常被表示成高级编码的形式。在 RFC 1034 中提供的例子中，使用一个类似主文件中的形式来显示资源记录的内容。在这个格式中，大多数资源记录显示在一行之内，虽然可以使用括号显示到多行中。

一行的开始是资源记录的属主。如果一行以一个空字符开始，属主就假设是与上一个资源记录一样。通常包含一些空行以增加可读性。

在属主之后，依次列出资源记录的 TTL，类型和类。类型和类使用上面所定义的缩写，TTL 是一个整数，在类型字段之前。为了避免词法分析时的二义性，类型和类的缩略语不能有交集，TTL 是整数，类型缩略语总是在最后。IN 类和 TTL 值在例子中通常省略，主要是为了更清楚。

资源数据或者资源记录的 RDATA 部份根据各种数据的表示方法给出。

例如，我们可能这样显示在一个消息中所携带的资源记录：

ISI.EDU.	MX	10 VENERA.ISI.EDU.
	MX	10 VAXA.ISI.EDU
VENERA.ISI.EDU	A	128.9.0.32
	A	10.1.0.52
VAXA.ISI.EDU	A	10.2.0.27
	A	128.9.0.33

MX 资源记录的 RDATA 部份由一个 16 位数字和一个紧随的域名组成。地址资源记录使用一个标准的 IP 地址格式来包含一个 32 位的互联网地址。

上述例子显示 6 条资源记录，即 3 个域名，每个域名带有 2 条资源记录。

这里是另一个可能的例子：

XX.LCS.MIT.EDU.	IN A	10.0.0.44
	CH A	MIT.EDU. 2420

这个显示了 XX.LCS.MIT.EDU 的两个地址，分别在不同的类中。

4.3.2 对 MX 记录的讨论

如同上面所描述的，域名服务器将消息存放为一系列资源数据，每个资源数据都包含一个关于一个给定域名（通常但不总是一个主机）的特定的信息片段。理解一个资源记录的最简单的方式是将其作为一个类型化数据对，即一个域名和与其匹配的相关数据，和一些附加的类型信息一起存储，用以帮助系统决定何时资源记录是相关的。

MX 记录用于控制电子邮件的投递。在记录中指定的数据是一个优先级和一个域名。优先级控制电子邮件尝试投递的顺序，数字最小的最优先。如果两个优先级相同，就随机选择一个服务器。如果一个给定优先级的服务器没有响应，邮件传输代理（MTA，mail transport agent）将会选择下一个更大的优先数。优先数大小没有绝对含义；它们仅仅是相对于这个域名的其它 MX 记录而言。所给出的域名是邮件将要被投递到的机器。它 **必须**有一个相关的地址记录（A 或者 AAAA）—CNAME 是不够的。

对于一个给定域，如果同时有一个 CNAME 记录和一个 MX 记录，MX 记录是错误的，将被忽略。作为替代，邮件将被投递到被 CNAME 所指向的 MX 记录所指定的服务器上。例如：

example.com.	IN	MX	10	mail.example.com.
	IN	MX	10	mail2.example.com.
	IN	MX	20	mail.backup.org.
mail.example.com.	IN	A	10.0.0.1	
mail2.example.com.	IN	A	10.0.0.2	

邮件投递先尝试 mail.example.com 和 mail2.example.com （以任何顺序），如果这些都没有成功，将会尝试投递到 mail.backup.org 。

4.3.3 设置 TTL

资源记录的生存期字段是一个 32 位的整数，它的单位为秒，主要用于解析器缓存资源记录。TTL 描述一个资源记录在被丢弃前可以被缓存多长时间。当前用于一个区文件中的有以下三种类型的 TTL。

SOA SOA 的最后一个字段是否定缓存 TTL。它控制从这台服务器发出的没有这个域名（NXDOMAIN）的响应会在其它服务器中缓存多长时间。

最大的否定缓存时间是 3 小时（3h）。

$TTL 在区文件顶部（在 SOA 之前）的 $TTL 指令给出对每个没有指定 TTL 集的资源记录一个缺省的 TTL。

RR TTLs 每个资源记录可以有一个以秒为单位的 TTL，它将控制其它服务器可以缓存它多长时间。

所有这三种 TTL 的缺省单位都是秒，不过单位都可以被显式指定，例如，1h30m 。

4.3.4 IPv4 中的反向映射

反向名字解析（即将 IP 地址翻译成名字）是通过使用 in-addr.arpa 域和 PTR 记录来实现的。in-addr.arpa 域中的条目是以自左向右表示从大到小的方式组成的。这与 IP 地址通常书写方式的顺序相反。这样，一个 IP 地址为 10.1.2.3 的机器对应的 in-addr.arpa 名字为 3.2.1.10.in-addr.arpa。这个名字应该有一个 PTR 资源记录，并且其数据字段是机器的名字，如果机器有多个名字，作为可选项，也可以有多个 PTR 记录。例如，在 example.com 域中：

$ORIGIN	2.1.10.in-addr.arpa
3	IN PTR foo.example.com.

注解：这个例子中的 $ORIGIN 行仅用于提供上下文；它不一定出现在实际使用中。它们用于这里，仅仅是指明例子是相对于所列出的起点。

4.3.5 其它区文件指令

主文件格式最初由 RFC 1035 定义，后来被扩展。虽然主文件格式本身是类独立的，但主文件中的所有记录都必须是属于同一个类。

主文件指令包括 $ORIGIN ， $INCLUDE 和 $TTL 。

@ (at 符号)

当 asperand 或 at 符号（@）（译注：即圈 a）用于标记（或名字）字段中时，它表示当前原点。在区文件的开始处，它就是 <zone_name>（后跟一个结尾的点）。

$ORIGIN 指令

语法：$ORIGIN domain-name [comment]

$ORIGIN 设置域名，它将被添加到任何不完整记录的后面。当一个区刚被读入时，有一个隐含的 $ORIGIN <zone_name>. ，（后跟一个结尾的点）。当前的 $ORIGIN 被添加到 $ORIGIN 参数所指定的域名之后，如果它不是一个绝对名字。

```
$ORIGIN example.com.
WWW    CNAME   MAIN-SERVER
```

相当于

```
WWW.EXAMPLE.COM. CNAME MAIN-SERVER.EXAMPLE.COM.
```

$INCLUDE 指令

语法：$INCLUDE filename [origin] [comment]

读入并处理文件 filename ，就像它在这一点包含此文件进来。如果设定了 origin ，文件就使用 $ORIGIN 所设定的值处理；否则，使用当前 $ORIGIN 。

在被包含的文件被读入之后，起点和当前域名恢复到它们 $INCLUDE 之前的值。

注解： RFC 1035 指定了当前起点应该在一个 $INCLUDE 指令之后恢复，但未对当前域名是否恢复作出规定。BIND 9 对两者都恢复。这可能构成 RFC 1035 的一个派生，或者一个特征，也许都是。

$TTL 指令

语法：$TTL default-ttl [comment]

为此命令之后的未定义 TTL 的记录设置缺省的生存期 (TTL)。有效的 TTL 值范围为 0-2147483647 秒。

$TTL 在 RFC 2308 中定义。

4.3.6 BIND 主文件扩展：$GENERATE 指令

语法：$GENERATE range lhs [ttl] [class] type rhs [comment]

$GENERATE 用于建立一系列资源记录，它们仅仅只差别一个循环变量。$GENERATE 可以轻易地生成在 RFC 2317 中所描述的支持/24 之下的反向授权所要求的一系列记录。

```
$ORIGIN 0.0.192.IN-ADDR.ARPA.
$GENERATE 1-2 @ NS SERVER$.EXAMPLE.
$GENERATE 1-127 $ CNAME $.0
```

等效于

```
0.0.0.192.IN-ADDR.ARPA. NS SERVER1.EXAMPLE.
0.0.0.192.IN-ADDR.ARPA. NS SERVER2.EXAMPLE.
1.0.0.192.IN-ADDR.ARPA. CNAME 1.0.0.0.192.IN-ADDR.ARPA.
2.0.0.192.IN-ADDR.ARPA. CNAME 2.0.0.0.192.IN-ADDR.ARPA.
...
127.0.0.192.IN-ADDR.ARPA. CNAME 127.0.0.0.192.IN-ADDR.ARPA.
```

两者都生成 A 和 MX 记录的集合。注意 MX 的右侧是一个被引号包含的字符串。在右侧被处理时，引号会被去掉。

```
$ORIGIN EXAMPLE.
$GENERATE 1-127 HOST-$ A 1.2.3.$
$GENERATE 1-127 HOST-$ MX "0 ."
```

等效于

```
HOST-1.EXAMPLE.  A  1.2.3.1
HOST-1.EXAMPLE.  MX 0 .
HOST-2.EXAMPLE.  A  1.2.3.2
HOST-2.EXAMPLE.  MX 0 .
HOST-3.EXAMPLE.  A  1.2.3.3
HOST-3.EXAMPLE.  MX 0 .
...
HOST-127.EXAMPLE. A  1.2.3.127
HOST-127.EXAMPLE. MX 0 .
```

range 这个可以有两种格式：start-stop 或 start-stop/step。如果使用第一种格式，step 就被设为 1。“start”，“stop” 和 “step” 都必须是介于 0 和 (2^31)-1 之间的正整数。“start” 必须小于等于 “stop”。

owner 描述所建立的资源记录的属主名。任何在 owner 串中的单个 $ （美元符号）都被循环变量所替代。要在输出部份输出 $ ，需要使用一个反斜线 \ 对 $ 进行转义，例如 \$ 。可选地，$ 符号后可以跟修饰符，其作用是改变循环器、宽度和进制的偏移量。

修饰符由一个 { （左花括号）引导，它紧接着 $ 符号，即 ${offset[,width[,base]]} 。例如，${-20,3,d} 从当前值减去 20，打印 “作为十进制数，以 0 填充，宽度为 3” 的结果。可用的输出格式是十进制（d ），八进制（o ），十六进制（x 或 X ，后者为大写输出）和半字节（n 或 N ，后者为大写输出）。

缺省修饰符是 ${0,0,d} 。如果 owner 不是完整名字，就将当前 $ORIGIN 添加在名字后面。

在半字节模式中，值会被当成一个倒置的十六进制串，每个十六进制数字都是一个单独的标记。宽度域包含标记分隔符。

为了对早期版本的兼容，$$ 仍然被识别，指示输出一个字面的 $。

ttl 这指定所生成的记录的生存期。如果未指定，就使用正常的 TTL 继承规则来继承。

class 和 ttl 的位置可以互换。

class 这指定所生成的记录的类。如果指定，必须与区的类一致。

class 和 ttl 的位置可以互换。

type 这可以是任何有效的类型。

rdata 这是一个包含即将创建的资源记录的 RDATA 的字符串。如果其中有空白字符，需要被引号包含；引号不会出现在生成的记录中。

$GENERATE 指令是一个 BIND 的扩展，并不是标准区文件格式的一部份。

4.3.7 附加文件格式

除了标准的文本格式，BIND 9 支持读或者导出其它格式区文件的能力。

raw 格式是区数据的一个二进制表示，类似于在区传送中使用的方式。由于它不要求对文本进行语法分析，装载时间显著缩短。

一个更快的替代方法是 map 格式，它是一个 BIND 9 内存区数据库的镜像；它适合使用 mmap() 函数直接装载到内存中；区几乎可以立即开始对请求提供服务。

对于一个主服务器，一个 raw 或 map 格式的区可以通过 named-compilezone 命令从一个文本区文件生成。对辅服务器或者动态区，它是在 named 完成区传送之后导出区或者应用上次的更新时自动生成的，如果使用 masterfile-format 选项指定了这些格式之中的一种。

如果需要手工修改一个二进制格式的区文件，必须先将其通过 named-compilezone 命令转换为文本格式。所有需要进行的修改都应在文本文件中，然后再使用 named-compilezone 命令将其转换为二进制格式。

注意 map 格式是极其体系相关的。一个 map 文件 **不能**用于一个与生成数据的系统具有不同指针大小、字节序或数据对齐的系统，通常应该只用于同样单个系统内部。由于 raw 格式使用网络字节顺序避免了体系依赖的数据对齐，所以它是尽可能可移植的，但是它还是最好用于同样的单个系统的内部。为导出一个 raw 或者 map 格式的区文件，或者给这样的文件做一个可移植的备份，推荐将其转换为 text 格式。

第 4.4 节 BIND 9 统计

BIND 9 维护许多统计信息并为用户提供几个接口来访问这些统计。可用的统计包括在 BIND 9 中有意义的所有统计计数器，以及其它被认为有用的信息。

统计信息分类成下列部份：

Incoming Requests（**入向请求**） 对每个 OPCODE 的进入的 DNS 请求数目。

Incoming Queries（**入向查询**） 对每个资源记录类型的进入的查询数目。

Outgoing Queries（**出向查询**） 从内部解析器向外发出的对每个资源记录类型的查询数目，每个视图各自维护。

Name Server Statistics（**名字服务器统计**） 关于进入请求处理的统计计数器。

Zone Maintenance Statistics（**区维护统计**） 关于区维护操作，如区传送，的统计计数器。

Resolver Statistics（**解析器统计**） 关于内部解析器所执行的名字解析的统计计数器，每个视图各自维护。

Cache DB RRsets（**缓存数据资源记录集**） Statistics counters related to cache contents, maintained per view. 与缓存内容相关的统计计数器，每个视图各自维护。

"NXDOMAIN" 计数器是被当做不存在被缓存的名字的数目。以资源记录类型命名的计数器指示在缓存数据库中每种类型的活跃资源记录集的数目。

如果一个资源记录类型名字前面有一个惊叹号（!），它表示特定名字的类型不存在也即 "NXRRSET"）。如果一个资源记录类型名字前面有一个井号（#），它表示在缓存中且 TTL 已经过期的这种类型的资源记录集的数目；这些资源记录集仅用于旧回答被开启时。如果一个资源记录类型名字前面有一个波浪号（~），它表示出现在缓存数据库中但被标记为垃圾回收的这种类型的资源记录集的数目；这些资源记录集不能被使用。

Socket I/O Statistics（**套接字 I/O 统计**） 网络相关事件的统计计数器。

对于服务器上的每个权威区，都收集并显示名字服务器统计的一个子集，当 zone-statistics 被设为 full （或 yes ），以便向后兼容。参见options 语句定义和用法 中对 zone-statistics 更详细的描述。

这些统计计数器和它们的区和视图名一起显示。如果服务器没有显式配置视图，视图名被省略。

当前有两种用户接口可以用来访问统计信息。一个是普通文本格式，导出到由 statistics-file 配置选项所指定的文件中；另一个是通过在配置文件中的 statistics-channels 语句所指定的统计通道远程访问。（参见statistics-channels 语句语法 ）。

4.4.1 统计文件

文本格式的统计导出以类似下面这一行开始：

```
+++ Statistics Dump +++ (973798949)
```

括号中的数是标准的 UNIX 风格的时间戳，是自 1970 年 1 月 1 日以来的秒数。后面紧接的行是一个统计信息的集合，其分类在上面已描述。每部份以类似下面这一行开始：

```
++ Name Server Statistics ++
```

每个部份由一些行组成，每行包含统计计数器的值，后面有文本描述。参见下面的可用的计数器。为简短起见，值为 0 的计数器没有显示在统计文件中。

统计导出结束是一个带有与开始行同样数字的行；例如：

--- Statistics Dump --- (973798949)

4.4.2 统计计数器

下表概括了 BIND 9 所提供的统计计数器。对每个计数器，给出了其缩写符号名；这些符号显示在可以通过 HTTP 统计通道访问的统计信息页面上。对计数器的描述，也显示在统计文件上，但是，在本文档中，为了更好的可读性可能做了一点修改。

名字服务器统计计数器

Requestv4 这指示收到的 IPv4 请求数量。注意：这也会记录非查询的请求。

Requestv6 这指示收到的 IPv6 请求数量。注意：这也会记录非查询的请求。

ReqEdns0 这指示收到的带 EDNS(0) 请求的数量。

ReqBadEDN SVer 这指示收到的带有一个不支持的 EDNS 版本的请求的数量。

ReqTSIG 这指示收到的带有 TSIG 的请求的数量。

ReqSIG0 这指示收到的带有 SIG(0) 的请求的数量。

ReqBadSIG 这指示收到的带有不正确（TSIG 或 SIG(0)）签名的请求的数量。

ReqTCP 这指示收到的 TCP 请求的数量。

AuthQryRej 这指示拒绝掉的权威（非递归）请求的数量。

RecQryRej 这指示拒绝掉的递归查询的数量。

XfrRej 这指示拒绝掉的区传送请求的数量。

UpdateRej 这指示拒绝掉的动态更新请求的数量。

Response 这指示已发出的响应的数量。

RespTruncated 这指示已发出的截断响应的数量。

RespEDNS0 这指示已发出的带 EDNS(0) 的响应的数量。

RespTSIG 这指示已发出的带 TSIG 的响应的数量。

RespSIG0 这指示已发出的带 SIG(0) 的响应的数量。

QrySuccess 这指示导致一个成功回答的查询的数量，表示请求返回了一个 NOERROR 应答，并至少带有一个回答资源记录。这相当于先前 BIND 9 版本中的 success 计数器。

QryAuthAns 这指示导致一个权威回答的查询的数量。

QryNoauthAns 这指示导致一个非权威回答的查询的数量。

QryReferral 这指示导致一个引用回答的查询的数量。这相当于先前 BIND 9 版本中的 referral 计数器。

QryNxrrset 这指示导致在 NOERROR 响应中没有数据的查询的数量。这相当于先前 BIND 9 版本中的 nxrrset 计数器。

QrySERVFAIL 这指示导致 SERVFAIL 的查询的数量。

QryFORMERR 这指示导致 FORMERR 的查询的数量。

QryNXDOMAIN 这指示导致 NXDOMAIN 的查询的数量。这相当于先前 BIND 9 版本中的 nxdomain 计数器。

QryRecursion 这指示导致服务器执行递归解析以获取最终回答的查询的数量。这相当于先前 BIND 9 版本中的 recursion 计数器。

QryDuplicate 这指示导致服务器试图进行递归解析但是却发现已经存在一个具有同样 IP 地址、端口、查询 ID、名字、类型和类的查询并且已经被处理过的查询的数量。这相当于先前 BIND 9 版本中的 duplicate 计数器。

QryDropped 这指示服务器所发现的已有的对同一名字、类型和类的递归查询并随后被丢弃的巨量递归查询的数量。这是由于 clients-per-query 和 max-clients-per-query 选项（参见clients-per-query）所解释的原因而被丢弃的查询的数目。这相当于先前 BIND 9 版本中的 dropped 计数器。

QryFailure 这指示其它查询失败的数量。这相当于先前 BIND 9 版本中的 failure 计数器。注意：提供这个计数器主要是为了向后兼容以前的版本。通常情况一些更细粒度的计数器如 AuthQryRej 和 RecQryRej 也落在这个计数器范围中，所以在实际中，这个计数器的意义不大。

QryNXRedir 这指示导致 NXDOMAIN 而被重定向的请求的数量。

QryNXRedirRLookup 这指示导致 NXDOMAIN 而被重定向并且结果是一个成功的远程查询的请求的数量。

XfrReqDone 这指示已完成的所请求的区传送的数量。

UpdateReqFwd 这指示已转发的更新请求的数量。

UpdateRespFwd 这指示已转发的更新响应的数量。

UpdateFwdFail 这指示已失败的动态更新转发的数量。

UpdateDone 这指示已完成的动态更新的数量。

UpdateFail 这指示已失败的动态更新的数量。

UpdateBadPrereq 这指示由于先决条件失败而被拒绝的动态更新的数量。

RateDropped 这指示被比率限制丢弃的响应的数量。

RateSlipped 这指示被比率限制截断的响应的数量。

RPZRewrites 这指示响应策略区重写的数量。

区维护统计计数器

NotifyOutv4 这指示已发送的 IPv4 通知的数量。

NotifyOutv6 这指示已发送的 IPv6 通知的数量。

NotifyInv4 这指示已收到的 IPv4 通知的数量。

NotifyInv6 这指示已收到的 IPv6 通知的数量。

NotifyRej 这指示拒绝掉的进入通知的数量。

SOAOutv4 这指示已发送的 IPv4 SOA 查询的数量。

SOAOutv6 这指示已发送的 IPv6 SOA 查询的数量。

AXFRReqv4 这指示已请求的 IPv4 AXFR。

AXFRReqv6 这指示已请求的 IPv6 AXFR。

IXFRReqv4 这指示已请求的 IPv4 IXFR。

IXFRReqv6 这指示已请求的 IPv6 IXFR。

XfrSuccess 这指示成功的区传送请求的数量。

XfrFail 这指示失败的区传送请求的数量。

解析器统计计数器

Queryv4 这指示已发送的 IPv4 查询的数量。

Queryv6 这指示已发送的 IPv6 查询的数量。

Responsev4 这指示已收到的 IPv4 响应的数量。

Responsev6 这指示已收到的 IPv6 响应的数量。

NXDOMAIN 这指示已收到的 NXDOMAIN 的数量。

SERVFAIL 这指示已收到的 SERVFAIL 的数量。

FORMERR 这指示已收到的 FORMERR 的数量。

OtherError 这指示已收到的其它错误的数量。

EDNS0Fail 这指示 EDNS(0) 查询失败的数量。

Mismatch 这指示已收到的不匹配响应的数量。意谓着 DNS ID，响应的源地址和/或响应的源端口与期望的不匹配。（端口必须是 53 或 port 所定义的。）这可能暗示一个缓存污染攻击。

Truncated 这指示已收到的截断响应的数量。

Lame 这指示已收到的不完整授权的数量。

Retry 这指示执行过的查询重试的数量。

QueryAbort 这指示由于配额控制而被终止的查询的数量。

QuerySockFail 这指示打开查询套接字时的失败的数量。这类失败的一个通常原因是由于在文件描述符上的限制。

QueryTimeout 这指示查询超时的数量。

GlueFetchv4 这指示发起的取 IPv4 NS 的地址操作的数量。

GlueFetchv6 这指示发起的取 IPv6 NS 的地址操作的数量。

GlueFetchv4Fail 这指示失败的取 IPv4 NS 的地址操作的数量。

GlueFetchv6Fail 这指示失败的取 IPv6 NS 的地址操作的数量。

ValAttempt 这指示试探过的 DNSSEC 验证的数量。

ValOk 这指示成功的 DNSSEC 验证的数量。

ValNegOk 这指示在否定信息上成功的 DNSSEC 验证的数量。

ValFail 这指示失败的 DNSSEC 验证的数量。

QryRTTnn 这提供请求往返时间（RTT）的频率表。每个 nn 指定相应的频率。在序列 nn_1，nn_2`，…，nn_m 中，nn_i 的值是其 RTT 在 nn_(i-1)（含）和 nn_i（不含）之间毫秒的请求的

数目。为方便起见，我们将 nn_0 定义为 0。最后的条目应该表示为 nn_m+ ，它表示其 RTT 等于或大于 nn_m 毫秒的请求的数目。

套接字 I/O 统计计数器

套接字 I/O 统计计数器是按每种套接字类型定义的，它们是 UDP4（UDP/IPv4），UDP6（UDP/IPv6），TCP4 （TCP/IPv4），TCP6 （TCP/IPv6），Unix （Unix Domain）和 FDwatch （朝套接字模块外打开的套接字）。在下面的表中 <TYPE> 表示一个套接字类型。并非所有套接字都可以使用所有的计数器；在描述字段会说明例外情况。

<TYPE>Open 这指示套接字成功打开的数量。这个计数器不适用于 FDwatch 类型。

<TYPE>OpenFail 这指示套接字打开失败的数量。这个计数器不适用于 FDwatch 类型。

<TYPE>Close 这指示套接字关闭的数量。

<TYPE>BindFail 这指示绑定套接字失败的数量。

<TYPE>ConnFail 这指示连接套接字失败的数量。

<TYPE>Conn 这指示成功建立连接的数量。

<TYPE>AcceptFail 这指示接受进入连接请求失败的数量。这个计数器不能用于 UDP 和 FDwatch 类型。

<TYPE>Accept 这指示成功接受进入连接请求的数量。这个计数器不能用于 UDP 和 FDwatch 类型。

<TYPE>SendErr 这指示套接字发送操作中的错误的数量。

<TYPE>RecvErr 这指示套接字接收操作中的错误的数量。这包括在一个收到 ICMP 错误消息通知的已连接 UDP 套接字上发送操作的错误。

第五章　高级 DNS 特征

第 5.1 节　通知（Notify）

DNS NOTIFY 是一个让主服务器通知其辅服务器某个区数据发生了变化的机制。作为对来自主服务器的 NOTIFY 的响应，辅服务器会检查自己的区版本是否是当前版本，如果不是，就发起区传送。

更多的关于 DNS NOTIFY 的信息，参见布尔选项 中对 notify 选项的描述以及区传送 中对区选项 also-notify 的描述。NOTIFY 协议由 RFC 1996 规定。

注解：一个辅服务器也可以成为其它辅服务器的主服务器，named 在缺省情况下会对其所载入的每个区发出 NOTIFY 消息。指定 notify master-only; 将使 named 只对其所载入区的主服务器发出 NOTIFY 。

第 5.2 节　动态更新（Dynamic Update）

动态更新是一个通过给主服务器发送一个特殊格式的 DNS 消息来增加、替换和删除其上记录的方法。这些消息的格式和含义由 RFC 2136 指定。

动态更新是通过在 zone 语句中包含一个 allow-update 或一个 update-policy 子句来打开的。

如果区的 update-policy 被设为 local，对区的更新将由密钥 local-ddns 来许可，后者是由 named 启动时生成的。更多详细信息，参见动态更新策略 。

使用 Kerberos 签名请求进行的动态更新可以通过使用 TKEY/GSS 协议来完成，要么是设置 tkey-gssapi-keytab 选项，要么是同时设置 tkey-gssapi-credential 和 tkey-domain 这两个选项。一旦开启这个功能，Kerberos 签名请求将与区的更新策略进行匹配，使用 Kerberos principal 作为请求的签名者。

更新一个安全的区（使用 DNSSEC 的区）遵循 RFC 3007：受更新影响的 RRSIG、NSEC 和 NSEC3

记录通过服务器使用一个在线区密钥来自动重新生成。更新授权是基于事务签名和一个显式的服务器策略。

5.2.1 日志文件

一个使用动态更新的区的所有变化将保存在这个区的日志文件中。这个文件在第一个更新发生时自动创建。日志文件的名字由相关区文件的名字加上扩展名 .jnl 组成，除非指定了名字。日志文件是二进制格式，不能手工编辑。

服务器会时常将更新后的区的完整内容转储到（“dump”）其区文件中。这个操作不是每次动态更新后立即执行，因为如果这样，一个大的区在频繁更新时将会导致服务器变得太慢。代替的方法是，转储操作延迟 15 分钟，以允许其它更新。在转储过程中，将会创建以 .jnw 和 .jbk 为后缀的瞬时文件；在普通情况下，这些文件将在转储完成后被删除掉，忽略它们也是安全的。

当服务器在停止或宕掉之后重启时，将重新装载日志文件，以便将自从上次区转储以后发生的所有更新合并到区中。

增量区传送所产生的变化也是以类似的方式记录到日志的。

动态更新区的区文件不能像通常那样进行手工编辑，因为没法保证包含最近的动态变化—这些变化只存在于日志文件中。保持动态更新区的区文件最新的唯一方法是执行 rndc stop 。

要手工修改一个动态区，按照下列步骤：首先，使用 rndc freeze zone 关闭到这个区的动态更新；这会使用存放在其 .jnl 文件中的变化来更新区的主文件。然后，编辑区文件。最后，执行 rndc thaw zone 重新装载修改后的区文件并打开动态更新。

rndc sync zone 将使用日志文件中的变化更新区文件而不停止动态更新；这对观察当前区状态是有用的。要在更新区文件之后删除 .jnl 文件，使用 rndc sync -clean 。

第 5.3 节　增量区传送（IXFR）

增量区传送（incremental zone transfer, IXFR）协议是一个为辅服务器提供只传送变化的数据的方式，以代替必须传送整个区的方法。IXFR 协议由 RFC 1995 指定。参见建议标准 。

当作为一个主服务器时，BIND 9 支持对这些区的 IXFR，条件是必要的变化历史信息是可用的。这些包括以动态更新维护的主区和通过 IXFR 获取数据的辅区。对于手工维护的主区，以及通过全量区传送（AXFR）获取数据的辅区，IXFR 仅在 ixfr-from-differences 选项被设置成 yes 时才支持。

当作为一个辅服务器时，BIND 9 将尝试使用 IXFR，除非其被显式关闭。更多关于关闭 IXFR 的信息，参见 server 语句的 request-ixfr 子句的描述。

当一台辅服务器通过 AXFR 接收一个区，它会创建一个新的区数据库副本，然后将其交换到适当的位置；在装载过程中，请求继续由旧数据库服务而不受干扰。但是，当通过 IXFR 接收一个区，修改应用到正在运行的区，可能会在传输期间降低查询性能。如果一台收到一条 IXFR 请求的服务器判断响应的大小类似于一条 AXFR 响应的大小，它可能希望发送 AXFR 作为替代。可以使用 max-ixfr-ratio 选项配置这个判断的阈值。

第 5.4 节　分割 DNS

对 DNS 空间的内部和外部解析器设置不同的视图通常被称为一个分割的 DNS 设置。有几个原因使某个组织想要这样设置其 DNS。

一个使用分割 DNS 的共同的原因是对“外部”互联网上的客户隐藏“内部”DNS 信息。关于这样做是否确实有用，有一些争议。内部 DNS 信息以多种渠道泄露（例如，通过电子邮件头部），并且大多数聪明的“攻击者”可以使用其它手段来取得他们所需要的信息。无论如何，由于列出外部客户端不可能访问到的内部服务器地址可以导致连接延迟和其它烦恼，一个组织可以选择使用分割 DNS 来向外部世界提供一个一致的自身视图。

另一个设置一个分割 DNS 系统的共同的原因是允许在过滤器之后或在 RFC 1918 空间（保留 IP 空间，在 RFC 1918 中说明）中的内部网络去查询互联网上的 DNS。分割 DNS 也可以用于允许来自外部的回复邮件进入内部网络。

5.4.1　分割 DNS 设置的例子

让我们假设一个名叫 Example, Inc. 的公司（example.com ）有几个公司站点，有一个使用保留地址空间的内部网络和一个外部停火区（DMZ），或称为网络的“外部”部份，是外部可以访问到的。

Example 公司想要其内部的客户端可以解析外部主机名并同外面的人们交换电子邮件。公司也想要其内部的解析器可以访问某些内部才有的区，这些区对内部网络之外的用户不可用。

为达到这个目标，公司设置两台名字服务器。一台在内部网（在保留地址空间），另一台在 DMZ 中的堡垒主机上，堡垒主机是一个“代理”主机，它可以同其两侧的网络通信。

内部服务器配置成除了对 site1.internal ，site2.internal ，site1.example.com 和 site2.example.com 之外的所有请求都转发给在 DMZ 的服务器。这些内部服务器都有关于 site1.example.com ，site2.example.com ， site1.internal 和 site2.internal 的全部信息。

为保护 site1.internal 和 site2.internal 域，内部服务器必须配置成不允许任何外部主机对这些域的请求来访问，其中也包括堡垒主机。

在堡垒主机上的外部服务器被配置成服务于 site1.example.com 和 site2.example.com 区的“公共”版本。其中可能包括这样的一些公共服务器（www.example.com 和 ftp.example.com ）的主机记录以及邮件交换（MX）记录（a.mx.example.com 和 b.mx.example.com ）。

另外，公共的 site1.example.com 和 site2.example.com 区应该有包括通配（* ）记录的特定 MX 记录指向堡垒主机。这是必须的，因为外部邮件服务器没有其它任何方式来查到如何将邮件投递到那些内部的主机。使用通配记录，邮件可以投递到堡垒主机，堡垒主机再将邮件转发到内部主机。

这里是一个通配 MX 记录的例子：

```
* IN MX 10 external1.example.com.
```

堡垒主机代表内部网络的任何主机接收邮件，它需要指定如何将邮件投递到内部主机。堡垒主机上的解析器需要配置成指向内部名字服务器，以完成 DNS 解析。

对内部主机的查询请求将由内部服务器回答，对外部主机的查询请求将转发到堡垒主机上的 DNS 服务器。

要让所有这些正常工作，内部客户端需要配置成 **只**请求内部名字服务器。这个也可以通过网络上的选择性过滤器来进行加强。

如果所有的东西都正确设置，Example 公司的内部客户端将能够：

- 查询 site1.example.com 和 site2.example.com 区上的所有主机。
- 查询 site1.internal 和 site2.internal 域上的所有主机。
- 查询互联网上的任何名字。
- 同内部网和外部网上的用户交换电子邮件。

互联网上的主机将能够：

- 查询 site1.example.com 和 site2.example.com 区上的所有主机。
- 同 site1.example.com 和 site2.example.com 区上的任何人交换电子邮件。

这里是对我们上述描述进行配置的一个例子。注意这里仅仅有配置信息；对于如何配置你的区文件的信息，参见样例配置 。

内部 DNS 服务器配置：

```
acl internals { 172.16.72.0/24; 192.168.1.0/24; };

acl externals { bastion-ips-go-here; };
```

（下页继续）

(续上页)

```
options {
   ...
   ...
   forward only;
   // forward to external servers
   forwarders {
   bastion-ips-go-here;
   };
   // sample allow-transfer (no one)
   allow-transfer { none; };
   // restrict query access
   allow-query { internals; externals; };
   // restrict recursion
   allow-recursion { internals; };
   ...
   ...
};

// sample primary zone
zone "site1.example.com" {
 type master;
 file "m/site1.example.com";
 // do normal iterative resolution (do not forward)
 forwarders { };
 allow-query { internals; externals; };
 allow-transfer { internals; };
};

// sample secondary zone
zone "site2.example.com" {
 type slave;
 file "s/site2.example.com";
 masters { 172.16.72.3; };
 forwarders { };
 allow-query { internals; externals; };
 allow-transfer { internals; };
};

zone "site1.internal" {
```

(下页继续)

(续上页)

```
 type master;
 file "m/site1.internal";
 forwarders { };
 allow-query { internals; };
 allow-transfer { internals; }
};

zone "site2.internal" {
 type slave;
 file "s/site2.internal";
 masters { 172.16.72.3; };
 forwarders { };
 allow-query { internals };
 allow-transfer { internals; }
};
```

外部（堡垒主机）DNS 服务器配置：

```
acl internals { 172.16.72.0/24; 192.168.1.0/24; };

acl externals { bastion-ips-go-here; };

options {
 ...
 ...
 // sample allow-transfer (no one)
 allow-transfer { none; };
 // default query access
 allow-query { any; };
 // restrict cache access
 allow-query-cache { internals; externals; };
 // restrict recursion
 allow-recursion { internals; externals; };
 ...
 ...
};

// sample secondary zone
zone "site1.example.com" {
 type master;
```

(下页继续)

(续上页)

```
 file "m/site1.foo.com";
 allow-transfer { internals; externals; };
};

zone "site2.example.com" {
 type slave;
 file "s/site2.foo.com";
 masters { another_bastion_host_maybe; };
 allow-transfer { internals; externals; }
};
```

堡垒主机上的 resolv.conf（或等效的）文件：

```
search ...
nameserver 172.16.72.2
nameserver 172.16.72.3
nameserver 172.16.72.4
```

第 5.5 节　TSIG

TSIG（Transaction SIGnatures，事务签名）是一个认证 DNS 消息的机制，最早在 RFC 2845 中规定。它允许使用一个共享密钥对 DNS 消息加密签名。TSIG 可以被用于任何 DNS 事务，可以作为一种限制授权的客户端对某个服务器功能访问（例如，递归请求）的方法，这时基于 IP 的访问控制已无法满足需求或者需要被覆盖，或者作为一种确保消息认证的方法，这时其对服务器完整性非常关键，例如配合动态更新消息或者从一个主服务器到一个辅服务器的区传送。

这个部份是一个在 BIND 中设置 TSIG 的指导。它描述了配置语法和创建 TSIG 密钥的过程。

named 支持 TSIG 用于服务器到服务器的通信，一些 BIND 附带工具也支持 TSIG 用于发送消息给 named：

- nsupdate - 动态 DNS 更新工具 支持 TSIG，通过 -k ，-l 和 -y 命令行选项，或在交互运行方式下通过 key 命令。
- dig - DNS 查找工具 支持 TSIG，通过 -k 和 -y 命令行选项。

5.5.1 生成一个共享密钥

TSIG 密钥可以使用 tsig-keygen 命令生成；命令的输出是一个适合放入 named.conf 的 key 指令。密钥名，算法名和大小可以由命令行参数指定；缺省分别是“tsig-key”，HMAC-SHA256 和 256 位。

任意一个有效 DNS 名字都可以用作一个密钥名，例如，在服务器 host1 和 host2 之间共享的密钥可以叫做“host1-host2.”，这个密钥可以使用

```
$ tsig-keygen host1-host2. > host1-host2.key
```

生成。

然后这个密钥被拷贝到两台主机上。在两台主机上密钥名和密码必须一致。（注意：从一台服务器到另一台服务器拷贝一个共享密码超出了 DNS 的范围。应该使用一个安全的传输机制：安全 FTP，SSL，ssh，电话，加密电子邮件等等。）

tsig-keygen 也可以被用作 ddns-confgen，在这种情况下，其输出包括在 named 中设置动态 DNS 所用的附加配置。参见ddns-confgen - ddns 密钥生成工具 获取详细信息。

5.5.2 装载一个新密钥

如果一个密钥在服务器 host1 和 host2 之间共享，每个服务器的 named.conf 文件可以添加如下内容：

```
key "host1-host2." {
  algorithm hmac-sha256;
  secret "DAopyf1mhCbFVZw7pgmNPBoLUq8wEUT7UuPoLENP2HY=";
};
```

（这是上面使用 tsig-keygen 生成的同一个密钥。）

由于这段文本包含一个密码，推荐无论是 named.conf 还是存放 key 指令的文件都不是任何人可读的，后者是通过 include 指令包含进 named.conf 。

一旦一个密钥被添加到 named.conf，服务器要重启或重新读入配置，才能识别密钥。如果服务器收到密钥签名的消息，它将能够验证签名。如果签名有效，响应将会使用同一个密钥签名。

为一台服务器所知的 TSIG 密钥可以使用命令 rndc tsig-list 列出。

5.5.3 指示服务器使用一个密钥

一个向其它服务器发送请求的服务器必须被告知是否使用一个密钥，以及如果使用，使用哪个密钥。

例如，必须为一个辅区定义中的 masters 语句中的每一个服务器指定一个密钥；在这种情况下，所有 SOA 请求消息，NOTIFY 消息和区传送请求（AXFR 或 IXFR）都将使用指定密钥签名。密钥也可在一个主区或辅区中的 also-notify 语句中指定，导致 NOTIFY 消息使用指定的密钥签名。

密钥也可以在一个 server 指令中指定。添加下列内容到 host1 ，如果 host2 的 IP 地址是 10.1.2.3，将会导致 **所有**从 host1 到 host2 的请求，包括普通 DNS 请求，使用 host1-host2. 密钥签名：

```
server 10.1.2.3 {
  keys { host1-host2. ;};
};
```

keys 语句中可以提供多个密钥，但只能使用第一个。由于这条指令不包含任何密码，所以它可以放在一个所有人可读的文件中。

从 host2 发向 host1 的消息将 **不会**被签名，除非在 host2 的配置文件中有一个类似的 server 指令。

无论何时，任何服务器只要发出了一个 TSIG 签名的 DNS 请求，它将会期待响应被同一个密钥签名。如果响应未被签名，或者签名无效，响应将会被拒绝。

5.5.4 基于 TSIG 的访问控制

TSIG 密钥可以在 ACL 定义和诸如 allow-query ，allow-transfer 和 allow-update 的 ACL 指令中指定。上述密钥在一个 ACL 元素中被表示成 key host1-host2. 。

这里是一个 allow-update 指令使用一个 TSIG 密钥的例子：

```
allow-update { !{ !localnets; any; }; key host1-host2. ;};
```

这允许动态更新仅仅在 UPDATE 请求来自一个 localnets 内的地址，**并且**它使用 host1-host2. 密钥签名时才会成功。

参见*动态更新策略* 查看更为灵活的 update-policy 语句的讨论。

5.5.5 错误

处理 TSIG 签名消息时可能产生一些错误：

- 如果一个知道 TSIG 的服务器收到一个它所不知道的密钥的签名消息，响应将不会被签名，TSIG 扩展错误码将被设置为 BADKEY。
- 如果一个知道 TSIG 服务器收到一个它所知道密钥的却无效的签名消息，响应将不会被签名，TSIG 扩展错误码将被设置为 BADSIG。
- 如果一个知道 TSIG 的服务器收到一个允许的时间范围之外的消息，响应将会被签名，TSIG 扩展错误码将被设置为 BADTIME，并且时间值将会被调整以使响应能够被成功验证。

在以上所有情况中，服务器都将返回一个 NOTAUTH 响应码（未认证的）。

第 5.6 节　TKEY

TKEY（事务 KEY）是一个用于在两台主机之间自动协商一个共享密码的机制，最早在 RFC 2930 中规定。

有几种 TKEY "模式" 来指定如何生成和分派一个密钥。BIND 9 只实现了这些模式中的一种：Diffie-Hellman 密钥交换。两台主机都需要有一个带 DH 算法的 KEY 记录（虽然并不要求这个记录出现在一个区中）。

TKEY 过程由一个客户端或服务器通过发出一个 TKEY 类型的请求到一个知道 TKEY 的服务器开始。请求必须在附加部分包括一个合适的 KEY 记录，并且必须用 TSIG 或者 SIG(0) 使用先前建立的密钥签名。服务器的响应，如果成功，必须在其回答部分包括一个 TKEY 记录。在这个事务之后，两个参与方都有足够的信息使用 Diffie-Hellman 密钥交换计算一个共享密码。共享密码就能被用于两台服务器之间签名随后的事务。

服务器所知的 TSIG 密钥，包括 TKEY 协商的密钥，可以使用 rndc tsig-list 列出。

TKEY 协商的密钥可以使用 rndc tsig-delete 从一个服务器删除。这也可以经由 TKEY 协议自身，通过发送一个认证的 TKEY 请求指定 "key deletion" 模式来完成。

第 5.7 节　SIG(0)

BIND 部份支持 RFC 2535 和 RFC 2931 中所指定的 DNSSEC SIG(0) 事务签名。SIG(0) 使用公钥/私钥来认证消息。访问控制的执行与 TSIG 密钥类似；可以基于密钥名授予或拒绝权限。

当收到一个 SIG(0) 签名消息时，仅仅在服务器知道并相信密钥的情况下才会验证它；服务器不会试图递归获取或验证密钥。

多个消息的 TCP 流的 SIG(0) 签名还不支持。

随同 BIND 9 发行的生成 SIG(0) 签名消息的唯一工具是 nsupdate 。

第 5.8 节　DNSSEC

通过 RFC 4033 、RFC 4034 和 RFC 4035 所定义的 DNS 安全（DNSSEC-bis）扩展，DNS 信息的加密认证是可能做到的。本节描述了建立和使用 DNSSEC 对区签名。

为了设置一个 DNSSEC 安全的区，有一系列必须遵循的步骤。BIND 9 在这个过程中用到了几个附带的工具，这将在下面详细解释。在所有的情况下，-h 选项都会打印除一个完整的参数清单。注意，DNSSEC 工具要求密钥集文件在工作目录或者在由 -d 选项所指定的目录中，而且 BIND 9.2.x 及更早的版本中所附带的工具与当前版本所附带的不兼容。

必须同父区和/或子区的管理员通信，以传送密钥。一个区的安全状态必须由其父区指明，以使支持 DNSSEC 的解析器相信它所得到的数据。这是通过在授权点提供或者不提供一个 DS 记录来完成的。

对其它信任这个区的数据的服务器，它们必须静态地配置这个区的区密钥或者在 DNS 树中这个区上面的另一个区的区密钥。

5.8.1　生成密钥

dnssec-keygen 程序用于生成密钥。

一个加密的区必须包含一个或多个区密钥。区密钥对区中的所有其它记录签名，也对任何安全授权的区的区密钥签名。区密钥必须有一个与区同样的名字，一个为 ZONE 的类型名，并且可以用于认证。推荐区密钥使用由 IETF 作为 “强制实现” 所指定的加密算法；当前有两种算法：RSASHA256 和 ECDSAP256SHA256。推荐 ECDSAP256SHA256 用于当前和将来的部署。

下列命令为 child.example 区生成一个 ECDSAP256SHA256 密钥：

```
dnssec-keygen -a ECDSAP256SHA256 -n ZONE child.example.
```

将产生两个输出文件：Kchild.example.+013+12345.key 和 Kchild.example.+013+12345.private（这里的 12345 是密钥标记的一个例子）。密钥文件名包含密钥名（child.example. ），算法（5 表示 RSASHA1，8 表示 RSASHA256，13 表示 ECDSAP256SHA256，15 表示 ED25519，等等）和密钥标记（在本例中为 12345）。私钥（在 .private 文件中）用于生成签名，公钥（在 .key 文件中）用于验证签名。

要生成同样属性的另一个密钥但是使用一个不同的密钥标记，只需重复上面的命令。

dnssec-keyfromlabel 程序是用来从一个隐藏硬件获取一个密钥对并生成密钥文件。其用法与 dnssec-keygen 类似。

公钥应该通过使用 $INCLUDE 语句来包含 .key 文件的方式插入到区文件中。

5.8.2 对区签名

dnssec-signzone 程序用于对区签名。

任何与安全子区相关的 keyset 文件都应该出现。区的签名者将为区生成 NSEC 、NSEC3 和 RRSIG 记录，如同指定 -g 选项时为子区生成的 DS 记录一样。如果未指定 -g 选项，就必须为安全的子区手工添加 DS 资源记录集。

缺省情况下，所有的区密钥都有一个可用的私钥用来生成签名。下列命令对区签名，假设它是一个名为 zone.child.example 的文件。

dnssec-signzone -o child.example zone.child.example

会产生一个输出文件：zone.child.example.signed 。这个文件必须在 named.conf 中作为输入文件被引用。

dnssec-signzone 也会生成一个 keyset 和 dsset 文件。它们用于向上级区管理员提供 DNSKEY（或者与之相关的 DS 记录），作为本区的安全入口点。

5.8.3 为 DNSSEC 配置服务器

为了使 named 验证来自其它服务器的响应，必须将 dnssec-validation 选项设置为 yes 或 auto 。

当 dnssec-validation 被设置为 auto 时，就会自动使用 DNS 根区的一个信任锚。这个信任锚作为 BIND 的一部份提供并使用 RFC 5011 密钥管理来保持更新。

当 dnssec-validation 被设置为 yes 时，仅当在 named.conf 中使用一个 trust-anchors 语句（或者 managed-keys 和 trusted-keys 语句，这两者都已被废弃了）至少显式配置了一个信任锚，才会进行 DNSSEC 验证。

当 dnssec-validation 被设置为 no 时，不会进行 DNSSEC 验证。

缺省值是 auto ，除非 BIND 编译时带有 configure --disable-auto-validation ，这种情况下缺省值是 yes 。

trust-anchors 中指定的密钥是区的 DNSKEY 资源记录的拷贝，它用于形成加密信任链的首个链接。使用关键字 static-key 或 static-ds 配置的密钥被直接装载到信任锚的表中，并且只能通过变

更配置来修改。使用 initial-key 或 initial-ds 配置的密钥用于初始化 RFC 5011 信任锚维护，并在 named 首次运行之后自动保持更新。

在本文档的后面有更多关于 trust-anchors 的详细描述。

BIND 9 在装载时不验证签名，所以不必在配置文件中指定权威区的区密钥。

在 DNSSEC 建立之后，一个典型的 DNSSEC 配置看起来就像以下内容。它有一个或多个根的公钥。这可以允许来自外部的响应被验证。本单位所控制的部份名字空间也需要几个密钥。其作用是确保 named 不受上级区安全的 DNSSEC 组成中的危险因素的影响。

```
trust-anchors {
  /* Root Key */
  "." initial-key 257 3 3 "BNY4wrWM1nCfJ+CXd0rVXyYmobt7sEEfK3clRbGaTwS
          JxrGkxJWoZu6I7PzJu/E9gx4UC1zGAHlXKdE4zYIpRh
          aBKnvcC2U9mZhkdUpd1Vso/HAdjNe8LmMlnzY3zy2Xy
          4klWOADTPzSv9eamj8V18PHGjBLaVtYvk/ln5ZApjYg
          hf+6fElrmLkdaz MQ2OCnACR817DF4BBa7UR/beDHyp
          5iWTXWSi6XmoJLbG9Scqc7l70KDqlvXR3M/lUUVRbke
          g1IPJSidmK3ZyCllh4XSKbje/45SKucHgnwU5jefMtq
          66gKodQj+MiA21AfUVe7u99WzTLzY3qlxDhxYQQ20FQ
          97S+LKUTpQcq27R7AT3/V5hRQxScINqwcz4jYqZD2fQ
          dgxbcDTClU0CRBdiieyLMNzXG3";
  /* Key for our organization's forward zone */
  example.com. static-ds 54135 5 2
↪"8EF922C97F1D07B23134440F19682E7519ADDAE180E20B1B1EC52E7F58B2831D"

  /* Key for our reverse zone. */
  2.0.192.IN-ADDRPA.NET. static-key 257 3 5 "AQOnS4xn/IgOUpBPJ3bogzwc
            xOdNax071L18QqZnQQQAVVr+i
            LhGTnNGp3HoWQLUIzKrJVZ3zg
            gy3WwNT6kZo6c0tszYqbtvchm
            gQC8CzKojM/W16i6MG/eafGU3
            siaOdS0yOI6BgPsw+YZdzlYMa
            IJGf4M4dyoKIhzdZyQ2bYQrjy
            Q4LB0lC7aOnsMyYKHHYeRvPxj
            IQXmdqgOJGq+vsevG06zW+1xg
            YJh9rCIfnm1GX/KMgxLPG2vXT
            D/RnLX+D3T3UL7HJYHJhAZD5L
            59VvjSPsZJHeDCUyWYrvPZesZ
            DIRvhDD52SKvbheeTJUm6Ehkz
            ytNN2SN96QRk8j/iI8ib";
```

（下页继续）

(续上页)

```
};

options {
  ...
  dnssec-validation yes;
};
```

注解：本例中列出的所有密钥都是无效的。特别是，根密钥是无效的。

当 DNSSEC 验证被打开并正确地配置后，解析器将会拒绝来自已签名的、安全的区中未通过验证的响应，并返回 SERVFAIL 给客户端。

响应可能因为以下任何一种原因而验证失败，包含错误的、过期的、或无效的签名，密钥与父区中的 DS 资源记录集不匹配，或者来自一个区的不安全的响应，而根据它的父区，应该是一个安全的响应。

注解：当验证者收到一个来自一个拥有签名父区的未签名区的响应，它必须向其父区确认这个是有意未签名的。它通过验证父区没有包含子区的 DS 记录，即通过签名的和验证了的 NSEC/NSEC3 记录，来确认这一点。

如果验证者 **能够**证明区是不安全的，其响应就是可以接受的。然而，如果不能证明，它就必须假设不安全的响应是伪造的；它就拒绝响应并在日志中记录一个错误。

日志记录的错误为 “insecurity proof failed” 和 “got insecure response; parent indicates it should be secure”。

第 5.9 节　DNSSEC，动态区，和自动化签名

5.9.1 从不安全转换到安全

有三种方法可以将一个区从不安全转换为安全：使用动态 DNS 更新，使用 auto-dnssec 区选项，或使用 dnssec-policy 为区设置一个 DNSSEC 策略。

对每一种方法，都需要配置 named，使其能够看到 K* 文件，而后者包含签名区时会用到的公钥和私钥部份。这些文件由 dnssec-keygen 生成，或者当使用了 dnssec-policy 时由 named 在需要时

创建。密钥应当放到在 named.conf 中所指定的密钥目录中：

```
zone example.net {
  type master;
  update-policy local;
  file "dynamic/example.net/example.net";
  key-directory "dynamic/example.net";
};
```

如果生成了一个 KSK 和一个 ZSK DNSKEY 密钥，这个配置将使区中所有的记录被 ZSK 签名，并使 DNSKEY 资源记录集被 KSK 签名。作为初始签名过程的一部份，还会生成一个 NSEC 链。

使用 dnssec-policy ，可以指定哪些密钥应该是 KSK 和/或 ZSK。要用一个密钥对所有记录签名，必须指定一个 CSK。例如：

```
dnssec-policy csk {
  keys {
    csk lifetime unlimited algorithm 13;
  };
};
```

5.9.2 动态 DNS 更新方法

要通过动态更新插入密钥：

```
% nsupdate
> ttl 3600
> update add example.net DNSKEY 256 3 7␣
↪AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EyAGfBfL8eQ8al35zz3Y I1m/
↪SAQBxIqMfLtIwqWPdgthsu36azGQAX8=
> update add example.net DNSKEY 257 3 7 AwEAAd/7odU/64o2LGsifbLtQmtO8dFDtTAZXSX2+X3e/
↪UNlq9IHq3Y0 XtC0Iuawl/qkaKVxXe2lo8Ct+dM6UehyCqk=
> send
```

虽然更新请求会几乎立即完成，但是只有在 named 有时间扫描整个区并生成 NSEC 和 RRSIG 记录之后，区的签名才会完成。区顶点的 NSEC 记录会在最后添加，作为一个完整 NSEC 链的信号。

要使用 NSEC3 来取代 NSEC 作签名，应该添加一条 NSEC3PARAM 记录到初始更新请求中。NSEC3 链中的 OPTOUT 位可以在 NSEC3PARAM 记录的标志字段中设置。

```
% nsupdate
> ttl 3600
> update add example.net DNSKEY 256 3 7␣
↪AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EyAGfBfL8eQ8al35zz3Y I1m/
↪SAQBxIqMfLtIwqWPdgthsu36azGQAX8=
> update add example.net DNSKEY 257 3 7 AwEAAd/7odU/64o2LGsifbLtQmtO8dFDtTAZXSX2+X3e/
↪UNlq9IHq3Y0 XtC0Iuawl/qkaKVxXe2lo8Ct+dM6UehyCqk=
> update add example.net NSEC3PARAM 1 1 100 1234567890
> send
```

再次强调，更新请求将会几乎立即完成；然而，所添加的记录不会马上可见，直到 named 有机会建立/删除相关的链。一个私有类型的记录将被创建，以记录操作状态（参见下面更详细的描述），并在操作完成之后被删除。

当初始签名及 NSEC/NSEC3 链正在生成时，其它更新也可能发生。

5.9.3 完全自动化区签名

要打开自动签名，建立一个 dnssec-policy 或在 named.conf 的区语句中增加 auto-dnssec 选项。auto-dnssec 有两个可能的参数：allow 或 maintain 。

使用 auto-dnssec allow ，named 可以在密钥目录中查找与区匹配的密钥，将其插入到区中，并使用它们来签名区。它仅仅在其收到一个 rndc sign <zonename> 时才这样做。

auto-dnssec maintain 包含上述功能，而且还可以根据密钥的时间元数据的时间表自动调整区的 DNSKEY 记录。（更多信息参见dnssec-keygen: DNSSEC 密钥生成工具 和dnssec-settime: 为一个 DNSSEC 密钥设置密钥定时元数据 。）

dnssec-policy 与 auto-dnssec maintain 类似，而 dnssec-policy 也能在需要时自动创建新密钥。此外，任何与 DNSSEC 签名相关的配置都从策略中提取，而忽略现存的 DNSSEC named.conf 选项。

named 将会定期搜索密钥目录查找与区匹配的密钥，如果密钥的元数据显示区发生了任何变化，诸如增加，删除或者撤销一个密钥，这个动作都会被执行。缺省时，每 60 分钟检查一次密钥目录；这个周期可以通过 dnssec-loadkeys-interval 调整，最大到 24 小时。rndc loadkeys 强制 named 立即检查密钥是否更新。

如果密钥被提供到密钥目录中，区第一次装载时，区就会立刻被签名，而不用等待 rndc sign 或 rndc loadkeys 命令。这些命令仍然可以用于计划外的密钥变更时。

当新的密钥被添加到一个区时，TTL 被设置为与任何已存在的 DNSKEY 资源记录集的 TTL 相匹

配。如果不存在 DNSKEY 资源记录集，TTL 将被设置为密钥创建时（使用 dnssec-keygen -L 选项）所指定的 TTL，或者为 SOA 的 TTL，如果有的话。

要使用 NSEC3 来取代 NSEC 作签名，需要在发布和激活密钥之前通过动态更新提交一个 NSEC3PARAM 记录。NSEC3 链中的 OPTOUT 位可以在 NSEC3PARAM 记录的标志字段中设置。NSEC3PARAM 记录将不会立即出现在区中，但它将被存储以供之后参考。当区被签名并且 NSEC3 链完成之后，NSEC3PARAM 将会出现在区中。

使用 auto-dnssec 选项要求区被配置成允许动态更新，这是通过在区配置中增加一个 allow-update 或 update-policy 语句来实现的。如果没有这个，配置就会失败。

5.9.4 私有类型记录

签名过程的状态由私有类型记录（带有一个缺省值 65534）发信号通知。当签名完成，这些带有非零初始字节的记录将会在最后一个字节有一个非零值。

如果一个私有类型记录的第一个字节不为 0，这个记录表明，要么区需要由与记录匹配的密钥来签名，要么与记录匹配的所有签名应当被删掉。

algorithm (octet 1)

key id in network order (octet 2 and 3)

removal flag (octet 4)

complete flag (octet 5)

只有被标志为“complete”的记录才能通过动态更新被删除。删除其它私有类型记录的企图将被静默地忽略掉。

如果第一个字节为零（这是一个保留的算法号，从来不会出现在一个 DNSKEY 记录中），这个记录指示正在进行转换为 NSEC3 链的过程。其余的记录包含一个 NSEC3PARAM 记录。标志字段基于标志位表明要执行哪种操作。

0x01 OPTOUT

0x80 CREATE

0x40 REMOVE

0x20 NONSEC

5.9.5 DNSKEY 轮转

随着不安全到安全的转换，轮转 DNSSEC 密钥可以使用两种方法完成：使用一个动态 DNS 更新，或者 auto-dnssec 区选项。

5.9.6 动态 DNS 更新方法

为通过动态更新执行密钥轮转，需要为新密钥添加 K* 文件，这样 named 就能够找到它们。然后可以通过动态更新添加新的 DNSKEY 资源记录集。然后将导致 named 使用新的密钥对区进行签名。当签名完成，将更新私有类型记录，使最后一个字节为非零。

如果这是一个 KSK，需要将新 KSK 通知上级域和所有的信任锚仓库。

在删除旧 DNSKEY 之前，区中最大 TTL 必须过期。如果正在更新一个 KSK，上级区中的 DS 资源记录集也必须更新，并允许其 TTL 过期。这就确保在删除旧 DNSKEY 时，所有的客户端能够验证至少一个签名。

可以通过 UPDATE 删除旧的 DNSKEY。需要小心指定正确的密钥。在更新完成后，named 将会清理由旧密钥生成的所有签名。

5.9.7 自动密钥轮转

当一个新密钥达到其激活日期（由 dnssec-keygen 或 dnssec-settime 所设置的）时，如果 auto-dnssec 区选项被设置为 maintain，named 将会自动执行密钥轮转。如果密钥的算法之前没有用于签名区，区将被尽可能快地被全部签名。但是，如果替代现有密钥的新密钥使用同样的算法，则区将被增量重签，在其签名有效期过期后，旧密钥的签名被新密钥的签名所替代。缺省时，这个轮转在 30 天内完成，之后就可以安全地将旧密钥从 DNSKEY 资源记录集中删掉。

5.9.8 通过 UPDATE 轮转 NSEC3PARAM

可以通过动态更新增加新的 NSEC3PARAM 记录。当生成了新的 NSEC3 链之后，NSEC3PARAM 标志字段被置为零。在这时，可以删除旧的 NSEC3PARAM 记录。旧的链将会在更新请求完成之后被删除。

5.9.9 从 NSEC 转换到 NSEC3

要做这个，必须添加一条 NSEC3PARAM 记录。在转换完成后，NSEC 链将被移除，同时 NSEC3PARAM 记录有一个 0 标志域。NSEC3 链将在 NSEC 链被去掉之前生成。

dnssec-policy 还不支持 NSEC3。

5.9.10 从 NSEC3 转换到 NSEC

要做这个，使用 nsupdate 删除所有带有一个零标志字段的 NSEC3PARAM 记录。在 NSEC3 链被删除之前先生成 NSEC 链。

5.9.11 从安全转换为不安全

要使用动态 DNS 将一个签名的区转换为未签名的区，需要使用 nsupdate 删除区顶点的所有 DNSKEY 记录。所有签名，NSEC 或 NSEC3 链，以及相关的 NSEC3PARAM 记录都会被自动地删除掉。这个发生在更新请求完成之后。

这要求 named.conf 中的 dnssec-secure-to-insecure 选项被设置为 yes 。

此外，如果使用了 auto-dnssec maintain 区命令，应该将其去掉或者将其值改为 allow ；否则它将被重签。

5.9.12 定期重签名

在任何支持动态更新的安全区中，named 会定期对因为某些更新动作而变为未签名的资源记录集进行重新签名。签名的生存期会被调整，这样就会将重新签名的负载分散在一段时间而不是集中在一起。

5.9.13 NSEC3 和 OPTOUT

named 仅仅支持一个区的所有 NSEC3 记录都有同样的 OPTOUT 状态才建立新的 NSEC3 链。named 支持更新那些在链中的 NSEC3 记录有混合 OPTOUT 状态的区。named 不支持变更一个单独 NSEC3 记录的 OPTOUT 状态，如果需要变更一个单独 NSEC3 记录的 OPTOUT 状态，就需要变更整个链。

第 5.10 节 动态信任锚管理

BIND 能够使用 RFC 5011 密钥管理维护 DNSSEC 信任锚。这个特性允许 named 跟踪关键的 DNSSEC 密钥变化，而不需要操作员改变配置文件。

5.10.1 验证解析器

为了配置一个验证解析器，使其用 RFC 5011 来维护一个信任锚点，需要使用一个 trust-anchors 语句及 initial-key 关键字来配置信任锚。关于这个的信息可以在dnssec-keys 语句定义和用法 中找到。

5.10.2 权威服务器

为设置一个使用 RFC 5011 信任锚点维护的权威区，需要为区生成两个（或更多）的密钥签名密钥（KSK）。使用其中一个对区签名；这个密钥就是“活跃的”KSK。所有不对区签名的 KSK 都是“备用”密钥。

任何配置成将活跃 KSK 用作一个 RFC 5011 所管理的信任锚点的验证解析器都会留意到区的 DNSKEY 资源记录集中的后备密钥，并将其存储以备将来参考。解析器将会定期重新检查区，并在 30 天之后，如果新密钥仍然存在，这个密钥将会作为这个区的一个有效信任锚点被解析器所接受。在这个 30 天的接受计时器结束后的任何时间，活跃的 KSK 可以被撤销，而区可以被“轮转”到新的所接受的密钥。

将一个备用密钥放入一个区的最简单的方法是使用 dnssec-keygen 和 dnssec-signzone 的“智能签名”特征。如果一个密钥的发布日期在过去，而激活日期未设置或是在未来，“dnssec-signzone -S ”将会把 DNSKEY 记录包含到区中，但是不会使用它签名：

```
$ dnssec-keygen -K keys -f KSK -P now -A now+2y example.net
$ dnssec-signzone -S -K keys example.net
```

为撤销一个密钥，增加了新的命令 dnssec-revoke 。这个命令增加密钥标志的 REVOKED 位，并重新生成 K*.key 和 K*.private 文件。

在撤销一个活跃的密钥之后，必须使用被撤销的 KSK 和新的活跃的 KSK 对区签名。智能签名自动完成这个工作。

一旦一个密钥被撤销并用于对其所在的 DNSKEY 资源记录集签名，这个密钥就再也不能被解析器接受为一个有效的信任锚点。然而，可以使用新的活跃密钥（这个新密钥在作为备用密钥时已经被解析器所接受）进行验证。

关于密钥轮换场景的更详细信息，参见 RFC 5011 。

当一个密钥被撤销时，其密钥 ID 会变化，即增加 128，并在超过 65535 时轮转。所以，例如，密钥“Kexample.com.+005+10000 ”变成了“Kexample.com.+005+10128 ”。

如果两个密钥的 ID 刚好相差 128，并且在其中一个被撤销时，两个密钥 ID 发生了碰撞，就会带来一些问题。为避免这种情况，如果出现的另一个密钥可能发生碰撞，dnssec-keygen 将不会生成

一个新的密钥。这个检查仅仅发生在当新密钥被写到存放区的所有其它在用密钥的同一个目录的时候。

旧版本的 BIND 9 没有这个预防措施。如果在用密钥撤销碰到先前版本所生成的密钥，或者在用密钥存储在多个目录或多个服务器上，将会发出警告。

预料将来发布的 BIND 9 将会以一个不同的方式应对这个问题，即通过将已撤销的密钥和其原始未撤销密钥的 ID 一起存储。

第 5.11 节　PKCS#11(Cryptoki) 支持

PKCS#11 (公钥加密标准 #11) 为控制硬件安全模块（HSM）和其它密码支持设备定义了一个平台独立的 API。

BIND 9 可以支持三种 HSM：AEP Keyper，在 Debian Linux，Solaris x86 和 Windows Server 2003 上测试通过；Thales nShield，在 Debian Linux 上测试通过；以及 Sun SCA 6000 密码加速板，在 Solaris x86 上测试通过。另外，BIND 可以与所有当前版本的 SoftHSM，一种基于软件的由 OpenDNSSEC 项目产出的 HSM 模拟器，一块工作。

PKCS#11 利用了一个“提供者库（provider library）”：一个动态加载库，它提供一个低级 PKCS#11 接口来驱动 HSM 硬件。PKCS#11 提供者库来自 HSM 厂商，它针对于由其控制的特定 HSM 有效。

BIND 9 中存在两种可用的对 PKCS#11 支持的机制：基于 OpenSSL 的 PKCS#11 和原生 PKCS#11。在使用第一种机制时，BIND 使用一个 OpenSSL 的修改版本，后者加载提供者库并间接操作 HSM；任何 HSM 不支持的加密操作都替代成由 OpenSSL 执行。第二种机制让 BIND 完全绕过 OpenSSL；BIND 自己加载提供者库，并使用 PKCS#11 API 直接驱动 HSM。

5.11.1　先决条件

关于 HSM 的安装，初始化，测试和故障解决方面的信息，参见由 HSM 厂商提供的文档。

5.11.2　原生 PKCS#11

原生 PKCS#11 模式仅能与一个能够执行 **每个** BIND 9 可能需要的加密操作的 HSM 一块工作。HSM 的提供者库必须带有一个 PKCS#11 API 的完全实现，因此所有这些函数都是可以访问的。在写作本文档时，仅有 Thales nShield HSM 和 SoftHSMv2 可以被用于这个功能。对其它 HSM，包括 AEP Keyper，Sun SCA 6000 和旧版本的 SoftHSM，使用基于 OpenSSL 的 PKCS#11。（注意：最终，当更多 HSM 支持原生 PKCS#11，可以预料基于 OpenSSL 的 PKCS#11 将被废弃。）

要在构建 BIND 带有对原生 PKCS#11 的支持，按如下配置：

```
$ cd bind9
$ ./configure --enable-native-pkcs11 \
  --with-pkcs11=provider-library-path
```

这将导致所有 BIND 工具，包括 named 和 dnssec-* 及 pkcs11-* 工具，使用在 provider-library-path 中指定的 PKCS#11 提供者库来加密。（提供者库的路径可以在 named 和 dnssec-* 工具中使用 -E ，或者在 pkcs11-* 工具中使用 -m 来覆盖。）

构建 SoftHSMv2

SoftHSMv2，SoftHSM 的最新开发版，可在 https://github.com/opendnssec/SoftHSMv2 得到。它是由 OpenDNSSEC 项目所开发的软件库（https://www.opendnssec.org），它提供了一个虚拟 HSM 的 PKCS#11 接口，以一个在本地文件系统上的 SQLite3 数据库的形式实现。它提供的安全性较一个真正的 HSM 较少，但是它允许你在没有可用的 HSM 时试验原生的 PKCS#11。SoftHSMv2 可以被配置成使用 OpenSSL 或者 Botan 库来执行加密功能，但当其用于在 BIND 中的原生 PKCS#11 时，必须使用 OpenSSL。

缺省时，SoftHSMv2 配置文件是 prefix/etc/softhsm2.conf（这里的 prefix 是在编译时配置的）。这个位置可以被 SOFTHSM2_CONF 环境变量覆盖。SoftHSMv2 加密存储必须在与 BIND 一起使用之前被安装和初始化。

```
$ cd SoftHSMv2
$ configure --with-crypto-backend=openssl --prefix=/opt/pkcs11/usr
$ make
$ make install
$ /opt/pkcs11/usr/bin/softhsm-util --init-token 0 --slot 0 --label softhsmv2
```

5.11.3 基于 OpenSSL 的 PKCS#11

基于 OpenSSL 的 PKCS#11 模式使用了一个 OpenSSL 库的修改版本；当前官方版本的 OpenSSL 不能完全支持 PKCS#11。ISC 提供了一个补丁来纠正这个情况。这个补丁是基于源于 OpenSolaris 项目所完成的工作；它被 ISC 修改以提供诸如 PIN 管理和密钥引用（译注：key-by-reference）等新特性。

打过补丁的 OpenSSL 提供了两种 PKCS#11 新“风味”，其中一种必须在配置时选择。正确的选择依赖于 HSM 硬件：

- ‘crypto-accelerator’用于具有硬件加密加速的 HSM，如 SCA 6000 板。这使得 OpenSSL 在 HSM 中运行所有支持的加密操作。
- ‘sign-only’用于那些设计功能主要是作为密钥存储设备，而缺乏硬件加速的 HSM。这类设备是高安全的，但是不需要在加密时比系统 CPU 更快——通常，他们会更慢。因此，最高效的方法是仅将其用于要求访问安全私钥的加密功能，而其它所有计算密集性操作都使用系统 CPU。AEP Keyper 是这类设备的一个例子。

修改的 OpenSSL 代码包含在 BIND 9 发行版中，以针对最新 OpenSSL 版本的带上下文的 diff 结果的形式。OpenSSL 0.9.8，1.0.0，1.0.1 和 1.0.2 都被支持；分别有各自版本的 diff 文件。在后面的例子中，我们使用了 OpenSSL 0.9.8，但是同样的方法也适用于 OpenSSL 1.0.0 到 1.0.2。

注解：这篇文档（2016 年 1 月）的最新 OpenSSL 版本是 0.9.8zh，1.0.0t，1.0.1q 和 1.0.2f。ISC 在新版本的 OpenSSL 发布时会提供升级的补丁。在以下例子中的版本号预期也会改变。

在构建带 PKCS#11 支持的 BIND 9 之前，需要在适当的位置使用这个补丁构建 OpenSSL 并使用你的 HSM 的 PKCS#11 提供者库的路径来配置它。

给 OpenSSL 打补丁

```
$ wget http://www.openssl.org/source/openssl-0.9.8zc.tar.gz
```

解压 tar 包：

```
$ tar zxf openssl-0.9.8zc.tar.gz
```

应用 BIND 9 发行版所带的补丁：

```
$ patch -p1 -d openssl-0.9.8zc \
    < bind9/bin/pkcs11/openssl-0.9.8zc-patch
```

注解：补丁文件可能与不同操作系统下的”patch”应用不兼容。你可能需要安装 GNU patch。

在构建 OpenSSL 时，将其放在一个非标准的位置，这样它不会干扰系统上的 OpenSSL 库。在下面的例子中，我们选择安装到”/opt/pkcs11/usr”。我们将在配置 BIND 9 时使用这个位置。

之后，在构建 BIND 9 时，定制构建的 OpenSSL 库的位置需要通过 configure 指定。

在 Linux 上为 AEP Keyper 构建 OpenSSL

AEP Keyper 是一个高安全密钥存储设备，但是不提供硬件加密加速。它能够执行加密操作，但是可能会减慢你的系统 CPU。因此，我们在构建 OpenSSL 时选择’sign-only’特性。

Keyper 专用的 PKCS#11 提供者库是随同 Keyper 软件分发的。在这个例子中，我们将其放在/opt/pkcs11/usr/lib 下：

```
$ cp pkcs11.GCC4.0.2.so.4.05 /opt/pkcs11/usr/lib/libpkcs11.so
```

```
$ cd openssl-0.9.8zc
$ ./Configure linux-x86_64 \
    --pk11-libname=/opt/pkcs11/usr/lib/libpkcs11.so \
    --pk11-flavor=sign-only \
    --prefix=/opt/pkcs11/usr
```

为 Solaris 上的 SCA 6000 构建 OpenSSL

SCA-6000 PKCS#11 提供者是作为一个系统库 libpkcs11 安装的。它是一个真正的加密加速器，能够比任何 CPU 快 4 倍以上，所以特性应该是‘crypto-accelerator’。

在这个例子中，我们正在 AMD64 系统上的 Solaris x86 平台上构建。

```
$ cd openssl-0.9.8zc
$ ./Configure solaris64-x86_64-cc \
    --pk11-libname=/usr/lib/64/libpkcs11.so \
    --pk11-flavor=crypto-accelerator \
    --prefix=/opt/pkcs11/usr
```

（对一个 32 位构建，使用”solaris-x86-cc”和/usr/lib/libpkcs11.so。）

在配置之后，运行 make 和 make test 。

为 SoftHSM 构建 OpenSSL

SoftHSM（版本 1）是一个由 OpenDNSSEC 项目 (http://www.opendnssec.org) 所提供的软件库，它提供了一个虚拟 HSM 的 PKCS#11 接口，以一个在本地文件系统上的 SQLite3 数据库的形式实现。SoftHSM 使用 Botan 库执行加密功能。虽然比一个真正的 HSM 更不安全，但是它允许你在没有可用的 HSM 时试验 PKCS#11。

在与 OpenSSL 一起使用 SoftHSM 之前，必须安装和初始化 SoftHSM 加密存储，并且 SOFTHSM_CONF 环境变量必须总是指向 SoftHSM 配置文件：

```
$ cd softhsm-1.3.7
$ configure --prefix=/opt/pkcs11/usr
$ make
$ make install
$ export SOFTHSM_CONF=/opt/pkcs11/softhsm.conf
$ echo "0:/opt/pkcs11/softhsm.db" > $SOFTHSM_CONF
$ /opt/pkcs11/usr/bin/softhsm --init-token 0 --slot 0 --label softhsm
```

SoftHSM 可以执行所有的加密操作，但是由于它只使用你系统的 CPU，在除了签名之外的其它事务上使用都没有优势。因此，我们在构建 OpenSSL 时选择’sign-only’特性。

```
$ cd openssl-0.9.8zc
$ ./Configure linux-x86_64 \
    --pk11-libname=/opt/pkcs11/usr/lib/libsofthsm.so \
    --pk11-flavor=sign-only \
    --prefix=/opt/pkcs11/usr
```

在配置之后，运行”make”和”make test”。

一旦你完成构建 OpenSSL，运行”apps/openssl engine pkcs11”来确认 PKCS#11 支持是正确编译的。输出应该是下列行中的一种，具体取决于所选的特性：

```
(pkcs11) PKCS #11 engine support (sign only)
```

或：

```
(pkcs11) PKCS #11 engine support (crypto accelerator)
```

接下来，运行”apps/openssl engine pkcs11 -t”。这将试图初始化 PKCS#11 引擎。如果能够顺利完成，它将会报告“[available]”。

如果输出正确，运行”make install”，将会把修改后的 OpenSSL 套件安装到 /opt/pkcs11/usr 。

为 Linux 配置带 AEP Keyper 的 BIND 9

```
$ cd ../bind9
$ ./configure \
```

（下页继续）

(续上页)

```
    --with-openssl=/opt/pkcs11/usr \
    --with-pkcs11=/opt/pkcs11/usr/lib/libpkcs11.so
```

为 Solaris **配置带** SCA 6000 **的** BIND 9

```
$ cd ../bind9
$ ./configure CC="cc -xarch=amd64" \
     --with-openssl=/opt/pkcs11/usr \
     --with-pkcs11=/usr/lib/64/libpkcs11.so
```

（对一个 32 位的构建，省略 CC=”cc -xarch=amd64”。）

如果 configure 报告 OpenSSL 不工作，你可能有一个 32/64 位体系结构的不匹配。或者，你可能没有为 OpenSSL 指定正确的路径（这个路径应该与 OpenSSL 配置时的–prefix 参数一样）。

为 SoftHSM **配置** BIND 9

```
$ cd ../bind9
$ ./configure \
    --with-openssl=/opt/pkcs11/usr \
    --with-pkcs11=/opt/pkcs11/usr/lib/libsofthsm.so
```

在配置后，运行”make”,”make test”和”make install”。

（注意：如果“make test”在“pkcs11”系统测试中失败，你可能是忘记设置 SOFTHSM_CONF 环境变量了。）

5.11.4 PKCS#11 工具

BIND 9 包含一个用以操作 HSM 的工具的最小集合，包括 pkcs11-keygen，用于在 HSM 内生成一个新的密钥对，pkcs11-list，用于列出当前可用的对象，pkcs11-destroy，用于删除对象，和 pkcs11-tokens，用于列出可用的符号。

在 UNIX/Linux 构建中，这些工具仅在 BIND 9 使用–with-pkcs11 选项配置时才被构建。（注意：如果–with-pkcs11 被设置为“yes”，而不是 PKCS#11 提供者的路径，这时这些工具会被构建，但是提供者将会保持未定义的状态。使用-m 选项或 PKCS11_PROVIDER 环境变量来指定提供者的路径。）

5.11.5 使用 HSM

对基于 OpenSSL 的 PKCS#11，我们必须设置运行时环境，以便装载 OpenSSL 和 PKCS#11 库：

```
$ export LD_LIBRARY_PATH=/opt/pkcs11/usr/lib:${LD_LIBRARY_PATH}
```

这导致 named 和其它的二进制可执行程序从 /opt/pkcs11/usr/lib 而不是缺省位置装载 OpenSSL 库。在使用原生 PKCS#11 时不需要本步骤。

一些 HSM 要求设置其它的环境变量。例如，在操作一个 AEP Keyper 时，也需要指定“machine”文件的位置，它存放提供者库所用到的 Keyper 的信息。如果机器文件在 /opt/Keyper/PKCS11Provider/machine , 使用：

```
$ export KEYPER_LIBRARY_PATH=/opt/Keyper/PKCS11Provider
```

无论何时运行使用 HSM 的任何工具，都必须设置这些环境变量，包含 pkcs11-keygen, pkcs11-list, pkcs11-destroy, dnssec-keyfromlabel, dnssec-signzone, dnssec-keygen 和 named 。

现在我们可以在 HSM 中创建和使用密钥。在这个例子中，我们将创建一个 2048 位的密钥并赋予其一个标记” sample-ksk”：

```
$ pkcs11-keygen -b 2048 -l sample-ksk
```

要确认密钥已经存在：

```
$ pkcs11-list
Enter PIN:
object[0]: handle 2147483658 class 3 label[8] 'sample-ksk' id[0]
object[1]: handle 2147483657 class 2 label[8] 'sample-ksk' id[0]
```

在使用这个密钥签名一个区之前，我们必须创建一对 BIND 9 密钥文件。“dnssec-keyfromlabel”应用程序完成这件事。在这个例子中，我们将使用 HSM 密钥” sample-ksk” 作为” example.net” 的密钥签名密钥：

```
$ dnssec-keyfromlabel -l sample-ksk -f KSK example.net
```

作为结果的 K*.key 和 K*.private 文件现在可以用于签名区。与包含公钥和私钥的普通 K* 文件不同，这些文件只包含公钥数据，和一个存储在 HSM 中的私钥的标识符。使用私钥进行签名是在 HSM 内部完成的。

如果你想要在 HSM 中生成第二个密钥用作一个区签名密钥，遵循上面同样的流程，使用一个不同的密钥标记，一个更小的密钥长度，并在 dnssec-keyfromlabel 的参数中去掉” -f KSK”：

（注意：当使用基于 OpenSSL 的 PKCS#11 时，标记是一个任意的字符串，它标识密钥。使用原生 PKCS#11 时，标记是一个 PKCS#11 URI 字符串，其中可能包含关于和 HSM 的更详细的信息，包括自身的 PIN。更详细的内容参见dnssec-keyfromlabel - DNSSEC 密钥生成工具 。）

```
$ pkcs11-keygen -b 1024 -l sample-zsk
$ dnssec-keyfromlabel -l sample-zsk example.net
```

作为选择，你也可能更喜欢使用 dnssec-keygen 来生成一个传统的存放在硬盘上的密钥：

```
$ dnssec-keygen example.net
```

这比一个 HSM 密钥提供更弱的安全性，但是由于安全的原因，HSM 可能更慢或者使用不方便，保留 HSM 并将其用于更小频率的密钥签名操作可能更为有效。如果你愿意，区签名密钥可以轮转更为频繁，以补偿密钥安全性的降低。（注意：在使用原生 PKCS#11 时，与使用硬盘上的密钥相比没有速度优势，因为加密操作无论如何将由 HSM 完成。）

现在你可以对区签名了。（注意：如果不给 dnssec-signzone 使用-S 选项，就需要将两个 K*.key 文件的内容添加到区的主文件中再签名。）

```
$ dnssec-signzone -S example.net
Enter PIN:
Verifying the zone using the following algorithms:
NSEC3RSASHA1.
Zone signing complete:
Algorithm: NSEC3RSASHA1: ZSKs: 1, KSKs: 1 active, 0 revoked, 0 stand-by
example.net.signed
```

5.11.6 在命令行指定引擎

在使用基于 OpenSSL 的 PKCS#11 时，OpenSSL 所使用的“engine”可以通过使用“-E <engine>”命令行选项在 named 和所有 BIND 的 dnssec-* 工具中指定。如果 BIND 9 是使用--with-pkcs11 选项构建的，这个选项缺省为“pkcs11”。通常是不需要指定引擎的，除非因为某种原因，你希望使用一个不同的 OpenSSL 引擎。

如果你希望关闭使用“pkcs11”引擎——因为调试目的，或者因为 HSM 不可用——就将引擎设置为空串。例如：

```
$ dnssec-signzone -E '' -S example.net
```

这将导致 dnssec-signzone 运行在如同没有使用--with-pkcs11 选项编译时的情况。

当使用原生 PKCS#11 模式构建时，“engine” 选项具有一个不同的含义：它指定 PKCS#11 提供者库的路径。这在测试一个新的提供者库时可能很有用。

5.11.7 以自动区重签的方式运行 named

如果你想要 named 使用 HSM 密钥动态重签区，和/或签名通过 nsupdate 插入的新记录，named 必须能够访问 HSM 的 PIN。在基于 OpenSSL 的 PKCS#11 中，这是通过将 PIN 放在 openssl.cnf 文件中来达到（在上面的例子中，/opt/pkcs11/usr/ssl/openssl.cnf ）。

openssl.cnf 文件的位置可以在运行 named 之前通过设置 OPENSSL_CONF 环境变量进行覆盖。

openssl.cnf 例子：

```
openssl_conf = openssl_def
[ openssl_def ]
engines = engine_section
[ engine_section ]
pkcs11 = pkcs11_section
[ pkcs11_section ]
PIN = <PLACE PIN HERE>
```

这也将允许 dnssec-* 工具无需 PIN 入口码就能够访问 HSM。（pkcs11-* 工具直接访问 HSM，不经过 OpenSSL，所以仍然需要一个 PIN 来使用它们。）

在原生 PKCS#11 模式，PIN 可以在一个作为密钥标记的一个属性所指定的文件中提供。例如，如果一个密钥有一个标记 pkcs11:object=local-zsk;pin-source=/etc/hsmpin ，就可以从文件 /etc/hsmpin 中读到 PIN。

> **警告**：在这个方式中，将 HSM 的 PIN 放在一个文本文件中可能减少使用一个 HSM 的安全优势。在以这种方式配置系统之前，确认这就是你想要的方式。

第 5.12 节 DLZ (Dynamically Loadable Zones，动态加载区)

动态加载区（DLZ）是一项 BIND 9 扩展，它允许直接从一个外部数据库中提取区数据。它对格式或模式没有要求。已经存在对应几种不同的数据库后端包括 PostgreSQL，MySQL 和 LDAP 的 DLZ 驱动，也可以写其它驱动。

早期，DLZ 驱动是静态链接到 named 二进制代码的，并通过编译时的配置选项打开（例如，configure --with-dlz-ldap ）。现在，驱动在 BIND 9 源码包中提供，在 contrib/dlz/drivers 中，仍然以这种方式链接。

在 BIND 9.8 或更高版本，可以通过 DLZ “dlopen” 驱动，它充当了一个通用中间层封装了一个实现 DLZ API 的共享对象，在运行时动态链接某些 DLZ 模块。“dlopen” 驱动缺省链接到 named，在使用这些动态可链接的驱动时就不再需要配置选项了，但是使用 contrib/dlz/drivers 下面的旧驱动时仍然需要。

当 DLZ 模块以文本格式给 named 提供数据，由 named 转换为 DNS 线上格式。这个转换缺乏任何内部缓存，给 DLZ 模块的查询性能带来了重大限制。因此，不推荐在大访问量服务器上使用 DLZ。然而，它可以被用于一个隐藏主配置中，让辅服务器通过 AXFR 获取区更新。注意，由于 DLZ 没有对 DNS notify 的内置支持；辅服务器不会收到数据库中区变化的通知信息。

5.12.1 配置 DLZ

通过 named.conf 中一个 dlz 语句配置一个 DLZ 数据库：

```
dlz example {
database "dlopen driver.so args";
search yes;
};
```

这指定了一个在回复请求时要搜索的 DLZ 模块；这个模块在 driver.so 中实现，并通过 dlopen DLZ 驱动在运行时装载。可以指定多个 dlz 语句；在回复一个请求时，所有 search 被设置为 yes 的 DLZ 模块都将被查询，以发现它们是否包含对请求名的答复；最好的可用答复将被返回给客户端。

上述例子中的 search 选项可以省略，因为 yes 是缺省值。

如果 search 被设置为 no ，在收到一个请求时，就 **不会**在这个 DLZ 模块中查找最佳匹配。作为替代，在这个 DLZ 中的区必须在一个 zone 语句中独立指定。这允许你使用标准的区选项语义配置一个区，同时却指定一个不同的数据库后端来存储区数据。例如，使用一个 DLZ 模块作重定向规则的后端存储来实现 NXDOMAIN 重定向：

```
dlz other {
database "dlopen driver.so args";
search no;
};
```

（下页继续）

(续上页)

```
zone "." {
type redirect;
dlz other;
};
```

5.12.2 样例 DLZ 驱动

为指导实现 DLZ 模块，目录 contrib/dlz/example 包含了一个基本的动态可链接 DLZ 模块 - 即一个可以由“dlopen” DLZ 驱动在运行时加载的模块。这个例子建立了一个区，其名字作为 dlz 语句中的一个参数被传递给模块：

```
dlz other {
database "dlopen driver.so example.nil";
};
```

在上面的例子中，模块被配置为建立一个区“example.nil”，它可以回复查询和 AXFR 请求，并接受 DDNS 更新。在运行时，在任何更新的前面，区中包含其顶点一条 SOA 记录，一条 NS 记录及一条 A 记录：

```
example.nil. 3600   IN    SOA    example.nil. hostmaster.example.nil. (
                123 900 600 86400 3600
            )
example.nil. 3600   IN    NS     example.nil.
example.nil. 1800   IN    A      10.53.0.1
```

样例驱动能够提取关于请求客户端的信息，并基于这个信息修改它的响应。为演示这个特性，例子驱动用请求的源地址响应对“source-addr.“zonename“>/TXT” 的请求。注意，这个记录将 **不会**被包含到 AXFR 或 ANY 响应中。通常，这个特性用于以一些其它的方式修改响应。例如，根据请求来自的不同网络而对同一个特定名字提供不同的地址记录。

DLZ 模块 API 的文档可以在 contrib/dlz/example/README 中找到。这个目录也包含头文件 dlz_minimal.h ，后者定义了这个 API 并应被包含在任何动态可链接 DLZ 模块中。

第 5.13 节 DynDB（动态数据库）

动态数据库是一个对 BIND 9 的扩展，如同 DLZ（参见DLZ (Dynamically Loadable Zones，动态加载区)）一样，它允许从一个外部数据库中提取区数据。与 DLZ 不一样的是，一个 DynDB 模

块提供了一个全功能的 BIND 区数据库接口。DLZ 将 DNS 请求翻译成实时数据库查找，导致相对低下的请求性能，并且由于其有限的 API 而无法处理 DNSSEC 签名数据，而一个 DynDB 模块可以从外部数据源中预装载到一个内存数据库，可以提供由原生 BIND 服务的区一样的性能和功能。

红帽建立了一个支持 LDAP 的 DynDB 模块，可以在 https://pagure.io/bind-dyndb-ldap 找到。

一个用于测试和开发者指导的样例 DynDB 模块被包含在 BIND 源代码中，在目录 bin/tests/system/dyndb/driver 中。

5.13.1 配置 DynDB

一个 DynDB 数据库是在 named.conf 中使用一个 “dyndb“ 语句来配置：

```
dyndb example "driver.so" {
  parameters
};
```

文件 driver.so 是一个 DynDB 模块，它实现了全部 DNS 数据库 API。可以指定多条 dyndb 语句，以装载不同的驱动或者同一个驱动的多个实例。由一个 DynDB 模块提供的区被添加到视图的区表中，并且在 BIND 响应请求时被当成一个普通的权威区。区配置由 DynDB 模块内部进行处理。

parameters 被作为一个不透明的字符串传递给 DynDB 模块的初始化程序。配置语法依赖于驱动的不同而有所差别。

5.13.2 样例 DynDB 模块

为指导 DynDB 模块的实现，目录 bin/tests/system/dyndb/driver 含有一个基本的 DynDB 模块。这个例子建立两个区，其名字被 dyndb 语句中的参数传送给模块，

```
dyndb sample "sample.so" { example.nil. arpa. };
```

在上述的例子中，模块被配置以建立一个区 “example.nil”，它可以响应查询请求和 AXFR 请求，并接受 DDNS 更新。运行时，在任何更新之前，区的顶点包含一个 SOA、NS 和一个 A 记录。

```
example.nil. 86400   IN    SOA    example.nil. example.nil. (
                                0 28800 7200 604800 86400
                            )
example.nil. 86400   IN    NS     example.nil.
example.nil. 86400   IN    A      127.0.0.1
```

当区被动态更新时，DynDB 模块将决定被更新的资源记录是否是一个地址（如，类型 A 或者 AAAA），如果是，它将自动更新一个反向区中对应的 PTR 记录。注意更新不会永久保存；所有的更新在服务器重启时会丢失。

第 5.14 节　目录区

一个“目录区”是一个特定的 DNS 区，它包含一个用于服务的区的列表，以及这些区的配置参数。在目录区中列出的区被称为“member zones”（成员区）。当一个目录区被装载或传送给一个支持这个功能的辅服务器时，辅服务器将会自动建立成员区。当目录区被更新后（例如，增加或删除成员区，或修改成员区的配置参数），这些变化会立即生效。因为目录区是一个普通的 DNS 区，这些配置变化可以使用标准的 AXFR/IXFR 区传送机制扩散。

为了多个 DNS 实现的互操作性，目录区的格式和行为被规范到一个互联网草案。最新版的 DNS 目录区草案可以在这里找到：https://datatracker.ietf.org/doc/draft-toorop-dnsop-dns-catalog-zones/.

5.14.1　操作原理

普通地，如果一个区由一个辅服务器服务提供服务，服务器上的 named.conf 文件必须列出区，或者区必须使用 rndc addzone 添加。在一个大量辅服务器和/或其中所服务的区被频繁修改，在所有辅服务器上维护区配置的一致性的成本就会很大。

一个目录区是一种减小这个管理负担的方法。它是一个列出应在辅服务器上提供服务的成员区的 DNS 区。当一个辅服务器收到一个对目录区的更新，它基于所收到的数据增加、删除或者重新配置成员区。

要使用一个目录区，首先必须在主服务器上作为一个普通区设置好，并在辅服务器上配置成使用它。它必须被添加到 named.conf 中 options 或 view 语句的一个 catalog-zones 列表中。与之相对的方式是，一个策略区作为一个普通区配置并且也在一个 response-policy 语句中列出。

使用目录区特性提供一个新的成员区：

- 和通常一样在主服务器上建立用于服务的成员区。这可以通过编辑 named.conf 或运行 rndc addzone 来完成。
- 在目录区中为新的成员区增加一个条目。这可以通过编辑目录区的主服务器文件并运行 rndc reload ，或者使用 nsupdate 更新这个区来完成。

对目录区的修改将使用普通的 AXFR/IXFR 机制从主服务器扩散到所有的辅服务器。当辅服务器收到对目录区的更新，它会检查新成员区条目，在辅服务器上创建区的实例，并将实例指向目录区数

据所指定的 masters 。新创建的成员区是一个普通的辅区，所以 BIND 会立即发起一个从主区的区内容的传送。一旦完成，辅服务器将开始服务于这个成员区。

从一个辅服务器删除一个成员区只是在目录区中删除成员区条目，不会有更多的要求。对目录区的修改使用普通的 AXFR/IXFR 传送机制扩散到辅服务器。辅服务器在处理更新时中会注意到成员区已被删除。它停止这个区的服务并将其从已配置区的清单中删去。然而，从主服务器删除成员区必须通过编辑配置文件或运行 rndc delzone 完成。

5.14.2 配置目录区

目录区是在 named.conf 中的 options 或 view 部份中使用一条 catalog-zones 语句来配置的。例如：

```
catalog-zones {
  zone "catalog.example"
     default-masters { 10.53.0.1; }
     in-memory no
     zone-directory "catzones"
     min-update-interval 10;
};
```

这个语句指定区 catalog.example 为一个目录区。这个区必须正确地配置在同一个视图中。在大多数配置中，它是一个辅区。

区名后面的选项不是必须的，可以以任何顺序指定：

default-masters 该选项定义一个目录区中成员区的缺省主服务器。这可以被一个目录区内的选项所覆盖。如果未包含这样的选项，成员区将从这个选项中所列的服务器传输它们的内容。

in-memory 如果该选项设置为 yes ，将使成员区仅存放于内存。这在功能上等效于配置一个辅区而不使用一个 file 选项。缺省是 no ；成员区的内容将会保存在一个本地的文件中，其名字由视图名、目录区名和成员区名自动生成。

zone-directory 如果 in-memory 未被设置为 yes ，该选项使得成员区主文件的本地拷贝被存放在一个指定的目录中。缺省是将区文件存放在服务器的工作目录。在 zone-directory 中一个非绝对路径被假设为相对于工作目录。

min-update-interval 该选项设置处理目录区更新的最小间隔，以秒计。如果一个目录区的更新（例如，通过 IXFR）发生于最近的更新后不到 min-update-interval 秒，则变化不会被执行，直到这个间隔时间过去之后。缺省是 5 秒。

目录区的定义基于每个视图。在一个视图中配置一个非空的 catalog-zones 语句将会自动对这个视图打开 allow-new-zones 。这意谓着在支持目录区的任何视图上，rndc addzone 和 rndc delzone 也可以工作。

5.14.3 目录区格式

目录区是一个普通的 DNS 区；所以，它必须拥有一个 SOA 和至少一个 NS 记录。

一个声明目录区格式的版本的记录也是必须的。如果所列的版本号是服务器不支持的，目录区不能被用于那台服务器。

```
catalog.example.    IN SOA . . 2016022901 900 600 86400 1
catalog.example.    IN NS nsexample.
version.catalog.example.    IN TXT "1"
```

注意这个记录必须有域名 version.catalog-zone-name。存储在一个目录区的数据的含义是由紧接在目录区域名之前的域名标记来指明的。

目录区选项可以全局为所有目录区设置，也可为一个单独的成员区设置。全局选项覆盖配置文件中的设置，而成员区选项覆盖全局选项。

全局选项设置在目录区的顶点，例如：

```
masters.catalog.example.    IN AAAA 2001:db8::1
```

BIND 当前支持下列选项：

- 一个简单的 masters 定义：

  ```
  masters.catalog.example.    IN A 192.0.2.1
  ```

  这个选项为成员区定义一个主服务器 - 它可以是一条 A 或者 AAAA 记录。如果设置了多个主服务器，其使用顺序是随机的。

- 一个带有 TSIG 密钥定义的 masters ：

  ```
  label.masters.catalog.example.    IN A 192.0.2.2
  label.masters.catalog.example.    IN TXT "tsig_key_name"
  ```

  这个选项使用一个 TSIG 密钥设置为成员区定义一个主服务器。TSIG 密钥必须配置在配置文件中。label 可以是任何有效的 DNS 标记。

- allow-query 和 allow-transfer ACLs:

```
allow-query.catalog.example.   IN APL 1:10.0.0.1/24
allow-transfer.catalog.example.    IN APL !1:10.0.0.1/32 1:10.0.0.0/24
```

这些选项等效于在 named.conf 配置文件中一个区定义中的 allow-query 和 allow-transfer 。ACL 被顺序处理；如果没有匹配任何规则，缺省规则是禁止访问。关于 APL 资源记录的语法，参见 RFC 3123 。

通过在目录区的 zones 子域中包含一条 PTR 记录增加一个成员区。记录标记是成员区线上格式的一个 SHA-1 哈希。PTR 记录的目标是成员区名。例如，要添加成员区 domain.example ：

```
5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN PTR domain.example.
```

这个哈希是必须的，用以为一个特殊的成员区识别选项。成员区特殊的选项是与全局选项同样的方式定义的，只是在成员区子域中：

```
masters.5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN A 192.0.2.2
label.masters.5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN AAAA␣
↪2001:db8::2
label.masters.5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN TXT "tsig_
↪key"
allow-query.5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN APL 1:10.0.0.
↪0/24
```

正如预料的一样，为一个特定区定义的选项覆盖在目录区中定义的全局选项。这进而覆盖配置文件的 catalog-zones 语句中定义的全局选项。

注意，如果特定区为这个选项定义了任何记录，就不会从全球记录继承选项。例如，如果区有一个类型 A 而没有 AAAA 的 masters 记录，它 **不能**从全局选项继承类型 AAAA 记录。

第 5.15 节　BIND 9 对 IPv6 的支持

BIND 9 对当前所定义的各种 IPv6 形式的名字到地址和地址到名字的查找提供完全的支持。它也可以在具有 IPv6 的系统上使用 IPv6 地址来发出请求。

对正向的查找，BIND 9 仅支持 AAAA 记录。RFC 3363 废除了 A6 记录的使用，相应地，作为客户端对 A6 记录的支持也从 BIND 9 中去掉了。然而，权威 BIND 9 名字服务器仍然可以正确装载包含 A6 记录的区文件，回答对 A6 记录的请求，并接受包含 A6 记录的区的区传送。

对 IPv6 反向查找，BIND 9 支持传统的“半字节”格式，既可以用于 ip6.arpa 域，也可以用于旧的、被废除的 ip6.int 域。旧版本的的 BIND 9 支持“二进制标记”（也被称为“位串”）格式，但是

对二进制标记的支持已经完全被 RFC 3363 所去掉了。多数 BIND 9 中的应用完全不再识别二进制标记格式，如果遇到将会报错。特别的，一个权威 BIND 9 名字服务器将不再装载一个包含二进制标记的区文件。

对 IPv6 地址格式和结构的概括，参见IPv6 地址（AAAA） 。

5.15.1 使用 AAAA 记录查找地址

IPv6 的 AAAA 记录与 IPv4 的 A 记录相对应，并且与被废除的 A6 记录不同，它在一个记录中指定完整的 IPv6 地址。例如，

```
$ORIGIN example.com.
host        3600   IN    AAAA   2001:db8::1
```

不推荐使用 IPv6 内嵌 IPv4 映射地址。如果一个主机有一个 IPv4 地址，使用一个 A 记录，而不是带有 ::ffff:192.168.42.1 的 AAAA 记录来作为其地址。

5.15.2 使用半字节格式从地址查名字

在使用半字节格式来查找一个地址时，地址元素只是简单地反转，并且在反转之后的名字后面添加 ip6.arpa. ，就像在 IPv4 中一样。例如，下面将提供对一个地址为 2001:db8::1 的主机进行反向名字查找。

```
$ORIGIN 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0  14400  IN   PTR  (
            host.example.com. )
```

第六章　BIND 9 安全考虑

第 6.1 节　访问控制表

访问控制表（Access Control Lists，ACL）是地址匹配表，你可以建立并命名，以备以后用于 allow-notify，allow-query，allow-query-on，allow-recursion，blackhole，allow-transfer，match-clients 等语句中。

ACL 允许对谁能够访问名字服务器有一个更精细的控制，而不会用巨大的 IP 地址表将你的配置文件搞混乱。

使用 ACL 并控制对你的服务器的访问是一个 **好主意**。限制外来者访问你的服务器可以帮助阻止对你的服务器的欺骗和拒绝服务（DoS）攻击。

访问控制表根据三个特征来匹配客户端：1) 客户端的 IP 地址；2) 用于签名请求的 TSIG 或 SIG(0) 密钥，如果有的话；和 3) 编码在一个 EDNS 客户端子网选项中的一个地址前缀，如果有的话。

这是一个基于客户端地址的 ACL 的例子：

```
// Set up an ACL named "bogusnets" that blocks
// RFC1918 space and some reserved space, which is
// commonly used in spoofing attacks.
acl bogusnets {
  0.0.0.0/8; 192.0.2.0/24; 224.0.0.0/3;
  10.0.0.0/8; 172.16.0.0/12; 192.168.0.0/16;
};

// Set up an ACL called our-nets. Replace this with the
// real IP numbers.
acl our-nets { x.x.x.x/24; x.x.x.x/21; };
options {
 ...
 ...
```

(下页继续)

(续上页)

```
 allow-query { our-nets; };
 allow-recursion { our-nets; };
 ...
 blackhole { bogusnets; };
 ...
};

zone "example.com" {
 type master;
 file "m/example.com";
 allow-query { any; };
};
```

这将允许来自任何地址对 example.com 的权威请求，但是只有指定的网络 our-nets 才能进行递归请求，并且指定网络 bogusnets 中的任何请求都被禁止。

除了网络地址和前缀，这些与 DNS 请求中的源地址比对，之外，ACL 也可包含 key 元素，这是指定一个 TSIG 或者 SIG(0) 密钥的名字。

当 BIND 9 构建时带有 GeoIP 支持，ACL 也可用于地理访问限制。这是通过指定一个如下形式的 ACL 元素完成的：geoip db database field value 。

field 指定在一个匹配中搜索哪个字段。可用的字段是 country，region，city，continent，postal（邮政编码），metro （城区编码），area （区域编码），tz （时区），isp，asnum 和 domain 。

value 是要在数据库中搜索的值，字符串需要用引号引上，如果其中包含空格或其它特殊字符。一个对自治系统号的搜索 asnum 可以使用字符串 ASNNNN 或整数 NNNN 来指定。当一个 country 搜索由两个字符的字符串所指定时，它必须是一个标准的 ISO-3166-1 两字符国家码；否则它被解释成国家的全名。类似的，如果搜索条目是 region 并且字符串是两个字符，它被当成一个州或者省的标准两字母；否则它被当成州或省的全名。

database 字段指定在一个匹配中搜索哪个 GeoIP 数据库。大多数情况下，这不是必须的，因为大多数搜索字段仅能在一个数据库中找到。然而，对 continent 或 country 的搜索可以从 city 或 country 数据库中得到回答，所以对于这些搜索类型，指定一个 database 将强制请求只从那个数据库，而不能是其它的数据库，得到回答。如果未指定 database ，这些请求就将首先从 city 数据库得到回答，如果它被安装了，然后从 country 数据库得到回答，如果它被安装了。有效的数据库名字是 country ， city ， asnum ， isp 和 domain 。

一些 GeoIP ACL 的例子：

```
geoip country US;
geoip country JP;
geoip db country country Canada;
geoip region WA;
geoip city "San Francisco";
geoip region Oklahoma;
geoip postal 95062;
geoip tz "America/Los_Angeles";
geoip org "Internet Systems Consortium";
```

ACL 使用一个“首先匹配”逻辑，而不是“最佳匹配”：如果一个地址前缀与一个 ACL 元素匹配，则这个 ACL 就被认为匹配成立，即使一个稍后的元素将会匹配得更准确。例如，ACL { 10/8; !10.0.0.1; } 实际将会匹配来自 10.0.0.1 的请求，因为第一个元素表明请求应该被接受，第二个元素就被忽略。

当使用“嵌套”ACL 时（即，ACL 包含或者被包含于其它 ACL），一个嵌套 ACL 的否定匹配将导致外部 ACL 继续查找以成功匹配。这使得可以构造复杂的 ACL，在其中，多个客户端特性可以被同时检查。例如，要构造一个允许仅来自一个特定网络 **并且**仅当其被一个特定密钥签名的请求的 ACL，使用：

```
allow-query { !{ !10/8; any; }; key example; };
```

在嵌套 ACL 中，任何 **不在** 10/8 网络前缀中的地址都会被拒绝，这会终止 ACL 的处理过程。任何 **在** 10/8 网络前缀中的地址会被接受，但是这导致此嵌套 ACL 的一个否定匹配，所以外部 ACL 继续处理。如果请求被密钥 example 签名，它将被接受，否则将被拒绝。这样，这个 ACL 仅在 **两个**条件都为真时匹配。

第 6.2 节　Chroot 和 Setuid

在 UNIX 服务器上，可以通过为 named 设定 -t 选项使 BIND 运行在 chroot **的**环境中（使用 chroot() 函数）。这可以帮助增进系统的安全性，它通过将 BIND 放入一个“沙箱”，后者将在服务器遇到危险时把损坏限制在一个局部范围内。

UNIX 版本的 BIND 的另外一个能力是作为一个非特权用户（-u user）身份运行服务。我们建议在使用 chroot 特征时以一个非特权用户身份运行。

这里是有个命令行的例子，即将 BIND 加载到一个 chroot 沙箱，/var/named ，并且通过 setuid 以用户 202 的身份运行 named：

/usr/local/sbin/named -u 202 -t /var/named

6.2.1 chroot 环境

为了让一个 chroot 环境在一个特定目录（例如，/var/named ）中正常工作，环境必须包含 BIND 运行所需的所有东西。从 BIND 的视角，/var/named 是文件系统的根；必须调整像 directory 和 pid-file 这样的选项的值来满足这个需求。

与 BIND 的早期版本不同，典型地，不需要静态地编译 named ，也不需要在新的根下面安装共享库。然而，依赖于操作系统，可能需要设置诸如这样的路径：/dev/zero ，/dev/random ，/dev/log 以及 /etc/localtime 。

6.2.2 使用 setuid 函数

在运行 named 服务之前，在 BIND 要写的文件上使用 touch 应用程序（改变文件访问和修改时间）或者 chown 应用程序（设置用户 id 和/或组 id）。

注解：如果 named 后台进程是以一个非特权用户身份运行的，如果服务器重新加载，它将不能够绑定到新的受限端口上。

第 6.3 节 动态更新的安全

应该严格限制对动态更新设施的访问。在早期的 BIND 版本中，仅有的方法是基于请求进行动态更新的主机的 IP 地址，通过在 allow-update 区选项中列出一个 IP 地址或者网络前缀来实现。由于更新 UDP 包的源地址非常容易伪造，这个方法是不安全的。另外要注意的是，如果 allow-update 选项中所允许的地址包含一个可以执行转发动态更新的辅服务器的 IP 地址，主服务器将可能遭受到很简单的攻击，即通过发送更新到辅服务器，辅服务器使用自己的地址作源地址将其转发到主服务器 - 使主服务器毫无怀疑地接受。

由于这些原因，我们强烈推荐通过事务签名（TSIG）来加密和认证所进行的更新。这就是说，allow-update 选项仅应该列出 TSIG 密钥名，而不是 IP 地址或网络前缀。作为另外的选择，也可以使用新的 update-policy 选项。

一些站点选择将所有的动态更新的 DNS 数据保存到一个子域并将子域授权到一个单独的区。在这种方法中，包含像公共 web 和邮件服务器的 IP 地址这样的关键数据的顶级区就完全不允许对其进行动态更新。

第七章　排除故障

第 7.1 节　常见问题

7.1.1　它不工作；我如何判定哪里出错了？

解决安装和配置问题的最好方法是通过预先设置日志文件来采取预防性的措施。日志文件提供了暗示和信息的源头，后者用于识别哪里出现了错误以及如何来解决问题。

7.1.2　EDNS 合规性问题

EDNS（扩展的 DNS）是一项最早在 1999 年指定的标准。它是 DNSSEC 验证，DNS COOKIE 选项以及其它特性的基础。仍然有一些故障的或者过时的 DNS 服务器和防火墙在使用，它们在收到带 EDNS 的请求时会产生错误的行为；例如，它们会丢弃 EDNS 请求而不是回复 FORMERR。BIND 和其它递归名字服务器在面对这个情况时，习惯上采用变通的方法，以不同方式重试，最终回退到不带 EDNS 的普通 DNS 请求。

这样的变通方法导致不必要的解析延迟，增加代码复杂性，并阻碍开发新的 DNS 特性。2019 年 2 月，所有主流的 DNS 软件开发商都同意去除这些变通方法；参见 https://dnsflagday.net 以获取更多细节。这个变化在 BIND 的 9.14.0 版本开始实现。

作为一种结果，在没有人工干预时，某些域名可能不能解析。在这种情况下，可以针对有问题的服务器添加 server 子句来恢复解析，指定 edns no 或 send-cookie no ，依赖于具体的不合规情况。

要决定使用哪个 server 子句，运行下列命令发送请求给有问题域名的权威服务器：

```
dig soa <zone> @<server> +dnssec
dig soa <zone> @<server> +dnssec +nocookie
dig soa <zone> @<server> +noedns
```

如果第一条命令失败但是第二条成功，服务器很可能需要 send-cookie no 。如果头两条失败但是第三条成功，服务器需要通过 edns no 将 EDNS 完全关闭。

请联系不合规域名的管理员，并鼓励他们升级他们的有问题的 DNS 服务器。

第 7.2 节　增加和修改序列号

区的序列号只是数字—它们与日期没有联系。大多数人将它们设置成表示日期的一个数，通常是 YYYYMMDDRR 的格式。偶尔地，他们会错误地将它们设置为一个“未来的日期”然后试图通过将其设置为“当前的日期”来纠正错误。这就产生了问题，因为序列号是用来指示一个区被更新了。如果一个辅服务器上的序列号小于主服务器上的序列号，辅服务器将试图更新它的区拷贝。

将主服务器的序列号设置成为一个比辅服务器上更小的数意味着辅服务器将不会执行对它的区拷贝的更新。

解决方案是将数字增加 2147483647（2^31-1），重新装载区并确保所有的辅服务器都更新为新的区序列号，然后重新设置数字为你想要的，再重新装载一次区。

第 7.3 节　从哪里获得帮助?

BIND 用户邮件列表，在 https://lists.isc.org/mailman/listinfo/bind-users ，是用户间互相支持的优秀资源。另外，ISC 维护了一个有帮助文档的知识库，在 https://kb.isc.org 。

互联网系统联盟（Internet Systems Consortium，ISC）提供了对 BIND 9，ISC DHCP 和 Kea DHCP 的年度支持协议。所有付费支持合同包括高级安全通知；一些级别包含服务级别协议（service level agreements，SLA），优质软件特性，以补丁修补和特性需求的增强优先级。

更多信息，请联系 info@isc.org 或访问 https://www.isc.org/contact/ 。

第八章　发行注记

目录

- 发行注记
  - 介绍
  - 关于版本编号的注释
  - 支持的平台
  - 下载
  - BIND 9.16.8 注记
    - 新特性
    - 特性变化
    - 漏洞修补
  - BIND 9.16.7 注记
    - 新特性
    - 漏洞修补
  - BIND 9.16.6 注记
    - 安全修补
    - 新特性
    - 特性变化
    - 漏洞修补
  - BIND 9.16.5 注记

    * 新特性
    * 漏洞修补
- BIND 9.16.4 注记
    * 安全修补
    * 新特性
    * 特性变化
    * 漏洞修补
- BIND 9.16.3 注记
    * 已知问题
    * 特性变化
    * 漏洞修补
- BIND 9.16.2 注记
    * 安全修补
    * 已知问题
    * 特性变化
    * 漏洞修补
- BIND 9.16.1 注记
    * 已知问题
    * 特性变化
    * 漏洞修补
- BIND 9.16.0 注记
    * 新特性
    * 特性变化
    * 去掉的特性
- 许可证
- 生命周期结束

– 谢谢您

第 8.1 节　介绍

BIND 9.16 是 BIND 的一个稳定分支。本文档汇总了此分支上自上一个产品版本以来的重大变化。关于变化和漏洞修订的一个更详细的清单，请参见文件 CHANGES。

第 8.2 节　关于版本编号的注释

截止 BIND 9.13/9.14，BIND 采用了“奇数-不稳定/偶数-稳定”的版本编号规范，BIND 9.16 包含了在 BIND 9.15 开发过程中增加的新特性。今后，9.16 分支将被限制于漏洞修订，而新特性开发将在不稳定的 9.17 分支中进行。

第 8.3 节　支持的平台

要在类 UNIX 系统上编译，BIND 要求支持 POSIX.1c 线程（IEEE Std 1003.1c-1995），关于 IPv6 的高级套接字 API（RFC 3542 ），和由 C 编译器提供的标准原子操作。

在目标平台上，libuv 异步 I/O 库和 OpenSSL 加密库必须是可用的。对于公共密钥加密（即，DNSSEC 签名和验证），可以使用一个 PKCS#11 提供者来代替 OpenSSL。但是仍然需要 OpenSSL 进行通常的加密操作，诸如哈希和随机数生成。

更多信息可以在 PLATFORMS.md 文件中找到，它被包含在 BIND 9 的源码发布中。如果你的编译器和系统库提供了上述特征，BIND 9 就应当能够编译和运行。如果不是这种情况，BIND 开发团队通常接受补丁以支持仍然被各自厂商支持的系统。

第 8.4 节　下载

最新版本的 BIND 9 软件总是可以在 https://www.isc.org/download/ 找到。那里，你会发现每个版本的附加信息，源码和为微软 Windows 操作系统预编译的版本。

第 8.5 节　BIND 9.16.8 注记

8.5.1　新特性

- 增加一个新的 rndc 命令，rndc dnssec -rollover，它针对一个特定的密钥启动一次手动轮转。[GL #1749]
- 增加一个新的 rndc 命令，rndc dumpdb -expired，它转储缓存数据库，其中包括等待清理的已过期的资源记录集，到 dump-file，出于诊断的目的。[GL #1870]

8.5.2　特性变化

- 2020 DNS 旗帜日：缺省的 EDNS 缓存大小被从 4096 修改为 1232 字节。根据多家机构已完成的测量，这应当不会带来任何操作问题，因为大多数互联网“核心”能够处理介于 1400-1500 字节之间的 IP 消息大小；1232 的大小被选择作为保守的最小值，DNS 操作人员可以将其修改为估计的路径 MTU 减去估计的头空间。在实践中，在运营 DNS 的社区中可见的最小 MTU 是 1500 字节，这是最大的以太网负载大小，因此在可靠的网络中，最大 DNS/UDP 负载大小的一个有益的缺省值是 1400 字节。[GL #2183]

8.5.3　漏洞修补

- 在一个不能正确报告可用内存页数量和/或每个内存页大小的环境中运行时，named 报告了一个无效的内存大小。[GL #2166]
- 当配置了多个转发者时，named 可能因为在 lib/dns/message.c 中的 REQUIRE(msg->state == (-1)) 断言失败而导致崩溃。这个已被修正。[GL #2124]
- 对于使用 Ed25519 或 Ed448 算法的 KASP 策略，由于创建的密钥大小与预期的密钥大小不匹配，named 错误地执行连续的密钥轮转。[GL #2171]
- 如果一个 RPZ 区包含的名字混用大小写字母，在更新其内容时，可能会导致错误地忽略该 RPZ 区域中的一些处理规则。[GL #2169]

第 8.6 节　BIND 9.16.7 注记

8.6.1　新特性

- 增加了一个新的 rndc 命令，rndc dnssec -checkds ，它发送信号给 named ，一个给定区或密钥的一条 DS 记录已被发布到父区或者已从父区撤回。这个命令替代料基于时间的 parent-registration-delay 配置选项。[GL #1613]
- 当 named 增加一条 CDS/CDNSKEY 记录到区中时记录日志。[GL #1748]

8.6.2　漏洞修补

- 在罕见的情况下，当存储在红黑树中的节点数目超过内部哈希表所允许的最大数目时，named 可能会发生一个断言失败并退出。[GL #2104]
- 消除在旧操作系统上看到的 EPROTO(71) 错误代码的虚假系统日志消息，未处理的 ICMPv6 错误导致返回一个通用协议错误，而不是更具体的错误代码。[GL #1928]
- 当开启了查询名字最小化时，named 在解析 IPv6 部份左边有额外标记的 ip6.arpa. 名字时会失败。例如，当 named 试图在一个类似 A.B.1.2.3.4.(...).ip6.arpa. 的名字上的请求名最小化时，它在最左边的 IPv6 标记，即 1.2.3.4.(...).ip6.arpa. ，位置停止，而不考虑额外标记（A.B ）。这会导致在解析名字时的请求循环：如果 named 收到 NXDOMAIN 答复，同样的请求重复发送直到发送的请求数达到 max-recursion-queries 配置选项的值。[GL #1847]
- 通过拒绝一个单一点（. ）和/或 m 作为值而对 LOC 记录的分析变得更加严格。这些变化阻止区文件在加载时使用这样的值。对负海拔不是整数的处理也已修正。[GL #2074]
- 修正了几个由 OSS-Fuzz 发现的问题。（这些都不是安全问题。）[GL !3953] [GL !3975]

第 8.7 节　BIND 9.16.6 注记

8.7.1　安全修补

- 通过发送一个特别构造的大 TCP DNS 消息，可能触发一个断言失败。这在 CVE-2020-8620 中公开。

  ISC 要感谢思科系统公司的 Emanuel Almeida，是他让我们注意到这个漏洞。[GL #1996]

- 在 QNAME 最小化和转发都被开启的某种请求解析场景中，named 可能会在一个断言检查失败后崩溃。为防止这种崩溃，如果在任何使用转发的时候，对于给定请求的解析过程，现在总是禁用 QNAME 最小化。这在 CVE-2020-8621 中公开。

  ISC 要感谢 Joseph Gullo，是他让我们注意到这个漏洞。[GL #1997]

- 当验证一个 TSIG 签名请求的响应时，可能触发一个断言失败。这在 CVE-2020-8622 中公开。

  ISC 要感谢 Oracle 公司的 Dave Feldman，Jeff Warren 和 Joel Cunningham，是他们让我们注意到这个漏洞。[GL #2028]

- 当 BIND 9 是带有原生 PKCS#11 支持编译时，其用于确定 PKCS#11 RSA 公钥位数的代码，在遇到一个特别构造的包时可能触发一个断言失败。这在 CVE-2020-8623 中公开。

  ISC 要感谢 Lyu Chiy，是他让我们注意到这个漏洞。[GL #2037]

- subdomain 类型的 update-policy 规则被错误当成 zonesub 规则处理，这允许 subdomain 规则中使用的密钥更新特定子域之外的名字。这个问题已被修正，通过确认 subdomain 规则是按照 ARM 中的描述再次被处理。

  ISC 要感谢 credativ GmbH 公司的 Joop Boonen，是他让我们注意到这个漏洞。[GL #2055]

8.7.2 新特性

- 引入了一个新的配置选项 stale-cache-enable ，用以开启或关闭在缓存中保留旧答复。[GL #1712]

8.7.3 特性变化

- BIND 的缓存数据库实现已被更新，使用了更好的发行版中的更快的哈希函数。此外，有效的 max-cache-size （显式配置，默认值基于系统内存或设置为 unlimited ）现在可以预分配固定大小的哈希表。这可以防止在需要增加哈希表大小时中断查询解析。[GL #1775]

- 收到的 TTL 为 0 的资源记录不再保留在缓存中，以用于旧答复。[GL #1829]

8.7.4 漏洞修补

- 通配符 RPZ passthru 规则可能会被稍后出现在 response-policy 语句中从 RPZ 区域加载的其它规则错误地覆盖。这个已被修订。[GL #1619]

- IPv6 重复地址检测（Duplicate Address Detection，DAD）机制可能无意中阻止 named 绑定到新的 IPv6 接口，通过为每个 IPv6 地址发送多个路由套接字消息。当其配置成监听 any 或一个特定范围的地址时，named 监视新接口的 bind() 。新的 IPv6 接口在完全可用之前可能处于“试探性”状态。当使用了 DAD，路由套接字发送两个消息：一个是接口首次出现时，然后第二个是当其完全“完成”。named 过早尝试 bind() 到新接口将会失败。通过在套接字上设置 IP_FREEBIND 选项并且如果在首次对此地址的 bind() 调用失败并带有 EADDRNOTAVAIL 时，重新尝试 bind() 到每个 IPv6 地址，这个问题已被解决。[GL #2038]
- 解决了递归客户端统计报告中的一个错误，该错误可能导致下溢，甚至是负统计数据。在某些情况下，进入的请求可能会触发对某些符合条件的资源记录集的预取，如果预取代码在递归之前执行，那么递归客户端统计不会增加。相反，在处理回复时，如果递归代码在预取之前执行，则相同的计数器将被递减而没有对应的增加。[GL #1719]
- 引入 KASP 支持无意中导致第二个字段 sig-validity-interval 总是以小时计算，即使在应该以天计算的情况下也是如此。这个已被修订。（感谢 Tony Finch。）[GL !3735]
- 修订了 LMDB 锁代码，以使 rndc reconfig 可以在 FreeBSD 且 LMDB >= 0.9.26 下面正常工作。[GL #1976]

第 8.8 节　BIND 9.16.5 注记

8.8.1 新特性

- 新的 rndc 命令 rndc dnssec -status 显示当前的 DNSSEC 策略和使用的密钥，密钥的状态和轮转状态。[GL #1612]

8.8.2 漏洞修补

- 在 named 正在等待一个递归响应时，如果一个 TCP 套接字被关闭，可能会发生一个竞争条件。试图发送一个响应到正在关闭的连接会触发函数 isc__nm_tcpdns_send() 中的一个断言失败。[GL #1937]
- 当 named 试图使用一个正在关闭的 UDP 接口时可能会发生一个竞争条件。这会触发 uv__udp_finish_close() 中的一个断言失败。[GL #1938]
- 修复了当服务器处于负载状态且根区尚未加载时断言失败的问题。[GL #1862]
- 当清理 lib/dns/rbtdb.c 中被重新使用的死节点时，named 可能崩溃。[GL #1968]
- 当服务进程停止时收到一个新的 rndc 连接，named 会崩溃。[GL #1747]

- 由 dns_keynode_dsset() 返回的 DS 资源记录集被用于一个非线程安全的方式。这可能导致一个 INSIST 被触发。[GL #1926]
- 当安装 CMocka 时，会正确处理缺失的 kyua 命令，make check 不会意外失败，但是 Kyua 不会。[GL #1950]
- 当用作 check-names 的参数时，primary 和 secondary 关键字未被正确处理，而被忽略了。[GL #1949]
- rndc dnstap -roll <value> 没有限制保存文件的数目为 <value> [GL !3728]
- 如果 DNSKEY 资源记录集中不支持的算法比支持的算法出现更早，在接受一个正确签名的资源记录集时，验证器可能失败。如果它检测到一个错误格式的公钥，它也可能停止。[GL #1689]
- 在客户端查询时，无意中禁用了 blackhole ACL。被阻止的 IP 地址不能用于上游查询，但是来自这些地址的查询仍然可以被回答。[GL #1936]

第 8.9 节　BIND 9.16.4 注记

8.9.1　安全修补

- 当试图填充一个超大的 TCP 缓冲区时，可能触发一个断言。这个在 CVE-2020-8618 中披露。[GL #1850]
- 当以某个模式查询一个带有内部通配符标记的区时，可能触发一个 INSIST 失败。这个在 CVE-2020-8619 中披露。[GL #1111] [GL #1718]

8.9.2　新特性

- 文档从 DocBook 转换为 reStructuredText。BIND 9 ARM 现在使用 Sphinx 生成并发布于 Read the Docs。发行注记不再作为一个独立文档伴随发行版一起提供。[GL #83]
- named 和 named-checkzone 现在会拒绝加载主区，当其在区顶点有一个 DS 资源记录集时。通过 UPDATE 在区顶点添加 DS 记录会被记入日志，并忽略。DS 记录属于父区，不应在区顶点。[GL #1798]
- dig 和其它工具现在输出扩展 DNS 错误（Extended DNS Error，EDE）选项，当其出现在一个请求或响应中时。[GL #1835]

8.9.3 特性变化

- max-stale-ttl 的缺省值从 1 周改为 12 小时。如果一个或多个域出现问题，这个选项控制 named 在缓存中保留过期资源记录集的时间，作为一种潜在的缓解机制。注意，缓存内容的保留与是否在响应客户机查询时使用陈旧的答案无关（stale-answer-enable yes|no 和 rndc serve-stale on|off ）。当权威服务器没有响应而使用陈旧答案的功能必须显示开启，在具有这个特性的所有 BIND 9 版本上，将自动保留过期的缓存内容。[GL #1877]

  警告：这个变化可能对于那些期望陈旧的缓存内容会自动保留 1 周的管理员来说是值得注意的。在 named.conf 中增加选项 max-stale-ttl 1w; 以保持 named 之前的行为。

- listen-on-v6 { any; } 为每个接口创建一个独立的套接字。之前，只在遵循 RFC 3493 和 RFC 3542 的系统上创建一个套接字。这个变化自 BIND 9.16.0 引入，但它意外地从文档中遗漏了。[GL #1782]

8.9.4 漏洞修补

- 当通过 IXFR 全部更新一个大区的 NSEC3 链时，当回应对不存在数据的请求时，需要 DNSSEC 证明不存在，可能会在辅服务器上出现暂时的性能损失（换句话说，请求要求服务器发现并返回 NSEC3 数据）。导致这种延迟的不必要的处理步骤现在已被删除。[GL #1834]
- 如果一个数据库节点的名字在查找时，数据库被修改，named 可能会崩溃，并带有一个断言失败。[GL #1857]
- 一个在 lib/isc/unix/socket.c 中可能的死锁被修正了。[GL #1859]
- 之前，named 在 netmgr 代码中不会销毁某些互斥量和条件变量，这会在 FreeBSD 中导致一个内存泄漏。这个已被修正。[GL #1893]
- 在 lib/dns/resolver.c:log_formerr() 中的一个数据竞争可能导致一个断言失败，已被修正。[GL #1808]
- 之前，当序列号大于或等于当前序列号时，provide-ixfr no; 无法返回最新的响应。[GL #1714]
- dnssec-policy keymgr 中的一个缺陷已被修正，对给定密钥的后继是否存在的检查会错误地返回 true ，如果密钥环（keyring）中的任何其它密钥具有一个后继。[GL #1845]
- 当使用 dnssec-policy 创建一个后继密钥时，当前活跃密钥（前驱）的 “goal” 状态不会改变，因而永远不会从区中删除。[GL #1846]

- 由于一个未初始化的 DSCP 值，named-checkconf -p 可能在 server-addresses 语句中包含虚假的文本。这个已被修正。[GL #1812]
- ARM 已被更新，指示当 named 启动时，生成 TSIG 会话密钥，而不管是否需要它。[GL #1842]

第 8.10 节　BIND 9.16.3 注记

8.10.1　已知问题

- BIND 在链接到 libuv 1.36 时，启动时会宕掉。这个问题与 libuv 中对 recvmmsg() 的支持相关，在 libuv 1.35 中首次出现。这个问题在 libuv 1.37 中解决了，但是相关的 libuv 代码进行了修改，要求在库的初始化时，为了打开对 recvmmsg() 的支持，需要设置一个特定的标志。本 BIND 版本在需要时设置了这个特定的标志，所以 BIND 在针对 libuv 1.35 或 libuv >= 1.37 编译时开启了对 recvmmsg() 的支持；libuv 1.36 不能与 BIND 一起使用。[GL #1761] [GL #1797]

8.10.2　特性变化

- BIND 9 不再为 UDP 套接字设置接收/发送缓冲区大小，而是依赖系统缺省值。[GL #1713]
- 缺省的 rwlock 实现被修改为回退到原生的 BIND 9 rwlock 实现。[GL #1753]
- 原生的 PKCS#11 EdDSA 实现被更新到 PKCS#11 v3.0，这样就又可以使用了。由 Aaron Thompson 贡献。[GL !3326]
- OpenSSL ECDSA 实现被更新，以通过 OpenSSL 引擎支持 PKCS#11（参见 lip11 项目的 engine_pkcs11）。[GL #1534]
- OpenSSL EdDSA 实现被更新，以通过 OpenSSL 引擎支持 PKCS#11。请注意支持 EdDSA 的 OpenSSL 引擎是必须的，因而这个代码仅是暂时为了概念验证的。由 Aaron Thompson 贡献。[GL #1763]
- 进入 AXFR 区传送中的消息 ID 现在进行一致性检查。对于消息 ID 不一致的流，将记录到日志。[GL #1674]
- 区计时器现在会输出到统计通道。对于主区，只有装载时间会输出。对于辅区，输出的计时器还包括过期和刷新计时器。由 Paul Frieden，Verizon Media 贡献。[GL #1232]

8.10.3 漏洞修补

- dnstap 初始化中的一个漏洞可能阻止某些 dnstap 数据被记录，特别是对于递归解析器。[GL #1795]
- 当运行在一个具有 Linux 能力的系统上时，named 在启动后会尽快放弃根特权。这将产生一个误导性的日志消息，unable to set effective uid to 0: Operation not permitted ，这个已被去掉。[GL #1042] [GL #1090]
- 当 named-checkconf 运行时，有时它会错误地设置其退出码。它只反映所发现的最后一个视图的状态；对其它配置视图所发现的任何错误都不会报告。感谢 Graham Clinch. [GL #1807]
- 当不带 LMDB 支持编译时，当一个区的名字中带有一个双引号（"）并使用 rndc addzone 增加时，named 的重启会失败。感谢 Alberto Fernández. [GL #1695]

第 8.11 节 BIND 9.16.2 注记

8.11.1 安全修补

- 当 BIND 9 被配置为一个转发服务器时，DNS 重绑定保护失效。由 Tobias Klein 发现并负责地报告。[GL #1574]

8.11.2 已知问题

- 我们收到一些报告，在某些环境下，接收一个 IXFR 时可能导致对请求的处理显著变慢。其中一些与 RPZ 处理相关，这在新版本中已被修订（参见下面）。其它似乎发生在 NSEC3 相关的变化中（例如一个操作员修改了用于散列计算中的 NSEC3 salt）。这些正在研究中。[GL #1685]

8.11.3 特性变化

- 先前的 DNSSEC 签名统计使用了大量的内存。每个区要跟踪的密钥数目减少到四个，这对 99% 的全签名区已经足够。[GL #1179]

8.11.4 漏洞修补

- 当一个 RPZ 策略区通过区传送更新且有大量记录被删除时，named 可能有一小段时间，即从 RPZ 概要数据库中删除名字这段时间，没有响应。现在这个数据库经过一个较长的时间段就会增量地清理，减少了这个延迟。[GL #1447]
- 当试图从 auto-dnssec maintain 到基于 dnssec-policy 迁移一个已签名的区时，已存在的密钥会立即被删掉并被新的密钥所替代。由于密钥轮转计时限制未被遵守，可能某些客户端无法验证响应，直到所有的旧 DNSSEC 信息都从缓存中过期。BIND 现在查找现存密钥的时间元数据并使其并入 DNS 策略操作。[GL #1706]

第 8.12 节 BIND 9.16.1 注记

8.12.1 已知问题

- 用于监听的 UDP 网络端口不能再同时用于发送流量。一个会触发这个问题的例子配置可能会是一个将同样的 address:port 对用于 listen-on(-v6) 语句且用于 notify-source(-v6) 或 transfer-source(-v6) 语句。而这个问题会影响到所有操作系统，在其中一些系统上它只会记录日志（例如，“unable to create dispatch for reserved port”）。当前还没有计划使这样的组合设置能够重新工作。

8.12.2 特性变化

- 系统所提供的 POSIX 线程读写锁实现现在作为缺省项，替代了原生的 BIND 9 实现。请注意 glibc 版本 2.26 到 2.29 有一个 bug，可能导致 BIND 9 死锁。对此的修补发布在 glibc 2.30 中，并且当前大多数 Linux 发行都已修补或更新了 glibc，但有一个值得注意的例外，Ubuntu 18.04 (Bionic) 正在修补中。如果你在一个受影响的操作系统上运行，编译 BIND 9 时带上 --disable-pthread-rwlock，直到一个 glibc 的修补版本可用为止。[GL !3125]

8.12.3 漏洞修补

- 修正了在内联区中重签的问题，即导致签名延迟或者完全没有签名。

第 8.13 节　BIND 9.16.0 注记

注意：这个部份仅列出自 BIND 9.14（上一个稳定的 BIND 分支）以来的变化。

8.13.1　新特性

- 一种基于 libuv 的新的异步网络通信系统现在用于 named 中监听进入的请求和对其响应。这个变化将使提升性能和在未来实现新的协议层（例如，DNS over TLS）更容易。[GL #29]
- 新的 dnssec-policy 选项允许为区配置一个密钥和签名策略（KASP）。这个选项使 named 能够按需生成新密钥并自动轮转 ZSK 和 KSK 密钥。（注意这个语句的语法与 dnssec-keymgr 所用的 DNSSEC 策略不同。）[GL #1134]
- 为了使 DNSSEC 密钥的配置更清晰，trusted-keys 和 managed-keys 语句都被作废了，而新的 trust-anchors 语句现在应该可以用于两种类型的密钥。

  当与关键字 initial-key 一起使用时，trust-anchors 具有与 managed-keys 同样的特性，即它配置一个通过 RFC 5011 维护的信任锚。

  当与新的关键字 static-key 一起使用时，trust-anchors 具有与 trusted-keys 同样的特性，即它配置一个永久信任锚，不会自动更新。这个用法不推荐用于根密钥。）[GL #6]
- trust-anchors 语句中增加了两个新的关键字：initial-ds 和 static-ds 。这些允许在使用信任锚时以 DS 格式替代 DNSKEY 格式。DS 格式允许信任锚配置还没有发布的密钥；这是 IANA 宣布未来根密钥所用的格式。

  与关键字 initial-key 和 static-key 一样，initial-ds 配置一个由 RFC 5011 维护的动态信任锚，而 static-ds 配置一个永久信任锚。[GL#6] [GL #622]
- dig ，mdig 和 delv 现在都可以接受一个 +yaml 选项并以一个详细的 YAML 格式打印输出。[GL #1145]
- dig 现在有了一个新的命令行选项：+[no]unexpected 。缺省时，dig 不会接受一个源地址不是它所请求的地址的回复。增加 +unexpected 参数使其可以处理来自非预期源地址来的回复。[RT #44978]
- dig 现在接受一个新的命令行选项，+[no]expandaaaa ，它使 AAAA 记录中的 IPv6 地址以完整的 128-位符号输出，而不是缺省的 RFC 5952 格式。[GL #765]
- 统计通道组现在可以被切换了。[GL #1030]

8.13.2 特性变化

- 当同一个名字配置了静态和受管理的 DNSSEC 密钥，或者当一个静态密钥被用于配置一个根区的信任锚并且 dnssec-validation 被设置成缺省值 auto ，自动 RFC 5011 密钥轮转将被关闭。这个组合设置永远不会工作，但是在语法分析时却没有检查。这个已经被纠正，现在它是一个致命的配置错误。[GL #868]
- DS 和 CDS 记录现在只使用 SHA-256 摘要生成，取代了使用 SHA-1 和 SHA-256。这影响 dnssec-dsfromkey 的缺省输出、dnssec-signzone 生成的 dsset 文件、基于 keyset 文件由 dnssec-signzone 添加到一个区的 DS 记录、基于密钥文件中的 "sync" 时间参数由 named 和 dnssec-signzone 添加到一个区的 CDS 记录以及 dnssec-checkds 执行的检查。[GL #1015]
- 如果为根区配置了一个静态密钥，named 将会记录一条警告日志。[GL #6]
- 增加了一个基于 SipHash 2-4 的 DNS Cookie (RFC 7873) 算法，并作为缺省的。旧的非缺省基于 HMAC-SHA 的 DNS Cookie 算法被去掉了，只有缺省的 AES 算法因为遗留（legacy）的原因而被保留。这个变化对大多数普通场景没有操作上的影响。[GL #605]

  如果你运行着多个 DNS 服务器（BIND 9 的不同版本或者来自多个厂商的 DNS 服务器），响应来自于同一个 IP 地址（anycast 或者负载均衡场景），要确认所有的服务器配置成同样的 DNS Cookie 算法和同一个服务器密码，以获得最好的性能。
- 来自 dnssec-signzone 和 dnssec-verify 的信息现在被输出到标准输出。标准错误输出仅用于输出警告和错误，以及用户请求使用 -f - 选项将签名区输出到标准输出。增加了一个新的配置选项 -q ，以关闭标准输出上的所有输出，除了签名区的名字。[GL #1151]
- DNSSEC 验证代码已被重构，为了更加清晰和减少代码重复。[GL #622]
- configure 的 --with-tuning=large 选项所开启的编译时设置现在已经缺省生效。之前使用的缺省编译时设置可以在 configure 时使用 --with-tuning=small 来开启。[GL !2989]
- JSON-C 是 BIND 统计中唯一支持的 JSON 库。configure 选项从 --with-libjson 改名为 --with-json-c 。需要根据 json-c 库的定制路径相应设置 PKG_CONFIG_PATH 环境变量，因为新的 configure 选项不会将库安装路径作为一个可选参数。[GL #855]
- 当没有指定 --prefix 并且没有显式指定 --sysconfdir 和 --localstatedir 时，./configure 不会将 --sysconfdir 设置为 /etc ，或将 --localstatedir 设置为 /var 。作为替代，两者将分别被设置为 Autoconf 的缺省值 $prefix/etc 和 $prefix/var 。[GL #658]

8.13.3 去掉的特性

- dnssec-enable 选项已被作废并不再有效了。如果签名并且具有其它的 DNSSEC 数据，DNSSEC 响应总是被开启。[GL #866]
- DNSSEC 后备验证（DLV）现已被作废。dnssec-lookaside 被标记为废弃；当用于 named.conf 时，它将生成一个警告并被忽略。所有开启了使用后备验证的代码都已从验证器，delv 和 DNSSEC 工具中删除了。[GL #7]
- cleaning-interval 选项已被删除。[GL !1731]

第 8.14 节 许可证

BIND 9 是一个遵循 Mozilla 公共许可证，版本 2.0（参见 LICENSE 文件获得完整文本）的开源软件。

这个许可证要求如果你修改 BIND 并分发到你的组织之外，这些修改必须在同样的许可证下公开。它不要求你发布或公开你没有修改的部份。这个要求对于使用 BIND 的，无论修不修改，只要没有重分发的人以及对分发 BIND 但是没有修改的人没有影响。

那些希望讨论许可证遵守性的人可以在 https://www.isc.org/contact/ 联系 ISC。

第 8.15 节 生命周期结束

BIND 9.16 的生命结束日期还未决定。在未来的某个时间点 BIND 9.16 将被指定为延长支持版本（ESV）。直到那时，当前的 ESV 是 BIND 9.11，将会至少被支持到 2021 年 12 月。参见 https://kb.isc.org/docs/aa-00896 以获取 ISC 的软件支持策略的详细信息。

第 8.16 节 谢谢您

感谢每一个帮助我们的您，使得本版本得以发布。

第九章　DNS 和 BIND 的简要历史

尽管域名系统“官方地”开始于 1984 年，随着 RFC 920 的公布，但这个新系统的核心是由 1983 年的 RFC 882 和 RFC 883 描述的。从 1984 年到 1987 年，ARPAnet（今天的互联网的前身）成为一个在快速扩展的、运转中的网络环境中开发新的命名/寻址机制的试验床。新的 RFC 于 1987 年被写作并出版，它们修改了原始的文档以合并进基于这个工作模式的进展。RFC 1034，“域名–概念和设施”和 RFC 1035，“域名–实现和规范”出版并成为所有 DNS 实现的标准。

第一个可以工作的域名服务器，名叫 “Jeeves”，由 Paul Mockapetris 于 1983-84 年在位于南加州大学信息科学研究所（USC-ISI）和 SRI 国际网络信息中心（SRI-NIC）上的 DEC Tops-20 机器上写成的。一个在 Unix 上的 DNS，伯克利互联网名字域（Berkeley Internet Name Domain，BIND）包，是稍后由一组加州大学伯克利分校的研究生在美国国防部高级研究项目管理局（DARPA）的资助下完成的。

BIND 版本直到 4.8.3 都是由在加州大学伯克利分校的计算机系统研究组（CSRG）所维护。Douglas Terry，Mark Painter，David Riggle 和 Songnian Zhou 组成了最初的 BIND 项目组。之后，Ralph Campbell 在软件包上做了一些增加的工作。Kevin Dunlap，一个数字设备公司（DEC）的雇员，作为赞助给 CSRG 的资源，为 BIND 工作了 2 年，从 1985 年到 1987 年，在此期间还有许多其他人也对 BIND 的开发作出了贡献：Doug Kingston，Craig Partridge，Smoot Carl-Mitchell，Mike Muuss，Jim Bloom 和 Mike Schwartz。BIND 的维护随后被移交给了 Mike Karels 和 Øivind Kure。

BIND 版本 4.9 和 4.9.1 由数字设备公司发行（现在是 Compaq 计算机公司）。Paul Vixie，那时是 DEC 的一名雇员，成为了 BIND 的主要日常维护者。他的助手有 Phil Almquist，Robert Elz，Alan Barrett，Paul Albitz，Bryan Beecher，Andrew Partan，Andy Cherenson，Tom Limoncelli，Berthold Paffrath，Fuat Baran，Anant Kumar，Art Harkin，Win Treese，Don Lewis，Christophe Wolfhugel 及其他一些人。

在 1994 年，BIND 版本 4.9.2 由 Vixie 公司赞助。Paul Vixie 成为 BIND 的首席架构师/程序员。

BIND 版本自 4.9.3 之后由互联网系统联盟及其前身互联网软件联盟进行开发和维护，并由 ISC 的赞助商提供支持。

作为共同架构设计师和程序员，Bob Halley 和 Paul Vixie 在 1997 年 5 月发布了第一个 BIND 版

本 8 的产品级版本。

BIND 版本 9 于 2000 年 9 月发布，它几乎重写了 BIND 体系结构的各个方面。

BIND 版本 4 和 8 已经正式被废弃了。不再在 BIND 版本 4 或 BIND 版本 8 上做进一步的开发了。

今天，在从 ISC（https://www.isc.org/contact/）购买专业支持服务和/或为我们的任务捐款的一些公司的赞助下，并且经过许多人的不懈地努力工作，BIND 开发工作才成为可能。

第十章　通用 DNS 参考信息

第 10.1 节　IPv6 地址（AAAA）

IPv6 地址是 128 位接口和接口集合的标识符，它被引入 DNS 以促进可扩展的互联网路由。有三种类型的地址：Unicast，标识单个接口；Anycast，标识一组接口；和 Multicast，标识一组接口。在这里，我们描述全球 Unicast（单播）地址机制。更多信息，参见 RFC 3587，“IPv6 全局单播地址格式”。

IPv6 单播地址由一个 **全局路由前缀**，一个 **子网标识符**，和一个 **接口标识符**所组成。

全局路由前缀由上层提供者或 ISP 提供，并且大致地与 IPv4 地址范围的 **网络**部份相对应。子网标识符标识局域子网，就像在 IPv4 的/16 网络中划分出/24 的子网。接口标识符是一个给定网络上的单个地址；在 IPv6 中，地址属于接口而不属于主机。

IPv6 划分子网的能力比 IPv6 更灵活：子网划分可以在位边界，相当于无类域间路由（Classless InterDomain Routing，CIDR）的方式，并且 DNS PTR 表示（“半字节” 格式）使反向区的设置更为简单。

接口标识符必须在局部连接上是唯一的，并且通常由 IPv6 实现自动生成，虽然在需要时，通常可以覆盖缺省设置。一个典型的 IPv6 地址可能是这样的：2001:db8:201:9:a00:20ff:fe81:2b32

IPv6 地址规范通常包含一大串 0，所以架构师引入了指示这个的缩写方式。一对冒号（:: ）指示可能出现的最长 0 串，其在地址中仅能出现一次。

第 10.2 节　参考书目（和建议读物）

10.2.1　请求注释（RFC）

BIND 9 力求严格遵守 IETF 标准。据我们所知，BIND 9 遵守以下 RFC，并在以下编号的注释中列出了注意事项和例外情况。这些 RFC 中的许多是由 ISC 的员工或前员工所撰写。这个列表是不完全的。

包括 DNS 的互联网协议族的规范文档是作为请求注释（RFC）系列的技术备忘的一部份被出版的。标准本身是由互联网工程任务组（IETF）和互联网专家任务组（IESG）所定义。RFC 可以在 https://datatracker.ietf.org/doc/ 在线观看。

这些 RFC 中的一些，虽然是 DNS 相关的，不是实现软件需要关心的。

第 10.3 节　互联网标准

RFC 1034 - P. Mockapetris. Domain Names —Concepts and Facilities. November 1987.

RFC 1035 - P. Mockapetris. Domain Names —Implementation and Specification. November 1987. [1] [2]

RFC 1123 - R. Braden. Requirements for Internet Hosts - Application and Support. October 1989.

RFC 3596 - S. Thomson, C. Huitema, V. Ksinant, and M. Souissi. DNS Extensions to Support IP Version 6. October 2003.

RFC 5011 - M. StJohns. Automated Updates of DNS Security (DNSSEC) Trust Anchors.

RFC 6891 - J. Damas, M. Graff, and P. Vixie. Extension Mechanisms for DNS (EDNS(0)). April 2013.

第 10.4 节　建议标准

RFC 1982 - R. Elz and R. Bush. Serial Number Arithmetic. August 1996.

RFC 1995 - M. Ohta. Incremental Zone Transfer in DNS. August 1996.

RFC 1996 - P. Vixie. A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY). August 1996.

RFC 2136 - P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). April 1997.

RFC 2163 - A. Allocchio. Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM). January 1998.

RFC 2181 - R. Elz and R. Bush. Clarifications to the DNS Specification. July 1997.

RFC 2308 - M. Andrews. Negative Caching of DNS Queries (DNS NCACHE). March 1998.

RFC 2539 - D. Eastlake, 3rd. Storage of Diffie-Hellman Keys in the Domain Name System (DNS). March 1999.

RFC 2782 - A. Gulbrandsen, P. Vixie, and L. Esibov. A DNS RR for Specifying the Location of Services (DNS SRV). February 2000.

RFC 2845 - P. Vixie, O. Gudmundsson, D. Eastlake, 3rd, and B. Wellington. Secret Key Transaction Authentication for DNS (TSIG). May 2000.

RFC 2930 - D. Eastlake, 3rd. Secret Key Establishment for DNS (TKEY RR). September 2000.

RFC 2931 - D. Eastlake, 3rd. DNS Request and Transaction Signatures (SIG(0)s). September 2000. [3]

RFC 3007 - B. Wellington. Secure Domain Name System (DNS) Dynamic Update. November 2000.

RFC 3110 - D. Eastlake, 3rd. RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS). May 2001.

RFC 3225 - D. Conrad. Indicating Resolver Support of DNSSEC. December 2001.

RFC 3226 - O. Gudmundsson. DNSSEC and IPv6 A6 Aware Server/Resolver Message Size Requirements. December 2001.

RFC 3492 - A. Costello. Punycode: A Bootstring Encoding of Unicode for Internationalized Domain Names in Applications (IDNA). March 2003.

RFC 3597 - A. Gustafsson. Handling of Unknown DNS Resource Record (RR) Types. September 2003.

RFC 3645 - S. Kwan, P. Garg, J. Gilroy, L. Esibov, J. Westhead, and R. Hall. Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG). October 2003.

RFC 4025 - M. Richardson. A Method for Storing IPsec Keying Material in DNS. March 2005.

RFC 4033 - R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. March 2005. [4]

RFC 4034 - R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. March 2005.

RFC 4035 - R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. March 2005.

RFC 4255 - J. Schlyter and W. Griffin. Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints. January 2006.

RFC 4343 - D. Eastlake, 3rd. Domain Name System (DNS) Case Insensitivity Clarification. January 2006.

RFC 4398 - S. Josefsson. Storing Certificates in the Domain Name System (DNS). March 2006.

RFC 4470 - S. Weiler and J. Ihren. Minimally Covering NSEC Records and DNSSEC On-line Signing. April 2006. [5]

RFC 4509 - W. Hardaker. Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs). May 2006.

RFC 4592 - E. Lewis. The Role of Wildcards in the Domain Name System. July 2006.

RFC 4635 - D. Eastlake, 3rd. HMAC SHA (Hashed Message Authentication Code, Secure Hash Algorithm) TSIG Algorithm Identifiers. August 2006.

RFC 4701 - M. Stapp, T. Lemon, and A. Gustafsson. A DNS Resource Record (RR) for Encoding Dynamic Host Configuration Protocol (DHCP) Information (DHCID RR). October 2006.

RFC 4955 - D. Blacka. DNS Security (DNSSEC) Experiments. July 2007. [6]

RFC 5001 - R. Austein. DNS Name Server Identifier (NSID) Option. August 2007.

RFC 5155 - B. Laurie, G. Sisson, R. Arends, and D. Blacka. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. March 2008.

RFC 5452 - A. Hubert and R. van Mook. Measures for Making DNS More Resilient Against Forged Answers. January 2009. [7]

RFC 5702 - J. Jansen. Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC. October 2009.

RFC 5936 - E. Lewis and A. Hoenes, Ed. DNS Zone Transfer Protocol (AXFR). June 2010.

RFC 5952 - S. Kawamura and M. Kawashima. A Recommendation for IPv6 Address Text Representation. August 2010.

RFC 6052 - C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and X. Li. IPv6 Addressing of IPv4/IPv6 Translators. October 2010.

RFC 6147 - M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum. DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. April 2011.

[8]

RFC 6594 - O. Sury. Use of the SHA-256 Algorithm with RSA, Digital Signature Algorithm (DSA), and Elliptic Curve DSA (ECDSA) in SSHFP Resource Records. April 2012.

RFC 6604 - D. Eastlake, 3rd. xNAME RCODE and Status Bits Clarification. April 2012.

RFC 6605 - P. Hoffman and W. C. A. Wijngaards. Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC. April 2012. [9]

RFC 6672 - S. Rose and W. Wijngaards. DNAME Redirection in the DNS. June 2012.

RFC 6698 - P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. August 2012.

RFC 6725 - S. Rose. DNS Security (DNSSEC) DNSKEY Algorithm IANA Registry Updates. August 2012. [10]

RFC 6840 - S. Weiler, Ed., and D. Blacka, Ed. Clarifications and Implementation Notes for DNS Security (DNSSEC). February 2013. [11]

RFC 7216 - M. Thomson and R. Bellis. Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS. April 2014.

RFC 7344 - W. Kumari, O. Gudmundsson, and G. Barwood. Automating DNSSEC Delegation Trust Maintenance. September 2014. [12]

RFC 7477 - W. Hardaker. Child-to-Parent Synchronization in DNS. March 2015.

RFC 7766 - J. Dickinson, S. Dickinson, R. Bellis, A. Mankin, and D. Wessels. DNS Transport over TCP - Implementation Requirements. March 2016.

RFC 7828 - P. Wouters, J. Abley, S. Dickinson, and R. Bellis. The edns-tcp-keepalive EDNS0 Option. April 2016.

RFC 7830 - A. Mayrhofer. The EDNS(0) Padding Option. May 2016. [13]

RFC 8080 - O. Sury and R. Edmonds. Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC. February 2017.

RFC 8482 - J. Abley, O. Gudmundsson, M. Majkowski, and E. Hunt. Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY. January 2019.

RFC 8490 - R. Bellis, S. Cheshire, J. Dickinson, S. Dickinson, T. Lemon, and T. Pusateri. DNS Stateful Operations. March 2019.

RFC 8624 - P. Wouters and O. Sury. Algorithm Implementation Requirements and Usage Guidance for DNSSEC. June 2019.

RFC 8749 - W. Mekking and D. Mahoney. Moving DNSSEC Lookaside Validation (DLV) to Historic Status. March 2020.

第 10.5 节　信息参考类 RFC

RFC 1535 - E. Gavron. A Security Problem and Proposed Correction With Widely Deployed DNS Software. October 1993.

RFC 1536 - A. Kumar, J. Postel, C. Neuman, P. Danzig, and S. Miller. Common DNS Implementation Errors and Suggested Fixes. October 1993.

RFC 1591 - J. Postel. Domain Name System Structure and Delegation. March 1994.

RFC 1706 - B. Manning and R. Colella. DNS NSAP Resource Records. October 1994.

RFC 1713 - A. Romao. Tools for DNS Debugging. November 1994.

RFC 1794 - T. Brisco. DNS Support for Load Balancing. April 1995.

RFC 1912 - D. Barr. Common DNS Operational and Configuration Errors. February 1996.

RFC 2230 - R. Atkinson. Key Exchange Delegation Record for the DNS. November 1997.

RFC 2352 - O. Vaughan. A Convention for Using Legal Names as Domain Names. May 1998.

RFC 2825 - IAB and L. Daigle. A Tangled Web: Issues of I18N, Domain Names, and the Other Internet Protocols. May 2000.

RFC 2826 - Internet Architecture Board. IAB Technical Comment on the Unique DNS Root. May 2000.

RFC 3071 - J. Klensin. Reflections on the DNS, RFC 1591, and Categories of Domains. February 2001.

RFC 3258 - T. Hardie. Distributing Authoritative Name Servers via Shared Unicast Addresses. April 2002.

RFC 3363 - R. Bush, A. Durand, B. Fink, O. Gudmundsson, and T. Hain. Representing Internet Protocol Version 6 (IPv6) Addresses in the Domain Name System (DNS). August 2002. [14]

RFC 3493 - R. Gilligan, S. Thomson, J. Bound, J. McCann, and W. Stevens. Basic Socket Interface Extensions for IPv6. March 2003.

RFC 3496 - A. G. Malis and T. Hsiao. Protocol Extension for Support of Asynchronous Transfer Mode (ATM) Service Class-aware Multiprotocol Label Switching (MPLS) Traffic Engineering. March 2003.

RFC 3833 - D. Atkins and R. Austein. Threat Analysis of the Domain Name System (DNS). August 2004.

RFC 4074 - Y. Morishita and T. Jinmei. Common Misbehavior Against DNS Queries for IPv6 Addresses. June 2005.

RFC 4892 - S. Woolf and D. Conrad. Requirements for a Mechanism Identifying a Name Server Instance. June 2007.

RFC 6781 - O. Kolkman, W. Mekking, and R. Gieben. DNSSEC Operational Practices, Version 2. December 2012.

RFC 7043 - J. Abley. Resource Records for EUI-48 and EUI-64 Addresses in the DNS. October 2013.

RFC 7129 - R. Gieben and W. Mekking. Authenticated Denial of Existence in the DNS. February 2014.

RFC 7553 - P. Faltstrom and O. Kolkman. The Uniform Resource Identifier (URI) DNS Resource Record. June 2015.

RFC 7583 - S. Morris, J. Ihren, J. Dickinson, and W. Mekking. DNSSEC Key Rollover Timing Considerations. October 2015.

第 10.6 节　试验性 RFC

RFC 1183 - C. F. Everhart, L. A. Mamakos, R. Ullmann, P. Mockapetris. New DNS RR Definitions. October 1990.

RFC 1464 - R. Rosenbaum. Using the Domain Name System to Store Arbitrary String Attributes. May 1993.

RFC 1712 - C. Farrell, M. Schulze, S. Pleitner, and D. Baldoni. DNS Encoding of Geographical Location. November 1994.

RFC 1876 - C. Davis, P. Vixie, T. Goodwin, and I. Dickinson. A Means for Expressing Location Information in the Domain Name System. January 1996.

RFC 2345 - J. Klensin, T. Wolf, and G. Oglesby. Domain Names and Company Name Retrieval. May 1998.

RFC 2540 - D. Eastlake, 3rd. Detached Domain Name System (DNS) Information. March 1999.

RFC 3123 - P. Koch. A DNS RR Type for Lists of Address Prefixes (APL RR). June 2001.

RFC 6742 - RJ Atkinson, SN Bhatti, U. St. Andrews, and S. Rose. DNS Resource Records for the Identifier-Locator Network Protocol (ILNP). November 2012.

RFC 7314 - M. Andrews. Extension Mechanisms for DNS (EDNS) EXPIRE Option. July 2014.

RFC 7929 - P. Wouters. DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP. August 2016.

第 10.7 节　当前最佳实践 RFC

RFC 2219 - M. Hamilton and R. Wright. Use of DNS Aliases for Network Services. October 1997.

RFC 2317 - H. Eidnes, G. de Groot, and P. Vixie. Classless IN-ADDR.ARPA Delegation. March 1998.

RFC 2606 - D. Eastlake, 3rd and A. Panitz. Reserved Top Level DNS Names. June 1999. [15]

RFC 3901 - A. Durand and J. Ihren. DNS IPv6 Transport Operational Guidelines. September 2004.

RFC 5625 - R. Bellis. DNS Proxy Implementation Guidelines. August 2009.

RFC 6303 - M. Andrews. Locally Served DNS Zones. July 2011.

RFC 7793 - M. Andrews. Adding 100.64.0.0/10 Prefixes to the IPv4 Locally-Served DNS Zones Registry. May 2016.

RFC 8906 - M. Andrews and R. Bellis. A Common Operational Problem in DNS Servers: Failure to Communicate. September 2020.

第 10.8 节　已成历史的 RFC

RFC 2874 - M. Crawford and C. Huitema. DNS Extensions to Support IPv6 Address Aggregation and Renumbering. July 2000. [4]

RFC 4431 - M. Andrews and S. Weiler. The DNSSEC Lookaside Validation (DLV) DNS Resource Record. February 2006.

第 10.9 节　关于“未知”类型的 RFC

RFC 1033 - M. Lottor. Domain Administrators Operations Guide. November 1987.

RFC 1101 - P. Mockapetris. DNS Encoding of Network Names and Other Types. April 1989.

第 10.10 节　已废弃并且未实现的试验性 RFC

RFC 974 - C. Partridge. Mail Routing and the Domain System. January 1986.

RFC 1521 - N. Borenstein and N. Freed. MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. September 1993 [16]

RFC 1537 - P. Beertema. Common DNS Data File Configuration Errors. October 1993.

RFC 1750 - D. Eastlake, 3rd, S. Crocker, and J. Schiller. Randomness Recommendations for Security. December 1994.

RFC 2010 - B. Manning and P. Vixie. Operational Criteria for Root Name Servers. October 1996.

RFC 2052 - A. Gulbrandsen and P. Vixie. A DNS RR for Specifying the Location of Services. October 1996.

RFC 2065 - D. Eastlake, 3rd and C. Kaufman. Domain Name System Security Extensions. January 1997.

RFC 2137 - D. Eastlake, 3rd. Secure Domain Name System Dynamic Update. April 1997.

RFC 2168 - R. Daniel and M. Mealling. Resolution of Uniform Resource Identifiers Using the Domain Name System. June 1997.

RFC 2240 - O. Vaughan. A Legal Basis for Domain Name Allocation. November 1997.

RFC 2535 - D. Eastlake, 3rd. Domain Name System Security Extensions. March 1999. [17] [18]

RFC 2537 - D. Eastlake, 3rd. RSA/MD5 KEYs and SIGs in the Domain Name System (DNS). March 1999.

RFC 2538 - D. Eastlake, 3rd and O. Gudmundsson. Storing Certificates in the Domain Name System (DNS). March 1999.

RFC 2671 - P. Vixie. Extension Mechanisms for DNS (EDNS0). August 1999.

RFC 2672 - M. Crawford. Non-Terminal DNS Name Redirection. August 1999.

RFC 2673 - M. Crawford. Binary Labels in the Domain Name System. August 1999.

RFC 2915 - M. Mealling and R. Daniel. The Naming Authority Pointer (NAPTR) DNS Resource Record. September 2000.

RFC 2929 - D. Eastlake, 3rd, E. Brunner-Williams, and B. Manning. Domain Name System (DNS) IANA Considerations. September 2000.

RFC 3008 - B. Wellington. Domain Name System Security (DNSSEC) Signing Authority. November 2000.

RFC 3090 - E. Lewis. DNS Security Extension Clarification on Zone Status. March 2001.

RFC 3152 - R. Bush. Delegation of IP6.ARPA. August 2001.

RFC 3445 - D. Massey and S. Rose. Limiting the Scope of the KEY Resource Record (RR). December 2002.

RFC 3490 - P. Faltstrom, P. Hoffman, and A. Costello. Internationalizing Domain Names in Applications (IDNA). March 2003. [19]

RFC 3491 - P. Hoffman and M. Blanchet. Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN). March 2003. [19]

RFC 3655 - B. Wellington and O. Gudmundsson. Redefinition of DNS Authenticated Data (AD) Bit. November 2003.

RFC 3658 - O. Gudmundsson. Delegation Signer (DS) Resource Record (RR). December 2003.

RFC 3755 - S. Weiler. Legacy Resolver Compatibility for Delegation Signer (DS). May 2004.

RFC 3757 - O. Kolkman, J. Schlyter, and E. Lewis. Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag. May 2004.

RFC 3845 - J. Schlyter. DNS Security (DNSSEC) NextSECure (NSEC) RDATA Format. August 2004.

RFC 4294 - J. Loughney, Ed. IPv6 Node Requirements. [20]

RFC 4408 - M. Wong and W. Schlitt. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. April 2006.

RFC 5966 - R. Bellis. DNS Transport Over TCP - Implementation Requirements. August 2010.

RFC 6844 - P. Hallam-Baker and R. Stradling. DNS Certification Authority Authorization (CAA) Resource Record. January 2013.

RFC 6944 - S. Rose. Applicability Statement: DNS Security (DNSSEC) DNSKEY Algorithm Implementation Status. April 2013.

第 10.11 节　BIND 9 中不再支持的 RFC

RFC 2536 - D. Eastlake, 3rd. DSA KEYs and SIGs in the Domain Name System (DNS). March 1999.

10.11.1　注释

[1] Queries to zones that have failed to load return SERVFAIL rather than a non-authoritative response. This is considered a feature.

[2] CLASS ANY queries are not supported. This is considered a feature.

[3] When receiving a query signed with a SIG(0), the server is only able to verify the signature if it has the key in its local authoritative data; it cannot do recursion or validation to retrieve unknown keys.

[4] Compliance is with loading and serving of A6 records only. A6 records were moved to the experimental category by RFC 3363.

[5] Minimally covering NSEC records are accepted but not generated.

[6] BIND 9 interoperates with correctly designed experiments.

[7] named only uses ports to extend the ID space; addresses are not used.

[8] Section 5.5 does not match reality. named uses the presence of DO=1 to detect if validation may be occurring. CD has no bearing on whether validation occurs.

[9] Compliance is conditional on the OpenSSL library being linked against a supporting ECDSA.

[10] RSAMD5 support has been removed. See RFC 6944.

[11] Section 5.9 - Always set CD=1 on queries. This is not done, as it prevents DNSSEC from working correctly through another recursive server.

When talking to a recursive server, the best algorithm is to send CD=0 and then send CD=1 iff SERVFAIL is returned, in case the recursive server has a bad clock and/or bad trust anchor. Alternatively, one can send CD=1 then CD=0 on validation failure, in case the recursive server is under attack or there is stale/bogus authoritative data.

[12] Updating of parent zones is not yet implemented.

[13] named does not currently encrypt DNS requests, so the PAD option is accepted but not returned in responses.

[14] Section 4 is ignored.

[15] This does not apply to DNS server implementations.

[16] Only the Base 64 encoding specification is supported.

[17] Wildcard records are not supported in DNSSEC secure zones.

[18] Servers authoritative for secure zones being resolved by BIND 9 must support EDNS0 (RFC2671), and must return all relevant SIGs and NXTs in responses, rather than relying on the resolving server to perform separate queries for missing SIGs and NXTs.

[19] BIND 9 requires --with-idn to enable entry of IDN labels within dig, host, and nslookup at compile time. ACE labels are supported everywhere with or without --with-idn.

[20] Section 5.1 - DNAME records are fully supported.

10.11.2 互联网草案

互联网草案（ID）是互联网工程任务组（IETF）粗略草案的工作文档。它们本质上是 RFC 前期的开发阶段。实现者要小心不要将 ID 当成归档的标准，并且不要在正式文档中引用它们，除非附带

有它们处于“工作进程中”的免责声明。ID 具有六个月的生命，之后它们将被删除，除非被作者所更新。

10.11.3 其它关于 BIND 的文档

Paul Albitz and Cricket Liu. DNS 和 BIND. Copyright 1998 Sebastopol, CA: O’Reilly and Associates.

第十一章　手册页

第 11.1 节　rndc.conf - rndc 的配置文件

11.1.1　概要

rndc.conf

11.1.2　描述

rndc.conf 是 BIND 9 名字服务器控制工具 rndc 的配置文件。这个文件有与 named.conf 相似的结构和语法。语句包含在花括号之内，并以分号结束。语句中的子句也以分号结束。所支持的通常的注释风格为：

C 风格：/* */

C++ 风格：// 到行尾

Unix 风格：# 到行尾

rndc.conf 比 named.conf 简短得多。文件使用三个语句：一个 options 语句，一个 server 语句和一个 key 语句。

options 语句包含五个子句。default-server 子句后跟名字或者一个名字服务器的地址。在没有为 rndc 提供名字服务器参数时，将使用这个主机。default-key 子句后跟一个密钥名，这个密钥由一个 key 语句标识。如果在 rndc 命令行没有提供 keyid ，并且在一个匹配的 server 语句中没有找到 key 子句，就使用这个缺省的密钥来认证服务器的命令和响应。default-port 子句后跟要连接到的远程名字服务器的端口。如果在 rndc 命令行没有提供 port 选项，并且在一个匹配的 server 语句中没有找到 port 子句，就使用这个缺省的端口来连接。default-source-address 和 default-source-address-v6 子句可以分别用来设置 IPv4 和 IPv6 的源地址。

在 server 关键字之后，server 语句包含一个字符串，代表一个名字服务器的主机名或地址。这个语句有三个可能的子句：key ， port 和 addresses 。密钥名必须匹配文件中一个 key 语句的名字。端

口号指定要连接到的端口。如果提供了一个 addresses 子句，将使用这些地址来代替服务器名字。每个地址可以带有一个可选的端口。如果使用了 source-address 或者 source-address-v6 ，这些会分别用于指定 IPv4 和 IPv6 的源地址。

key 语句以一个标识密钥名字的字符串开始。这个语句有两个子句。algorithm 定义 rndc 用到的认证算法；当前仅支持 HMAC-MD5（为了兼容), HMAC-SHA1, HMAC-SHA224, HMAC-SHA256（缺省），HMAC-SHA384 和 HMAC-SHA512。它后跟一个 secret 子句，后者包含 base-64 编码的算法的认证密钥。这个 base-64 字符串使用双引号引起来。

有两个通常的方式来生成密钥的 base-64 字符串。BIND 9 程序 rndc-confgen 可以用来生成一个随机密钥，或者 mmencode 程序，也叫做 mimencode ，可以用来生成一个已知输入的 base-64 字符串。mmencode 不随 BIND 9 提供，但是它在许多系统上是可用的。关于每个命令行的样例，参见例子部份。

11.1.3 例子

```
options {
 default-server  localhost;
 default-key    samplekey;
};
```

```
server localhost {
 key          samplekey;
};
```

```
server testserver {
 key    testkey;
 addresses   { localhost port 5353; };
};
```

```
key samplekey {
 algorithm     hmac-sha256;
 secret       "6FMfj43Osz4lyb24OIe2iGEz9lf1llJO+lz";
};
```

```
key testkey {
 algorithm   hmac-sha256;
```

（下页继续）

(续上页)

```
 secret    "R3HI8P6BKw9ZwXwN3VZKuQ==";
};
```

在上面的例子中，rndc 缺省将使用 localhost(127.0.0.1) 作为服务器，和名为 samplekey 的密钥。到服务器 localhost 的命令将使用密钥 samplekey，后者也必须使用同样的名字和密钥定义在服务器的配置文件中。key 语句指明 samplekey 使用 HAMC-SHA256 算法，它的 secret 子句包含这个 HMAC-SHA256 密钥的 base-64 编码，并被包括在双引号中。

如果使用 rndc -s testserver , rndc 将会连接到服务器 localhost 的 5353 端口 , 并使用密钥 testkey。

使用 rndc-confgen 生成一个随机密钥：

rndc-confgen

一个完整的 rndc.conf 文件，包含随机生成的密钥，将会被写到标准输出。还会打印出为 named.conf 提供的被注释掉的 key 和 controls 语句。

使用 mmencode 生成一个 base-64 密钥：

echo "known plaintext for a secret" | mmencode

11.1.4 名字服务器配置

名字服务器必须被配置成接受 rndc 连接和识别 rndc.conf 文件中所指定的密钥 , 这通过在 named.conf 中的 controls 语句来实现。详细情况参见 BIND 9 管理员参考手册中的 controls 语句部份。

11.1.5 参见

rndc(8), rndc-confgen(8), mmencode(1), BIND 9 管理员参考手册。

第 11.2 节 rndc - 名字服务器控制工具

11.2.1 概要

rndc [-b source-address] [-c config-file] [-k key-file] [-s server] [-p port] [-q] [-r] [-V] [-y key_id] [[-4] | [-6]] {command}

11.2.2 描述

rndc 控制一个名字服务器的运行。它替代了旧 BIND 发布版本所提供的 ndc 工具。如果 rndc 在没有命令行选项或参数时被调用，它将打印出其所支持的命令和可用选项和参数的简短总结。

rndc 通过一个 TCP 连接与名字服务器通信，发送经过数字签名认证的命令。在当前版本的 rndc 和 named 中，只支持的认证算法是 HMAC-MD5（为了兼容），HMAC-SHA1，HMAC-SHA224，HMAC-SHA256（缺省），HMAC-SHA384 和 HMAC-SHA512。它们在连接的两端使用共享密钥。它为命令请求和名字服务器的响应提供 TSIG 类型的认证。所有经由通道发送的命令都必须被一个服务器所知道的 key_id 签名。

rndc 读一个配置文件来决定如何联系名字服务器并决定使用哪一个算法和密钥。

11.2.3 选项

-4 只使用 IPv4。

-6 只使用 IPv6。

-b source-address 使用 source-address 作为连接服务器的源地址。允许多个实例设置其 IPv4 和 IPv6 源地址。

-c config-file 使用 config-file 作为缺省的配置文件 /etc/rndc.conf 的替代。

-k key-file 使用 key-file 作为缺省的密钥文件 /etc/rndc.key 的替代。如果 config-file 不存在，/etc/rndc.key 中的密钥将用于认证发向服务器的命令。

-s server server 是与 rndc 的配置文件中 server 语句相匹配的服务器的名字或地址。如果命令行没有提供服务器，会使用 rndc 配置文件中 options 语句中的 default-server 子句所命名的主机。

-p port 发送命令到 TCP 端口 port，以取代 BIND 9 的缺省控制通道端口，953。

-q 安静模式：从服务器返回的消息文本将不被打印，除非存在错误。

-r 指示 rndc 打印出 named 在执行了请求的命令之后的返回码（例如，ISC_R_SUCCESS，ISC_R_SUCCESS 等等）。

-V 打开详细日志。

-y key_id 使用配置文件中的密钥 key_id。key_id 必须被 named 所知道，带有同样的算法和密钥字符串，以便成功通过控制消息的验证。如果没有指定 key_id，rndc 将首先在所用的服务器的 server 语句中查找 key 子句，或者如果没有为主机提供 server 语句，就查找 options

语句中的 default-key 子句。注意配置文件中包含有用于发送认证控制命令到名字服务器的共享密钥。因此它不应该具有通常的读或写的权限。

11.2.4 命令

rndc 所支持的命令列表，可以通过不带任何参数运行 rndc 来查看。

当前支持的命令是：

addzone zone [class [view]] configuration 在服务器运行时增加一个区。这个命令要求 allow-new-zones 选项被设置为 yes 。在命令行所指定的 configuration 字符串就是通常被放在 named.conf(5) 中的区配置文本。

配置被保存在名为 viewname.nzf 的文件中（或者，如果 named(8) 编译时带有 liblmdb，就保存到一个名为 viewname.nzd 的 LMDB 数据库文件中）。viewname 视图的名字，除非视图名字中包含有不兼容用于文件名的字符，这种情况下就使用视图名字的加密哈希之后的字符串替代。当 named(8) 被重启时，这个文件将被转载到视图的配置中，这样被添加的区将会在重启后能够持续。

这个例子 addzone 命令将会添加区 example.com 到缺省视图：

$rndc addzone example.com '{ type master; file "example.com.db"; };'

（注意围绕区配置文本的花括号和分号。）

参见 rndc delzone 和 rndc modzone 。

delzone [-clean] zone [class [view]] 在服务器运行时删除一个区。

如果指定了 -clean 参数，区的主文件（和日志文件，如果有的话）将会随着区一块被删除。没有 -clean 选项时，区文件必须手动清除。（如果区是" slave" 或" stub" 类型时，rndc delzone 命令的输出将报告需要清理的文件。）

如果区最初是通过 rndc addzone 增加，它就会被永久地删除。然而，如果它最初是在 named.conf 中配置，则最初的配置仍然存在；当服务器重启或重新读入配置时，区就会恢复回来。要永久地删除，必须从 named.conf 中删除。

参见 rndc addzone 和 rndc modzone 。

dnssec (-status | -rollover -key id [-alg algorithm] [-when time] | -checkds [-key id [-alg algorithm]] [-whe
这个命令允许你交互操作一个给定区的 "dnssec-policy"。

rndc dnssec -status 显示指定区的 DNSSEC 签名状态。

rndc dnssec -rollover 允许你为一个特定的密钥调度一次密钥轮转（覆盖原来的密钥寿命）。

rndc dnssec -checkds 将使 named 知道给定密钥的 DS 记录已能被看到发布在父区或从父区撤回。这是为了完成 KSK 轮转所必需的。如果指定了 -key id 参数，就查找带有给定标识符的密钥，否则如果区中仅有一个密钥充当 KSK，就假设为这个密钥的 DS 记录（如果有多个密钥带有同样的标记，使用 -alg algorithm 来选择正确的算法）。DS 记录被发布或撤回的时间被设置为现在，除非使用 -when time 参数另行指定了。

dnstap (-reopen | -roll [number]) 关闭和重新打开 DNSTAP 输出文件。rndc dnstap -reopen 允许输出文件被改名，这样 named(8) 可以截断并重新打开它。rndc dnstap -roll 使输出文件自动轮转，类似于日志文件；最近的输出文件在其名字后添加 ".0"；更早的最近输出文件被移动为 ".1"，诸如此类。如果指定了 number，备份日志文件的个数被限制为这个数。

dumpdb [-all | -cache | -zones | -adb | -bad | -expired | -fail] [view …] 转储服务器指定视图的缓存（缺省情况）和/或区到转储文件中。如果未指定视图就转储所有视图。（参见 BIND 9 管理员参考手册中的 dump-file 选项。）

flush 刷新服务器的缓存。

flushname name [view] 从视图的 DNS 缓存，如果合适，和从视图的名字服务器地址库，不存在缓存和 SERVFAIL 缓存中刷新给定的名字。

flushtree name [view] 从视图的 DNS 缓存，地址库，不存在缓存和 SERVFAIL 缓存中刷新给定的名字及其所有子域。

freeze [zone [class [view]]] 冻结对一个动态更新区的更新。如果没有指定区，就冻结对所有区的更新。这就允许对一个动态更新方式正常更新的区进行手工编辑。它也会导致日志文件中的变化被同步到主区文件。在区被冻结时，所有的动态更新尝试都会被拒绝。

参见 rndc thaw 。

halt [-p] 立即停止服务器。所有由动态更新或 IXFR 所作的最新改变没有被存到区文件中，但是在服务器重新启动时，将从日志文件中向前滚动。如果指定了 -p ，将返回 named(8) 的进程号。这可以让一个外部进程来检查 named(8) 是否完全被中断。

参见 rndc stop 。

loadkeys [zone [class [view]]] 从密钥目录取给定区的所有 DNSSEC 密钥。如果它们在其发布期内，将它们合并到区的 DNSKEY 资源记录集中。然而，与 rndc sign 不同，不会立即使用新密钥重签区，但是允许随时间推移进行增量重签。

这个命令要求使用 dnssec-policy 配置区，或者 auto-dnssec 区选项被设置为 maintain ，而且还要求区被配置为允许动态 DNS。（更详细情况参见管理员参考手册中的 "动态更新策略"。）

managed-keys (status | refresh | sync | destroy) [class [view]] 检查和控制用于处理 RFC 5011 DNSSEC 信任锚维护的 "被管理密钥" 数据库。如果指定一个视图，这些命令应用于这个视

图；否则就应用于所有视图。

- 在使用 status 关键字运行时，打印被管理密钥数据库的当前状态。
- 在使用 refresh 关键字运行时，强制发送一个针对所有被管理密钥的立即刷新请求，如果发现任何新的密钥，就更新被管理密钥数据库，而不等待通常的刷新间隔。
- 在使用 sync 关键字运行时，强制进行一个立即的转储被管理密钥数据库到磁盘（到文件 managed-keys.bind 或者 viewname.mkeys ）。这个对数据库的同步使用它的日志文件，这样数据库的当前内容可以可视化地检查。
- 在使用 destroy 关键字运行时，被管理密钥数据库被关闭和删除，所有密钥维护都被终止。这个命令只能在超级谨慎的情况下使用。

  当前存在的已经受信任的密钥不会从内存中删除；使用这条命令后 DNSSEC 验证可以继续进行。但是，密钥维护操作将会停止直到 named(8) 重启或者重读配置，并且所有已存在的密钥维护状态都会被删除。

  在这条命令后立即运行 rndc reconfig 或重启 named(8) 将会导致密钥维护从头开始初始化，就像服务器第一次启动时一样。这主要用于测试，但是也可以用于，例如，在发生信任锚轮转时开始获取新密钥，或者作为密钥维护问题的强力修复。

modzone zone [class [view]] configuration 在服务器运行时修改一个区的配置。这个命令要求 allow-new-zones 选项被设置为 yes 。与 addzone 一起使用时，命令行中指定的 configuration 字符串就是原本应该放在 named.conf 中的区配置文本。

如果区最初通过 rndc addzone 添加，配置变化将被永久记录，并在服务器重启或重新读入配置之后仍然有效。然而，如果它最初在 named.conf 中配置，最初的配置仍然保持在那里；当服务器重启或重新读入配置后，区将会恢复到其初始配置。为是变化永久化，必须也在 named.conf 中修改。

参见 rndc addzone 和 rndc delzone 。

notify zone [class [view]] 重新发出区的 NOTIFY 消息。

notrace 将服务器的调试级别设置为 0。

参见 rndc trace 。

nta [(-class class | -dump | -force | -remove | -lifetime duration)] domain [view] 为 domain 设置一个 DNSSEC 不存在信任锚（NTA），带有一个 duration 的生存时间。缺省的生存时间是通过 nta-lifetime 选项配置在 named.conf 中的，缺省是一小时。生存时间不能超过一周。

一个不存在信任锚选择性地关闭那些由于错误配置而不是攻击而明知会失败的区的 DNSSEC 验证。当被验证的数据处于或低于一个活跃的 NTA（并且在任何其它被配置的信任锚之上），

named(8) 将会终止 DNSSEC 验证过程并将数据当成不安全的而不是作为伪造的。这个过程会持续到 NTA 的生命周期结束。

NTA 持久化能够跨越 named(8) 服务器重启。一个视图的 NTA 被保存在一个名为 name.nta 的文件中，其中的 name 是视图的名字，或者当视图名中含有不能用于文件名的字符时，是根据视图名生成的加密哈希。

一个现存的 NTA 可以通过使用 -remove 选项删除。

一个 NTA 的生命周期可以使用 -lifetime 选项指定。TTL 风格的后缀可以用于指定生命周期，以秒，分或小时的格式。如果指定的 NTA 已经存在，它的生命周期会被更新为新的值。将 lifetime 设置为零等效于设置为 -remove 。

如果使用了 -dump ，任何其它参数都被忽略，打印出现存 NTA 的列表（注意这会包含已经过期但还未被清理的 NTA）。

通常，named(8) 会周期性测试以检查一个 NTA 之下的数据现在是否可以被验证（参考管理员参考手册中的 nta-recheck 选项获取详细信息）。如果数据可以被验证，这个 NTA 就被认为不再需要，允许提前过期。-force 覆盖这个特性并强制一个 NTA 持久到其完整的生命周期，不考虑在 NTA 不存在时数据是否可以被验证。

视图类可以使用 -class 指定。缺省是 IN 类，这是唯一支持 DNSSEC 的类。

所有这些选项都可以被简化，如，简化成 -l ， -r ， -d ， -f 和 -c 。

不能识别的选项被当做错误对待。要引用一个以连字符开始的域名或视图名，在命令行使用双连字符指示选项的结束。

querylog [(on | off)] 打开或关闭请求日志。（为向后兼容，可以不带参数使用这个命令，即请求日志在开和关之间切换。

请求日志也可以显式打开，通过在 named.conf 的 logging 部份指定 queries category 到一个 channel ，或者在 named.conf 的 options 部份指定 querylog yes; 。

reconfig 重新载入配置文件和新的区，但是不载入已经存在的区文件，即使其已经被修改过。这在有大量区的时候可以比完全的 reload 更快，因为它避免了去检查区文件的修改时间。

recursing 转储 named(8) 当前为其提供递归服务的请求列表，以及当前迭代请求所发向的域名列表。（第二个列表包含对给定域名的当前活跃获取的个数，以及由于 fetches-per-zone 选项而被传递或丢掉个数。）。

refresh zone [class [view]] 对指定的区进行区维护。

reload 重新载入配置文件和区文件。

reload zone [class [view]] 重新载入指定的区文件。

retransfer zone [class [view]] 重新从主服务器传送指定的区文件。

如果使用 inline-signing 配置区，区的签名版本将被丢弃；在重新传送非签名版本完成后，将使用所有新签名重新生成签名版本。

scan 扫描可用网络接口列表以查看变化，不执行完全的 reconfig，也不等待 interface-interval 计时器。

secroots [-] [view …] 为指定视图转储安全根（即，通过 trust-anchors 语句，或 managed-keys 或 trusted-keys 语句（这两个都被废弃了），或 dnssec-validation auto 配置的信任锚）和否定信任锚。如果没有指定视图，就转储所有视图。安全根指示它们是否配置成受信任密钥，被管理密钥，或者正在初始化的被管理密钥（还未被一个成功的密钥刷新请求更新的被管理密钥）。

如果第一个参数是 "-"，通过 rndc 响应通道返回输出，并输出到标准输出。否则，将返回写到安全根转储文件，缺省是 named.secroots，但可以在 named.conf 中通过 secroots-file 选项覆盖。

参见 rndc managed-keys 。

serve-stale (on | off | reset | status) [class [view]] 打开，关闭，重置或报告配置在 named.conf 中的旧答复服务的当前状态。

如果旧答复服务被 rndc-serve-stale off 关闭，那么，即使 named(8) 重新加载或重新配置，它仍然会关闭。rndc serve-stale reset 恢复 named.conf 中的配置。

rndc serve-stale status 将报告旧答复服务当前是被配置打开或关闭，或者被 rndc 关闭。它也会报告 stale-answer-ttl 和 max-stale-ttl 的值。

showzone zone [class [view]] 输出一个运行区的配置。

参见 rndc zonestatus 。

sign zone [class [view]] 从密钥目录取给定区的所有 DNSSEC 密钥（参见 BIND 9 管理员参考手册中的 key-directory），如果它们在其发布期内，将它们合并到区的 DNSKEY 资源记录集中。如果 DNSKEY 资源记录集发生了变化，就自动使用新的密钥集合对区重新签名。

这个命令要求使用 dnssec-policy 配置区，或者 auto-dnssec 区选项被设置为 allow 或 maintain，还要求区被配置为允许动态更新。（更详细情况参见管理员参考手册中的 "动态更新策略"。）

参见 rndc loadkeys 。

signing [(-list | -clear keyid/algorithm | -clear all | -nsec3param (parameters | none) | -serial value) zone 列出，编辑或删除指定区的 DNSSEC 签名状态记录。正在进行的 DNSSEC 操作（如签名或

生成 NSEC3 链）的状态以 DNS 资源记录类型 sig-signing-type 的形式存放在区中。rndc signing -list 转换这些记录成为人可读的格式，指明哪个密钥是当前签名所用，哪个已完成对区的签名，哪个 NSEC3 链被创建和删除。

rndc signing -clear 可以删除单一的一个密钥（以 rndc signing -list 用来显示密钥的同一格式所指定的），或所有密钥。在这两种情况下，只有完成的密钥才能被删除；任何记录指明，一个没有完成签名的密钥将会被保留。

rndc signing -nsec3param 为一个区设置 NSEC3 参数。这只是在与 inline-signing 区一起使用 NSEC3 时才有的支持机制。参数以与 NSEC3PARAM 资源记录同样的格式指定：hash 算法，flags，iterations 和 salt，按上述顺序。

当前，hash 算法唯一定义的值为 1，表示 SHA-1。flags 可以被设置为 0 或 1，取决与你是否希望设置 NSEC3 链中的 opt-out 位。iterations 定义额外次数的数字，它应用于生成 NSEC3 哈希的算法中。salt 是一个表示成十六机制数的一串数据，一个连字符（‘-’）表示不使用 salt，或者关键字 auto，它使 named(8) 生成一个随机 64 位 salt。

例如，要创建一个 NSEC3 链，使用 SHA-1 哈希算法，没有 opt-out 标志，10 次循环，以及一个值为“FFFF”的 salt，使用：rndc signing -nsec3param 1 0 10 FFFF zone 。要设置 opt-out 标志，15 次循环，没有 salt，使用：rndc signing -nsec3param 1 1 15 - zone 。

rndc signing -nsec3param none 删除一个现存的 NSEC3 链并使用 NSEC 替代它。

rndc signing -serial value 设置区的序列号为指定值。如果这个值将会使序列号后退，它将被拒绝。主要用途是在联机签名区中设置序列号。

stats 写服务器的统计信息到统计文件。（参见 BIND 9 管理员参考手册中的 statistics-file 选项。）

status 显示服务器的状态。注意，区数目包括内部的 bind/CH 区，如果没有显式配置根区还包括缺省的 ./IN 区。

stop -p 停止服务器，在之前先确保所有通过动态更新或 IXFR 所作的最新修改第一时间被存入被修改区的区文件中。如果指定了 -p，将返回 named(8) 的进程号。这可以让一个外部进程来检查 named(8) 是否完全被停止。

参见 rndc halt 。

sync -clean [zone [class [view]]] 将一个动态区中日志文件的变化部分同步到其区文件。如果指定了“-clean”选项，会将日志文件删除。如果未指定区，将同步所有区。

tcp-timeouts [initial idle keepalive advertised] 当不使用参数调用时，显示 tcp-initial-timeout，tcp-idle-timeout，tcp-keepalive-timeout 和 tcp-advertised-timeout 选项的当前值。当使用参数调用时，更新这些值。这允许一位管理员在面临一次拒绝服务攻击时能够快速调整。参见 BIND 9 管理员参考手册中对这些选项的描述以获取关于它们用法的详细信息。

thaw [zone [class [view]]] 解冻一个被冻结的动态更新区。如果没有指定区，就解冻所有被冻结的区。它会导致服务器重新从磁盘载入区，并在载入完成后打开动态更新功能。在解冻一个区后。动态更新请求将不会被拒绝。如果区被修改并且使用了 ixfr-from-differences 选项，将修改日志文件以对应到区的变化。否则，如果区被修改，将会删除所有现存的日志文件。

参见 rndc freeze 。

trace 将服务器的调试级别增加 1。

trace level 将服务器的调试级别设置为指定的值。

参见 rndc notrace 。

tsig-delete keyname [view] 从服务器删除所给出的 TKEY 协商的密钥。（这个命令不会删除静态配置的 TSIG 密钥。）

tsig-list 列出当前被配置由 named(8) 所使用的每个视图中的全部 TSIG 密钥的名字。这个列表包含静态配置的密钥和动态 TKEY 协商的密钥。

validation (on | off | status) [view …]“ 打开，关闭 DNSSEC 验证或检查 DNSSEC 验证的状态。缺省时，验证时打开的。

当验证被打开或者关闭时刷新缓存，以避免使用不同状态下可能不同的数据。

zonestatus zone [class [view]] 显示给定区的当前状态，包含主文件名以及它加载时包含的所有文件，最近加载的时间，当前序列号，节点数目，区是否支持动态更新，区是否作了 DNSSEC 签名，它是否使用动态 DNSSEC 密钥管理或 inline 签名，以及区的预期刷新或过期时间。

参见 rndc showzone 。

指定区名的 rndc 命令，例如 reload ， retransfer 或 zonestatus ，在应用于类型 redirect 的区时可能会有歧义。重定向区总是被称为 “.”，可能与 hint 类型的区或者根区的辅拷贝混淆。要指定一个重定向区，使用特定的区名 -redirect ，不带结尾的点。（如果带有结尾的点，这就会指定一个名为 “-redirect” 的区。）

11.2.5 限制

当前没有在不使用配置文件的方式下提供共享密码 key_id 的方式。

几个错误消息可以被清除。

11.2.6 参见

rndc.conf(5), rndc-confgen(8), named(8), named.conf(5), ndc(8), BIND 9 管理员参考手册。

第 11.3 节　nsec3hash - 生成 NSEC3 哈希

11.3.1　概要

nsec3hash {salt} {algorithm} {iterations} {domain}

nsec3hash -r {algorithm} {flags} {iterations} {salt} {domain}

11.3.2　描述

nsec3hash 基于一个 NSEC3 参数集生成一个 NSEC3 哈希。这可以用于检查一个签名区中 NSEC3 记录的有效性。

如果这个命令以 nsec3hash -r 的方式运行，它按照一个 NSEC3 记录的头四个字段的顺序接受参数，后面跟着域名：算法，标志，循环次数，salt 值，域名。这就很方便地拷贝并粘贴一条 NSEC3 或者 NSEC3PARAM 记录的一部分到一条命令行中确保了一个 NSEC3 hash 的正确性。

11.3.3　参数

salt 提供给 hash 算法的 salt。

algorithm 一个表示 hash 算法的数字。当前对 NSEC3 唯一支持的哈希算法是 SHA-1，由数字 1 表示；因此，“1”是这个参数唯一合法的值。

flags 提供与 NSEC3 记录表示格式的兼容性，但由于标志不影响哈希而被忽略。

iterations 哈希应该额外执行的次数。

domain 被哈希的域名。

11.3.4　参见

BIND 9 管理员参考手册，RFC 5155.

第 11.4 节　dnstap-read - 以人可读的格式输出 dnstap 数据

11.4.1　概要

dnstap-read [-m] [-p] [-x] [-y] {file}

11.4.2 描述

dnstap-read 从一个指定文件中读入 dnstap 数据并以人可读的格式输出它。缺省时，dnstap 数据是以简短的概括格式输出，但是如果指定 -y 选项，就会是以一个较长并且更详细的 YAML 格式替代。

11.4.3 选项

-m 跟踪内存分配；用于调试内存泄漏。

-p 在输出 dnstap 数据之后，以文本格式输出封装在 dnstap 帧中的 DNS 消息。

-x 在打印 dnstap 数据之后，打印封装在 dnstap 帧中的 DNS 消息的线上格式的十六进制转码。

-y 以详细 YAML 格式输出 dnstap 数据。

11.4.4 参见

named(8), rndc(8), BIND 9 管理员参考手册。

第 11.5 节 named-nzd2nzf - 转换一个 NZD 数据库到 NZF 文本格式

11.5.1 概要

named-nzd2nzf {filename}

11.5.2 描述

named-nzd2nzf 转换一个 NZD 数据库到 NZF 格式并输出到标准输出。这可以用于复查由 rndc addzone 添加到 named 中的区的配置。也可用于在从一个新版本的 BIND 回滚到一个旧版本时恢复旧文件格式。

11.5.3 参数

filename 将要输出其内容的 .nzd 文件的名字。

11.5.4 参见

BIND 9 管理员参考手册。

第 11.6 节 named-journalprint - 以人可读的格式打印区日志文件

11.6.1 概要

named-journalprint {journal}

11.6.2 描述

named-journalprint 以人可读的格式打印一个区的日志文件的内容。

日志文件是在动态区有变化时（例如，通过 nsupdate ）由 named 自动创建的。它们以二进制格式记录了每一个资源记录的增加和删除，允许当服务器在宕机或崩溃之后的重启后将改变重新应用到区中。缺省时，日志文件的名字由相应区文件的名字加上扩展名 .jnl 组成。

named-journalprint 将一个给定日志文件转换为一个人可读的文本格式。每行都以 “add” 或 “del” 开头，以指明记录是否被增加或删除，并以主区文件格式连续放置资源记录。

11.6.3 参见

named(8), nsupdate(1), BIND 9 管理员参考手册。

第 11.7 节 mdig - DNS 流水线查找工具

11.7.1 概要

mdig {@server} [-f filename] [-h] [-v] [[-4] | [-6]] [-m] [-b address] [-p port#] [-c class] [-t type] [-i] [-x addr] [plusopt…]

mdig {-h}

mdig [@server] {global-opt…} { {local-opt…} {query} …}

11.7.2 描述

mdig 是一个多个/流水线式的 dig 查询版本：作为在发出每个请求之后等待一个响应的替代，它一开始就发出所有请求。响应是以收到的顺序被显示，而不是请求发出的顺序。

mdig 的选项是 dig 的选项的一个子集，被分成 “任意位置选项”，可以用于任意位置，“全局选项”，必须放在请求名之前（否则它们将被忽略并带有一条警告），和 “本地选项”，应用在命令行的下一个请求。

@server 选项是一个强制的全局选项。它是要请求的名字服务器的名字或 IP 地址。（与 dig 不同，这个值不是取自 /etc/resolv.conf 。）它可以是一个点分十进制格式的 IPv4 地址，也可以是一个冒号分隔格式的 IPv6 地址，或者是一个主机名。当所提供的 server 参数是一个主机名时，mdig 在请求这个名字服务器之前先解析这个名字。

mdig 提供了一些请求选项，它们影响了查询和显示结果的方式。其中一些设置或重置请求头部中的标志位，一些决定输出响应的哪些部份，其它的决定超时和重试策略。

每个请求选项由一个带一个前导加号（+ ）的关键字来标识。某些关键字设置或者重置一个选项。这些可以由字符串 no 来否定关键字的含义。其它关键字给选项赋值，例如超时间隔。他们具有如 +keyword=value 的形式。

11.7.3 任意位置选项

-f 选项使 mdig 运行在批处理模式，通过从文件 filename 读入一个要查找的请求的列表进行处理。这个文件包含一些请求，每行一个。文件中的每个条目应当以它们被作为请求提交给 mdig 使用命令行接口同样的方式组织。

-h 使 mdig 打印出全部选项清单的详细帮助并退出。

-v 使 mdig 打印出版本号并退出。

11.7.4 全局选项

-4 选项强制 mdig 仅使用 IPv4 请求传输。

-6 选项强制 mdig 仅使用 IPv6 请求传输。

-b 选项设置请求的源 IP 地址为 address 。这必须是主机的一个网络接口上的一个有效地址，或者 “0.0.0.0”，或者 “::”。可以在其后添加 “#<port>” 指定一个可选的端口。

-m 选项开启内存使用调试。

-p 选项用于请求一个非标准端口。port# 是 mdig 以其作为替代标准的 DNS 端口 53 发送请求的端口号。这个选项是用于测试一个配置成在一个非标准端口监听请求的名字服务器。

全局请求选项是：

+[no]additional 显示 [不显示] 回复的附加部份。缺省是显示。

+[no]all 设置或清除所有显示标志。

+[no]answer 显示 [不显示] 回复的回答部份。缺省是显示。

+[no]authority 显示 [不显示] 回复的权威部份。缺省是显示。

+[no]besteffort 试图显示坏包消息的内容。缺省是不显示坏包回答。

+[no]cl 打印记录时显示 [不显示] 类（CLASS）。

+[no]comments 切换在输出中显示注释行的状态。缺省是打印注释。

+[no]continue 错误时继续（如，超时）。

+[no]crypto 切换 DNSSEC 记录中加密字段的显示。在调试大多数 DNSSEC 验证失败时这些字段的内容不是必须的，删除它们更容易查看通常的失败。缺省是显示这些字段。当省略时，它们被字符串 “[omitted]” 替代，或者在 DNSKEY 的情况下，显示密钥 id 作为替代，例如，“[key id = value]”。

+dscp[=value] 设置发送请求时用到的 DSCP 码点。有效的 DSCP 码点范围是 [0..63]。缺省没有显式设置码点。

+[no]multiline 以详细的多行格式并附带人所易读的注释打印如 SOA 这样的记录。缺省是将每个记录打印在一行中，以适应机器分析 mdig 的输出。

+[no]question 当一个回答返回时，打印 [不打印] 请求的问题部份。缺省是将问题部份作为一个注释打印。

+[no]rrcomments 切换在输出中显示每记录注释的状态（例如，便于人阅读的关于 DNSKEY 记录的密钥信息）。缺省是不打印记录注释，除非多行模式被激活。

+[no]short 提供一个简洁的回答。缺省是以冗长形式打印回答。

+split=W 将资源记录中较长的 hex-或 base64-格式的字段分割为 W 个字符的块（W 被向上取整到距其最近的 4 的倍数上）。+nosplit 或 +split=0 导致字段完全不被分割。缺省为 56 个字符，或者在多行模式时为 44 个字符。

+[no]tcp 在请求名字服务器时使用 [不使用]TCP。缺省行为是使用 UDP。

+[no]ttlid 输出记录时显示 [不显示]TTL。

+[no]ttlunits 显示 [不显示]TTL，以友好地人可读时间单位 “s”，“m”，“h”，“d” 和 “w”，分别代表秒，分，小时，天和周。隐含为 +ttlid。

+[no]vc 在请求名字服务器时使用 [不使用]TCP。这是为 +[no]tcp 提供向后兼容性而使用的替换语法。“vc” 表示 “virtual circuit”。

11.7.5 本地选项

-c 选项设置请求类为 class 。它可以是 BIND 9 所支持的任何有效请求类。缺省请求类是 “IN”。

-t 选项设置请求类型为 type 。它可以是 BIND 9 支持的任何有效请求类型。缺省请求类型是 “A”，除非提供了 -x 选项，指定带有 “PTR” 请求类型的一个反向查找。

反向查找——将地址映射到名字——是由 -x 选项简化。addr 是一个点分十进制形式的 IPv4 地址，或者是一个冒号分隔的 IPv6 地址。mdig 自动执行一个请求名类似 11.12.13.10.in-addr.arpa 的查找，并将请求类型和类分别设置为 PTR 和 IN。缺省时，IPv6 地址使用 IP6.ARPA 域下的半字节格式查找。

本地请求选项是：

+[no]aaflag +[no]aaonly 的同义词。

+[no]aaonly 在请求中设置 “aa” 标志。

+[no]adflag 设置 [不设置] 请求中的 AD（可靠的数据）位。它要求服务器返回回答和权威部份的所有记录是否都已按照服务器的安全策略验证。AD=1 指示所有记录都已被验证为安全并且回答不是来自于一个 OPT-OUT 范围。AD=0 指示回答中的某些部份是不安全的或者没有验证的。这个位缺省是置位的。

+bufsize=B 设置使用 EDNS0 公告的 UDP 消息缓冲大小为 B 字节。这个缓冲的最大值和最小值分别为 65535 和 0。在这个范围之外的值会被适当地调整到高或低。0 之外的值将会发送出一个 EDNS 请求。

+[no]cdflag 设置 [不设置] 请求中的 CD（关闭检查）位。这要求服务器不对响应执行 DNSSEC 验证。

+[no]cookie=#### 带可选值发送一个 COOKIE EDNS 选项。从先前的响应重放一个 COOKIE 将允许服务器标识一个先前的客户端。缺省值是 +nocookie 。

+[no]dnssec 通过在请求的附加部份放置 OPT 记录，并设置 DNSSEC OK 位（DO）来请求发送 DNSSEC 记录。

+[no]edns[=#] 指定请求所带的 EDNS 的版本。有效值为 0 到 255。设置 EDNS 版本会导致发出一个 EDNS 请求。+noedns 清除所记住的 EDNS 版本。缺省时 EDNS 被设置为 0。

+[no]ednsflags[=#] 设置必须为 0 的 EDNS 标志位（Z 位）为指定的值。十进制，十六进制和八进制都是可以的。设置一个命名标志（例如 DO）将被静默地忽略。缺省时，不设置 Z 位。

+[no]ednsopt[=code[:value]] 使用码点 code 和可选荷载 value 指定 EDNS 选项为一个十六进制字符串。一个任意数字值这两者之一。+noednsopt 清除将发送的 EDNS 选项。

+[no]expire 发送一个 EDNS 过期选项。

+[no]nsid 在发送一个请求时包含一个 EDNS 名字服务器 ID 请求。

+[no]recurse 切换请求中的 RD（期望递归）位设置。这个位缺省是置位的，意谓着 mdig 普通情况是发送递归的请求。

+retry=T 设置向服务器重新进行 UDP 请求的次数为 T 次，取代缺省的 2 次。与 +tries 不同，这个不包括初始请求。

+[no]subnet=addr[/prefix-length] 发送（不发送）一个 EDNS 客户端子网选项，带有指定的 IP 地址或网络前缀。

mdig +subnet=0.0.0.0/0，或简写为 mdig +subnet=0，发送一个 EDNS client-subnet 选项，附带一个空地址和一个为 0 的源前缀，它发信号给一个解析器，在解析这个请求时，必须 **不能**使用客户端的地址信息。

+timeout=T 设置一个请求的超时为 T 秒。UDP 传输的缺省超时是 5 秒，TCP 是 10 秒。试图将 T 设置成小于 1 将会得到请求超时为 1 秒的结果。

+tries=T 设置向服务器进行 UDP 请求的重试次数为 T 次，取代缺省的 3 次。如果 T 小于或等于 0，重试次数就静默地向上取整为 1。

+udptimeout=T 设置在 UDP 请求重试之间的超时。

+[no]unknownformat 以未知 RR 类型表示格式（RFC 3597）打印所有 RDATA。缺省是以类型的表示格式打印已知类型的 RDATA。

+[no]yaml 以详细的 YAML 格式打印响应。

+[no]zflag 设置 [不设置] 一个 DNS 请求中最后未赋值的 DNS 头部标志。这个标志缺省是关闭。

11.7.6 参见

dig(1), RFC 1035.

第 11.8 节　named-rrchecker - 针对单个 DNS 资源记录的语法检查器

11.8.1　概要

named-rrchecker [-h] [-o origin] [-p] [-u] [-C] [-T] [-P]

11.8.2　描述

named-rrchecker 从标准输入读取单个 DNS 资源记录并检查其在语句构成上是否正确。

-h 打印输出标准菜单。

-o origin 选项指定一个用于解释记录时的原点。

-p 以正规形式打印输出结果记录。如果不存在正规形式的定义，就以未知记录格式打印记录。

-u 以未知记录形式打印输出结果记录。

-C，-T 和 -P 分别打印输出已知的类，标准类型和私有类型助记符。

11.8.3　参见

RFC 1034, RFC 1035, named(8).

第 11.9 节　arpaname - 将 IP 地址翻译成对应的 ARPA 名字

11.9.1　概要

arpaname {ipaddress ···}

11.9.2　描述

arpaname 将 IP 地址（IPv4 和 IPv6）翻译成对应的 IN-ADDR.ARPA 或 IP6.ARPA 名字。

11.9.3　参见

BIND 9 管理员参考手册。

第 11.10 节　dnssec-revoke - 设置一个 DNSSEC 密钥中的 REVOKED 位

11.10.1　概要

dnssec-revoke [-hr] [-v level] [-V] [-K directory] [-E engine] [-f] [-R] {keyfile}

11.10.2　描述

dnssec-revoke 读一个 DNSSEC 密钥文件，设置在 RFC 5011 中定义的密钥中的 REVOKED 位，并创建一对新的密钥文件，其中包含现在已撤销的密钥。

11.10.3　选项

-h 输出用法消息并退出。

-K directory 设置存放密钥文件的目录。

-r 在写了新的密钥集合文件之后删去原来的密钥集合文件。

-v level 设置调试级别。

-V 打印版本信息。

-E engine 如果适用，指定要使用的加密硬件。

当 BIND 使用带 OpenSSL PKCS#11 支持构建时，这个缺省值是字符串“pkcs11”，它标识一个可以驱动一个加密加速器或硬件服务模块的 OpenSSL 引擎，当 BIND 使用带原生 PKCS#11 加密（–enable-native-pkcs11）构建时，它缺省是由 “–with-pkcs11” 指定的 PKCS#11 提供者库的路径。

-f 强制覆盖：导致 dnssec-revoke 输出新的密钥对，即使一个与被撤销密钥的算法和密钥 ID 都匹配的文件已经存在。

-R 打印带有 REVOKE 位的密钥的密钥标签但不激活密钥。

11.10.4　参见

dnssec-keygen(8), BIND 9 管理员参考手册, RFC 5011.

第 11.11 节　dnssec-cds - 基于 CDS/CDNSKEY 修改一个子区的 DS 记录

11.11.1　概要

dnssec-cds [-a alg…] [-c class] [-D] {-d dsset-file} {-f child-file} [-i [extension]] [-s start-time] [-T ttl] [-u] [-v level] [-V] {domain}

11.11.2　描述

dnssec-cds 命令基于在子区中发布的 CDS 或 CDNSKEY 记录修改一个授权点的 DS 记录。如果 CDS 和 CDNSKEY 记录都出现在子区中，优先使用 CDS。这使得一个子区可以向其父区通知即将对其自身的密钥签名密钥的修改；通过定期使用 dnssec-cds 轮询，父区可以保持 DS 记录的更新，并开启自动对 KSK 的轮转。

要求两个输入文件。-f child-file 选项指定一个包含子区的 CDS 和/或 CDNSKEY 记录加上 RRSIG 和 DNSKEY 记录的文件，这样它们就能够认证。-d path 选项指定一个包含当前 DS 记录的文件的位置。例如，这可能是由 dnssec-signzone 生成的一个 dsset- 文件，或者之前运行 dnssec-cds 的输出。

dnssec-cds 命令使用由 RFC 7344 指定的特定的 DNSSEC 验证逻辑。它要求 CDS 和/或 CDNSKEY 记录是由一个在现存 DS 记录中表示的密钥有效签名。这将是典型的预先存在的密钥签名密钥 (KSK)。

为了防止重放攻击，子记录上的签名不能比它们在之前运行 dnssec-cds 时的签名更老。这个时间是从 dsset- 文件的修改时间，或从 -s 选项获得的。

为了防止授权被破坏，dnssec-cds 确保 DNSKEY 资源记录集能够被新的 DS 资源记录集中的每个密钥算法验证，并且同样的密钥集合能够被每个 DS 摘要类型覆盖。

缺省时，替换的 DS 记录被写到标准输出；使用 -i 选项时，输入文件将被覆盖。当不需要修改时，替换的 DS 记录将与现有记录相同。如果 CDS/ CDNSKEY 记录指定子区不需要安全，则输出可以为空。

警告：当 dnssec-cds 失败时，需要小心，不要删除 DS 记录！

作为一种选择，dnssec-cds -u 将一个 nsupdate 脚本写到标准输出。你可以同时使用 -u 和 -i 选项来维护一个 dsset- 文件，如同执行一个 nsupdate 脚本。

11.11.3 选项

-a algorithm 指定用于转换 CDNSKEY 记录到 DS 记录的摘要算法。这个选项可以重复，这样会为每个 CDNSKEY 记录建立多个 DS 记录。在使用 CDS 记录时，这个选项无效。

algorithm 必须是 SHA-1，SHA-256 或 SHA-384 之一。这些值是大小写不敏感的，并且连字符可以省略。如果没有指定算法，缺省是 SHA-256。

-c class 指定区的 DNS 类。

-D 如果 CDS 和 CDNSKEY 记录都出现在子区中，从 CDNSKEY 生成 DS 记录。缺省优先 CDS 记录。

-d path 父区 DS 记录的位置。这个路径可以是包含 DS 记录的一个文件的名字，或者是一个目录，dnssec-cds 在这个目录中查找域的 dsset- 文件。

为了防止重放攻击，如果子记录的签名时间早于 dsset- 文件的修改时间，子记录将被拒绝。这个可以使用 -s 选项调整。

-f child-file 包含子区的 CDS 和/或 CDNSKEY 记录，及其 DNSKEY 记录和覆盖的 RRSIG 记录的文件，这样它们就能被认证。

下面的例子描述了如何生成这个文件。

-iextension 更新 dsset- 文件，而不是将 DS 记录写到标准输出。

在 -i 和 extension 之间必须没有空格。如果你没有提供 extension，旧的 dsset- 会被丢弃。如果提供了一个 extension，旧 dsset- 文件的备份就被保存在旧文件名跟上 extension 作为新文件名的文件中。

为了防止重放攻击，dsset- 文件的修改时间被设置为与子记录的签名起始时间相匹配，前提是该时间晚于文件的当前修改时间。

-s start-time 指定 RRSIG 记录可以接受的日期和时间。这可以是一个绝对或相对时间。一个绝对开始时间是由一个 YYYYMMDDHHMMSS 格式的数表示的；20170827133700 表示 UTC 时间 2017 年 8 月 27 日 13:37:00。一个相对于 dsset- 文件的时间由-N 表示，即文件修改时间之前 N 秒。一个相对于当前时间的时间用 now+N 表示。

如果没有指定 start-time，就使用 dsset- 文件的修改时间。

-T ttl 指定一个用于新的 DS 记录的 TTL。如果未指定，缺省是旧的 DS 记录的 TTL。如果旧的没有显式的 TTL，新的 DS 记录也没有。

-u 写一个 nsupdate 脚本到标准输出，而不是打印新的 DS 记录。如果不需要修改，输出可以为空。

注意：必须指定新记录的 TTL，要么是在原始的 dsset- 文件中，要么是使用 -T 选项，要么使用 nsupdate ttl 命令。

-V 打印版本信息。

-v level 设置调试级别。级别 1 用于为普通用户提供更详细信息；更高的级别是给开发者使用。

domain 授权点/子区顶点的名字。

11.11.4 退出状态

dnssec-cds 命令运行成功时退出码为 0，如果发生一个错误则为非零。

在成功的情况，DS 记录可能需要也可能不需要修改。

11.11.5 例子

在运行 dnssec-signzone 之前，你可以通过在每个 dsset- 文件上运行 dnssec-cds 来确保授权是最新的。

要获取 dnssec-cds 要求的子记录，你可以在下面的脚本中调用 dig 。如果 dig 失败，也没有关系，因为 dnssec-cds 会执行所有需要的检查。

```
for f in dsset-*
do
  d=${f#dsset-}
  dig +dnssec +noall +answer $d DNSKEY $d CDNSKEY $d CDS |
  dnssec-cds -i -f /dev/stdin -d $f $d
done
```

当父区通过 named 自动签名，你可以使用 dnssec-cds 和 nsupdate 来维护一个授权，如下所示。dsset- 文件允许脚本避免必须获取和验证父区的 DS 记录，并且它还维护了重放攻击保护时间。

```
dig +dnssec +noall +answer $d DNSKEY $d CDNSKEY $d CDS |
dnssec-cds -u -i -f /dev/stdin -d $f $d |
nsupdate -l
```

11.11.6 参见

dig(1), dnssec-settime(8), dnssec-signzone(8), nsupdate(1), BIND 9 管理员参考手册, RFC 7344.

第 11.12 节 dnssec-keygen: DNSSEC 密钥生成工具

11.12.1 概要

dnssec-keygen [-3] [-A date/offset] [-a algorithm] [-b keysize] [-C] [-c class] [-D date/offset] [-d bits] [-D sync date/offset] [-E engine] [-f flag] [-G] [-g generator] [-h] [-I date/offset] [-i interval] [-K directory] [-k policy] [-L ttl] [-l file] [-n nametype] [-P date/offset] [-P sync date/offset] [-p protocol] [-q] [-R date/offset] [-S key] [-s strength] [-T rrtype] [-t type] [-V] [-v level] {name}

11.12.2 描述

dnssec-keygen 为 DNSSEC（安全 DNS）生成密钥，在 RFC 2535 和 RFC 4034 中定义。它也可以为使用在 RFC 2845 中所定义的 TSIG（事务签名）或在 RFC 2930 中所定义的 TKEY（事务密钥）生成密钥。

密钥的 name 在命令行指定。对于 DNSSEC 密钥，这必须与为其生成密钥的区的名字相匹配。

dnssec-keymgr 命令用作 dnssec-keygen 的外包装，按照需要生成和更新密钥，以执行定义好的安全策略，诸如密钥按时轮转。使用 dnssec-keymgr 可能比直接使用 dnssec-keygen 更好。

11.12.3 选项

-3 使用一个 NSEC3 兼容算法生成一个 DNSSEC 密钥。如果这个选项与一个同时具有 NSEC 和 NSEC3 版本的算法一起使用，将会使用 NSEC3 版本；例如，dnssec-keygen -3a RSASHA1 指定了 NSEC3RSASHA1 算法。

-a algorithm 选择加密算法。对 DNSSEC 密钥，algorithm 的值必须为 RSASHA1, NSEC3RSASHA1, RSASHA256, RSASHA512, ECDSAP256SHA256, ECDSAP384SHA384, ED25519 或 ED448 之一。对 TKEY，其值必须为 DH（Diffie Hellman）；指定它的值将会自动设置 -T KEY 选项。

这些值是大小写不敏感的。在某些情况，也支持缩写，如 ECDSA256 代表 ECDSAP256SHA256，而 ECDSA384 代表 ECDSAP384SHA384。如果指定 RSASHA1 并且同时使用 -3 选项，将使用 NSEC3RSASHA1 替代。

这个选项 **必须**被指定，除了使用 -S 选项时，这时将从前一个密钥中拷贝算法。

在先前的版本，HMAC 算法可以用作生成 TSIG 密钥，但是这个特性到 BIND 9.13.0 已被移除了。使用 tsig-keygen 生成 TSIG 密钥。

-b keysize 指定密钥的位数。密钥大小的选择依赖于所使用的算法。RSA 密钥必须在 1024 和 4096 位之间。Diffie Hellman 密钥必须在 128 和 4096 位之间。椭圆曲线算法不需要这个参数。

如果未指定密钥大小，一些算法具有预定义的缺省值。例如，用于 DNSSEC 区签名密钥的 RSA 密钥具有一个缺省为 1024 位的大小；用于密钥签名密钥（KSK，由 -f KSK 生成）的缺省大小为 2048 位。

-C 兼容模式：生成一个旧风格的密钥，不带任何时间元数据。缺省时，dnssec-keygen 将在存放于私钥的元数据中包含密钥的创建日期，其它日期也可以在其中设置（发布日期，激活日期等等）。包含这些数据的密钥可能与旧版本的 BIND 不兼容；-C 选项防止了这些情况。

-c class 指示包含密钥的 DNS 记录应该具有指定的类。如果未指定，使用类 IN。

-d bits 以位为单位的密钥大小。对于算法 RSASHA1，NSEC3RSASA1，RSASHA256 和 RSASHA512，密钥大小必须在范围 1024-4096。DH 大小在 128 和 4096 之间。对于算法 ECDSAP256SHA256，ECDSAP384SHA384，ED25519 和 ED448，这个选项被忽略。

-E engine 如果适用，指定要使用的加密硬件。

当 BIND 使用带 OpenSSL PKCS#11 支持构建时，这个缺省值是字符串“pkcs11”，它标识一个可以驱动一个加密加速器或硬件服务模块的 OpenSSL 引擎，当 BIND 使用带原生 PKCS#11 加密（–enable-native-pkcs11）构建时，它缺省是由 “–with-pkcs11” 指定的 PKCS#11 提供者库的路径。

-f flag 在 KEY/DNSKEY 记录的标志字段中设置特定的标志。只能被识别的标志是 KSK（密钥签名密钥）和 REVOKE。

-G 生成一个密钥，但是不发布它，也不使用它签名。这个选项与-P 和-A 不兼容。

-g generator 如果生成一个 Diffie Hellman 密钥，使用这个生成器。允许值为 2 到 5。如果未指定生成器，如果可能使用来自 RFC 2539 的著名素数；否则缺省为 2。

-h 打印 dnssec-keygen 的选项和参数的简短摘要。

-K directory 设置写密钥文件的目录。

-k policy 为指定的 dnssec-policy 建立密钥。如果一个策略使用了多个密钥，dnssec-keygen 将生成多个密钥。这还将建立一个“.state”文件来跟踪密钥状态。

这个选项根据 dnssec-policy 配置建立密钥，因此它不能与 dnssec-keygen 提供的许多其它选项同时使用。

-L ttl 设置本密钥在被转换进一个 DNSKEY 资源记录中时的缺省 TTL 值。如果这个密钥被导入进一个区，这就被用作密钥的 TTL，除非区中已经有一个 DNSKEY 资源记录集，在后者的情况下，已经存在的 TTL 将会优先。如果未设置这个值并且不存在 DNSKEY 资源记录集，TTL 缺省将是 SOA TTL。将缺省的 TTL 设置为 0 或者 none 就和不设置它有同样的效果。

-l file 提供一个包含 dnssec-policy 语句（与使用 -k 时的策略设置相匹配）的配置文件。

-n nametype 指定密钥的拥有者类型。nametype 的值要么是 ZONE（对 DNSSEC 的区密钥（KEY/DNSKEY)），HOST 或 ENTITY（对一个与主机（KEY）相关的密钥），USER（对一个与用户（KEY）相关的密钥）或 OTHER（DNSKEY)。这些值是大小写不敏感的。缺省是 ZONE，用于 DNSKEY 生成。

-p protocol 为生成的密钥设置协议值，与 -T KEY 一起使用。协议是一个 0 到 255 之间的数。缺省是 3（DNSSEC)。这个参数的其它可能值在 RFC 2535 及其后继中列出。

-q 安静模式：关闭不必要的输出，也包含进度指示。在没有这个选项时，当交互式运行 dnssec-keygen 来生成一个 RSA 或 DSA 密钥对时，它会打印一串符号到 stderr ，以指示生成密钥的进度。一个‘.’表示发现一个随机数，它被传递给一个初始化过滤测试；‘+’表示一个随机数被传递给一个单轮 Miller-Rabin primality 测试；一个空格表示随机数被传递给所有的测试并且是一个合格的密钥。

-S key 创建一个新密钥，它是一个当前存在密钥的明确的后继。这个密钥的名字，算法，大小，和类型都被设置为与现存密钥向匹配。新密钥的激活日期设置为现存密钥的失效日期。其发布日期被设置为激活日期减去发布前间隔，后者缺省是 30 天。

-s strength 指定密钥的强度值。这个强度是 0 到 15 之间的一个数，当前在 DNSSEC 中没有定义其意图。

-T rrtype 为密钥指定所使用的资源记录类型。rrtype 必须是 DNSKEY 或 KEY。在使用一个 DNSSEC 算法时，缺省是 DNSKEY，但是与 SIG(0) 一起使用时，它可以被覆盖为 KEY。

-t type 指定密钥的使用，与 -T KEY 一起使用。type 必须是 AUTOCONF，NOAUTHCONF，NOAUTH 或 NOCONF 之一。缺省是 AUTHCONF。AUTH 为认证数据的能力，而 CONF 为加密数据的能力。

-V 打印版本信息。

-v level 设置调试级别。

11.12.4 定时选项

日期可以被表示成 YYYYMMDD 或 YYYYMMDDHHMMSS 格式。如果参数以‘+’或‘-’开始，它将会被解释成自当前时间始的偏移量。为方便起见，如果这个偏移量带有这些后缀之一，‘y’，‘mo’，‘w’，‘d’，‘h’或‘mi’，这个偏移量就分别被以年（定义为 365 个 24 小时的天，忽略闰年），月（定义为 30 个 24 小时的天），周，天，小时或分钟计算。没有后缀时，偏移量以秒计算。要显式阻止设置一个日期，使用‘none’或‘never’。

-P date/offset 设置一个密钥被发布到区的日期。在此日期之后，密钥将会被包含到区中，但不会用于对其签名。如果未设置，并且没有使用-G 选项，缺省是“now”。

-P sync date/offset 设置匹配这个密钥的 CDS 和 CDNSKEY 记录被发布到区的日期。

-A date/offset 设置密钥被激活的日期。在此日期之后，密钥将会被包含到区中并用于对其签名。如果未设置，并且没有使用-G 选项，缺省是“now”。如果设置，并且未设置-P，公开日期将被设置为激活日期减去提前公开的间隔。

-R date/offset 设置密钥被撤销的日期。在此日期之后，密钥将被标志为被撤销。它将会被包含到区中并用于对其签名。

-I date/offset 设置密钥退出的日期。在此日期之后，密钥仍然被包含在区中，但它不再被用于签名。

-D date/offset 设置密钥被删除的日期。在此日期之后，密钥将不再被包含在区中。(然而，它可能仍然保留在密钥仓库中。)

-D sync date/offset 设置匹配这个密钥的 CDS 和 CDNSKEY 记录被删除的日期。

-i interval 为一个密钥设置发布前间隔。如果设置，则发布日期与激活日期之间必须至少间隔这么多的日期。如果指定了激活日期而没有指定发布日期，则发布日期缺省为激活日期之前这么多时间；相反地，如果指定了发布日期但没有指定激活日期，则激活日期将被设置为在发布日期之后这么多时间。

正在被创建的密钥是另一个密钥的明确后继，则缺省的发布前间隔是 30 天；否则就是零。

与日期偏移量相伴，如果参数后面有后缀‘y’，‘mo’，‘w’，‘d’，‘h’，或‘mi’中的一个，则间隔的单位分别为年，月，周，天，小时，分钟。没有后缀的情况，间隔的单位为秒。

11.12.5 生成的密钥

当 dnssec-keygen 完全成功时，它打印一个 Knnnn.+aaa+iiiii 格式的字符串到标准输出。这是其生成的密钥的标识字符串。

- nnnn 是密钥名。
- aaa 是算法的数字表示。
- iiiii 是密钥标识符（或足迹）。

dnssec-keygen 创建两个文件，其名字类似这个打印的字符串。Knnnn.+aaa+iiiii.key 包含公钥，而 Knnnn.+aaa+iiiii.private 包含私钥。

.key 文件包含一个 DNSKEY 或者 KEY 记录。当一个区被 named 或者 dnssec-signzone -S 签名时，DNSKEY 记录是自动被包含进去的。在其它情况下，.key 文件可以手工或使用一个 $INCLUDE 语句插入到一个区文件中。

.private 文件包含算法相关字段。由于明显的安全原因，这个文件不能具有任何人可读的权限。

11.12.6 例子

要为区 example.com 生成一个 ECDSAP256SHA256 区签名密钥，执行命令：

dnssec-keygen -a ECDSAP256SHA256 example.com

命令将会打印下列格式的字符串：

Kexample.com.+013+26160

在这个例子中，dnssec-keygen 建立文件 Kexample.com.+013+26160.key 和 Kexample.com.+013+26160.private 。

要生成一个对应的密钥签名密钥，执行命令：

dnssec-keygen -a ECDSAP256SHA256 -f KSK example.com

11.12.7 参见

dnssec-signzone(8), BIND 9 管理员参考手册, RFC 2539, RFC 2845, RFC 4034.

第 11.13 节 dnssec-keyfromlabel - DNSSEC 密钥生成工具

11.13.1 概要

dnssec-keyfromlabel {-l label} [-3] [-a algorithm] [-A date/offset] [-c class] [-D date/offset] [-D sync date/offset] [-E engine] [-f flag] [-G] [-I date/offset] [-i interval] [-k] [-K directory] [-L ttl]

[-n nametype] [-P date/offset] [-P sync date/offset] [-p protocol] [-R date/offset] [-S key] [-t type] [-v level] [-V] [-y] {name}

11.13.2 描述

dnssec-keyfromlabel 生成一个密钥对的文件，文件指向存储在一个加密机硬件服务模块（HSM）中的一个密钥对象。私钥文件可以用于 DNSSEC 对区数据的签名，就如同它是一个由 dnssec-keygen 所创建的传统签名密钥一样，但是密钥介质是存放在 HSM 内，实际上签名也在其中进行。

密钥的 name 在命令行指定。这必须与为其生成密钥的区的名字相匹配。

11.13.3 选项

-a algorithm 选择加密算法。algorithm 的值必须为 RSASHA1，NSEC3RSASHA1，RSASHA256，RSASHA512，ECDSAP256SHA256，ECDSAP384SHA384，ED25519 或 ED448 之一。

如果未指定算法，缺省使用 RSASHA1，除非指定了 -3 选项，这时将使用 NSEC3RSASHA1。（如果使用了 -3 并指定了一个算法，将检查这个算法对 NSEC3 的兼容性。）

这些值是大小写不敏感的。在某些情况，也支持缩写，如 ECDSA256 代表 ECDSAP256SHA256，而 ECDSA384 代表 ECDSAP384SHA384。如果指定 RSASHA1 并且同时使用 -3 选项，将使用 NSEC3RSASHA1 替代。

对于 BIND 9.12.0，这个选项是必须的，除了使用 -S 选项时（这时将从前一个密钥中拷贝算法）。之前，对于新生成密钥的缺省算法是 RSASHA1。

-3 使用 NSEC3 兼容算法生成一个 DNSSEC 密钥。如果这个选项与同时具有 NSEC 和 NSEC3 两个版本的算法一起使用，将会使用 NSEC3 版本；例如，dnssec-keygen -3a RSASHA1 指定使用使用 NSEC3RSASHA1 算法。

-E engine 指定要使用的加密硬件。

当 BIND 使用 OpenSSL PKCS#11 支持构建时，这个缺省为字符串 “pkcs11”，它标识一个可以驱动加密加速器或硬件服务模块的 OpenSSL 引擎。当 BIND 使用原生 PKCS#11 加密（–enable-native-pkcs11）构建时，它缺省是由 “–with-pkcs11” 所指定的 PKCS#11 提供者库的路径。

-l label 指定加密硬件中密钥对的标记。

当 BIND 9 使用基于 OpenSSL 的 PKCS#11 支持构建时，这个标记是一个任意的字符串，它标识一个特定的密钥。它由一个可选的 OpenSSL 引擎名开头，后跟一个冒号，如

“pkcs11:keylabel”。

当 BIND 9 使用原生 PKCS#11 支持构建时，这个标记是一个 PKCS#11 URI 字符串，其格式为 “pkcs11:keyword=value[;keyword=value;…]” 关键字包括 “token”，它标识 HSM；“object”，它标识密钥；和 “pin-source”，它标识一个文件，从中可以获得 HSM 的 PIN 码。这个标记将存放在磁盘上的 “private” 文件中。

如果这个标记包含一个 pin-source 字段，使用生成密钥文件的工具将能够使用 HSM 签名和其它操作，而不需要一个操作人员手工输入一个 PIN。注意：使得 HSM 的 PIN 可以以这样的方式访问可能减小使用 HSM 的安全优势；在使用这个特性之前确认这就是你想要做的。

-n nametype 指定密钥的拥有者类型。nametype 的值是 ZONE（对 DNSSEC 的区密钥（KEY/DNSKEY)），HOST 或 ENTITY（对一个与主机（KEY）相关的密钥），USER（对一个与用户（KEY）相关的密钥）或 OTHER（DNSKEY)。这些值是大小写不敏感的。

-C 兼容模式：生成一个旧风格的密钥，不带任何元数据。缺省时，dnssec-keyfromlabel 将在存放于私钥的元数据中包含密钥的创建日期，其它日期也可以在其中设置（发布日期，激活日期等等)。包含这些数据的密钥可能与旧版本的 BIND 不兼容；-C 防止了这些情况。

-c class 指示包含密钥的 DNS 记录应该具有指定的类。如果未指定，使用类 IN。

-f flag 在 KEY/DNSKEY 记录的标志字段中设置特定的标志。只能被识别的标志是 KSK （密钥签名密钥）和 REVOKE。

-G 生成一个密钥，但是不发布它，也不使用它签名。这个选项与-P 和-A 不兼容。

-h 打印 dnssec-keyfromlabel 的选项和参数的简短摘要。

-K directory 设置写密钥文件的目录。

-k 生成 KEY 记录而不是 DNSKEY 记录。

-L ttl 设置本密钥在被转换进一个 DNSKEY 资源记录中时的缺省 TTL 值。如果这个密钥被导入进一个区，这就被用作密钥的 TTL，除非区中已经有一个 DNSKEY 资源记录集，在后者的情况下，已经存在的 TTL 将会优先。将缺省的 TTL 设置为 0 或者 none 来删除它。

-p protocol 为密钥设置协议值。协议是一个 0 到 255 之间的数。缺省是 3（DNSSEC)。这个参数的其它可能值在 RFC 2535 及其后继中列出。

-S key 生成一个密钥，作为一个现存密钥的明确后继。这个密钥的名字，算法，大小和类型要设置成与其前驱向匹配。新密钥的激活日期设置成现存密钥的失活日期。公开日期设置成激活日期减去预先公开的间隔，后者缺省为 30 天。

-t type 指定密钥的使用。type 必须是 AUTOCONF，NOAUTHCONF，NOAUTH 或 NOCONF 之一。缺省是 AUTHCONF。AUTH 为认证数据的能力，而 CONF 为加密数据的能力。

-v level 设置调试级别。

-V 打印版本信息。

-y 即使在密钥 ID 会与一个已存在的密钥冲突的情况下，也允许生成密钥文件，两种情况下密钥都会被撤销。（这仅在你确定不会使用 RFC 5011 信任锚点维护所涉及的密钥时，才是安全的。）

11.13.4 定时选项

日期可以被表示成 YYYYMMDD 或 YYYYMMDDHHMMSS 格式。如果参数以‘+’或‘-’开始，它将会被解释成自当前时间始的偏移量。为方便起见，如果这个偏移量带有这些后缀之一，‘y’，‘mo’，‘w’，‘d’，‘h’或‘mi’，这个偏移量就分别被以年（定义为 365 个 24 小时的天，忽略闰年），月（定义为 30 个 24 小时的天），周，天，小时或分钟计算。没有后缀时，偏移量以秒计算。要显式阻止设置一个日期，使用‘none’或‘never’。

-P date/offset 设置一个密钥被发布到区的日期。在此日期之后，密钥将会被包含到区中，但不会用于对其签名。如果未设置，并且没有使用-G 选项，缺省是“now”。

-P sync date/offset 设置匹配这个密钥的 CDS 和 CDNSKEY 记录被发布到区的日期。

-A date/offset 设置密钥被激活的日期。在此日期之后，密钥将会被包含到区中并用于对其签名。如果未设置，并且没有使用-G 选项，缺省是“now”。

-R date/offset 设置密钥被撤销的日期。在此日期之后，密钥将被标志为被撤销。它将会被包含到区中并用于对其签名。

-I date/offset 设置密钥退出的日期。在此日期之后，密钥仍然被包含在区中，但它不再被用于签名。

-D date/offset 设置密钥被删除的日期。在此日期之后，密钥将不再被包含在区中。（然而，它可能仍然保留在密钥仓库中。）

-D sync date/offset 设置匹配这个密钥的 CDS 和 CDNSKEY 记录被删除的日期。

-i interval 为一个密钥设置发布前间隔。如果设置，则发布日期与激活日期之间必须至少间隔这么多的日期。如果指定了激活日期而没有指定发布日期，则发布日期缺省为激活日期之前这么多时间；相反地，如果指定了发布日期但没有指定激活日期，则激活日期将被设置为在发布日期之后这么多时间。

正在被创建的密钥是另一个密钥的明确后继，则缺省的发布前间隔是 30 天；否则就是零。

与日期偏移量相伴，如果参数后面有后缀‘y’，‘mo’，‘w’，‘d’，‘h’，或‘mi’中的一个，则间隔的单位分别为年，月，周，天，小时，分钟。没有后缀的情况，间隔的单位为秒。

11.13.5 生成的密钥文件

当 dnssec-keyfromlabel 完全成功时，它打印一个 Knnnn.+aaa+iiiii 格式的字符串到标准输出。这是其生成的密钥的标识字符串。

- nnnn 是密钥名。
- aaa 是算法的数字表示。
- iiiii 是密钥标识符（或足迹）。

dnssec-keyfromlabel 创建两个文件，其名字类似这个打印的字符串。Knnnn.+aaa+iiiii.key 包含公钥，而 Knnnn.+aaa+iiiii.private 包含私钥。

.key 文件包含一个 DNS KEY 记录，可以（直接或使用一个 $INCLUDE 语句）插入到一个区文件中。

.private 文件包含算法相关字段。由于明显的安全原因，这个文件不能具有任何人可读的权限。

11.13.6 参见

dnssec-keygen(8), dnssec-signzone(8), BIND 9 管理员参考手册, RFC 4034, PKCS#11 URI 方案 (draft-pechanec-pkcs11uri-13).

第 11.14 节 dnssec-verify - DNSSEC 区验证工具

11.14.1 概要

dnssec-verify [-c class] [-E engine] [-I input-format] [-o origin] [-q] [-v level] [-V] [-x] [-z] {zonefile}

11.14.2 描述

dnssec-verify 验证一个区是被区中每个 DNSKEY 资源记录集中的算法完整地签名，并且 NSEC/NSEC3 链是完整的。

11.14.3 选项

-c class 指定区的 DNS 类。

-E engine 如果适用，指定要使用的加密硬件。

当 BIND 使用带 OpenSSL PKCS#11 支持构建时，这个缺省值是字符串“pkcs11”，它标识一个可以驱动一个加密加速器或硬件服务模块的 OpenSSL 引擎，当 BIND 使用带原生 PKCS#11 加密（–enable-native-pkcs11）构建时，它缺省是由 “–with-pkcs11” 指定的 PKCS#11 提供者库的路径。

-I input-format 输入区文件的格式。可能的格式为 "text" （缺省）和 "raw" 。这个选项的主要目的是用于动态签名区，使导出的区文件以一个非文本的格式，其中所包含的更新可以被独立地验证。使用这个选项对非动态区没有更多的意义。

-o origin 区原点。如果未设置，区文件的名字被当成原点。

-v level 设置调试级别。

-V 打印版本信息。

-q 安静模式：拟制不必要的输出。没有这个选项时，运行 dnssec-verify 将打印在用的密钥数目，用于验证区是否正确签名的算法，其它状态信息。使用这个选项时，所有非错误输出都被拟制，只剩退出码指示是否成功。

-x 只验证使用密钥签名密钥签名的 DNSKEY 资源记录集。没有这个标志时，假定 DNSKEY 资源记录集被所有活动的密钥签名。当设置了这个标志时，如果 DNSKEY 资源记录集未被区签名密钥签名也不成为一个错误。这对应着 dnssec-signzone 中的 -x 选项。

-z 在决定区是否被正确签名时，忽略密钥中的 KSK 标志。没有这个标志时，假设存在一个为撤销，自签名的 DNSKEY，它带有对应于每种算法的 KSK 标志集，并且 DNSKEY 资源记录集之外的其它资源记录集都被一个没有 KSK 标志集的另一个 DNSKEY 所签名。

设置了这个标志时，我们只要求对每种算法，都存在至少一个非活动的，自签名的 DNSKEY，不管其 KSK 标志状态，并且其它资源记录集被一个对应同样算法的非活动密钥签名，这个算法包含自签名密钥；同一密钥可以用于两个目的。这对应着 dnssec-signzone 中的 -z 选项。

zonefile 包含被签名区的文件。

11.14.4 参见

dnssec-signzone(8), BIND 9 管理员参考手册, RFC 4033.

第 11.15 节　dnssec-settime: 为一个 DNSSEC 密钥设置密钥定时元数据

11.15.1　概要

dnssec-settime [-f] [-K directory] [-L ttl] [-P date/offset] [-P ds date/offset] [-P sync date/offset] [-A date/offset] [-R date/offset] [-I date/offset] [-D date/offset] [-D ds date/offset] [-D sync date/offset] [-S key] [-i interval] [-h] [-V] [-v level] [-E engine] {keyfile} [-s] [-g state] [-d state date/offset] [-k state date/offset] [-r state date/offset] [-z state date/offset]

11.15.2　描述

dnssec-settime 读取一个 DNSSEC 私钥文件并按照 -P，-A，-R，-I 和 -D 选项的指定设置密钥定时的元数据。元数据可以用于 dnssec-signzone 或其它签名软件，以决定何时发布一个密钥，密钥是否可以用于对一个区签名等等。

如果命令行中没有设置这些选项，dnssec-settime 只是简单地打印已经存储在密钥中的密钥定时元数据。

当密钥的元数据字段被改变时，密钥对的两个文件（Knnnn.+aaa+iiiii.key 和 Knnnn.+aaa+iiiii.private）都被重新生成。

元数据字段存放在 private 文件中。在 key 文件中也以注释形式存放一份人可读的元数据描述。私钥文件的权限总是被设置为除属主外任何人都不可访问（模式 0600）。

在与状态文件同时工作时，可能更新这些文件中的定时元数据，如同使用 -s 。如果使用了这个选项，你也可以使用 -d （DS），-k （DNSKEY），-r （KSK 的 RRSIG）或 -z （ZSK 的 RRSIG）更新密钥状态。允许的状态为 HIDDEN，RUMOURED，OMNIPRESENT 和 UNRETENTIVE。

你还可以使用 -g 设置密钥的目标状态。这个只能是 HIDDEN 或 OMNIPRESENT（表示密钥是否应该从区中删去或者发布）。

除非是用于测试，不推荐手工操作状态文件。

11.15.3　选项

-f 强制更新一个不带元数据字段的旧格式的密钥。如果没有这个选项，dnssec-settime 在试图更新一个旧密钥时将会失败。有这个选项时，将会以新格式重新生成密钥，但是会保留原始的密钥数据。密钥的创建日期会被设置为当前时间。如果未指定其它值，密钥的发布日期和激活日期也将被设置为当前时间。

-K directory 设置存放密钥文件的目录。

-L ttl 设置本密钥在被转换进一个 DNSKEY 资源记录中时的缺省 TTL 值。如果这个密钥被导入进一个区，这就被用作密钥的 TTL，除非区中已经有一个 DNSKEY 资源记录集，在后者的情况下，已经存在的 TTL 将会优先。如果未设置这个值并且不存在 DNSKEY 资源记录集，TTL 缺省将是 SOA TTL。将缺省的 TTL 设置为 0 或者 none 从密钥中删除它。

-h 输出用法消息并退出。

-V 打印版本信息。

-v level 设置调试级别。

-E engine 如果适用，指定要使用的加密硬件。

当 BIND 使用带 OpenSSL PKCS#11 支持构建时，这个缺省值是字符串“pkcs11”，它标识一个可以驱动一个加密加速器或硬件服务模块的 OpenSSL 引擎，当 BIND 使用带原生 PKCS#11 加密（–enable-native-pkcs11）构建时，它缺省是由 “–with-pkcs11” 指定的 PKCS#11 提供者库的路径。

11.15.4 定时选项

日期可以被表示成 YYYYMMDD 或 YYYYMMDDHHMMSS 格式。如果参数以‘+’或‘-’开始，它将会被解释成自当前时间始的偏移量。为方便起见，如果这个偏移量带有这些后缀之一，‘y’，‘mo’，‘w’，‘d’，‘h’或‘mi’，这个偏移量就分别被以年（定义为 365 个 24 小时的天，忽略闰年），月（定义为 30 个 24 小时的天），周，天，小时或分钟计算。没有后缀时，偏移量以秒计算。要清除一个日期，使用‘none’或‘never’。

-P date/offset 设置一个密钥被发布到区的日期。在此日期之后，密钥将会被包含到区中，但不会用于对其签名。

-P ds date/offset 设置在父区中看到与此密钥相匹配的 DS 记录的日期。

-P sync date/offset 设置匹配这个密钥的 CDS 和 CDNSKEY 记录被发布到区的日期。

-A date/offset 设置密钥被激活的日期。在此日期之后，密钥将会被包含到区中并用于对其签名。

-R date/offset 设置密钥被撤销的日期。在此日期之后，密钥将被标志为被撤销。它将会被包含到区中并用于对其签名。

-I date/offset 设置密钥退出的日期。在此日期之后，密钥仍然被包含在区中，但它不再被用于签名。

-D date/offset 设置密钥被删除的日期。在此日期之后，密钥将不再被包含在区中。（然而，它可能仍然保留在密钥仓库中。）

-D ds date/offset 设置从父区中删除与此密钥相匹配的 DS 记录的日期。

-D sync date/offset 设置匹配这个密钥的 CDS 和 CDNSKEY 记录被删除的日期。

-S predecessor key 选择一个密钥，被修改的密钥是其明确的后继。前驱密钥的名字，算法，大小，和类型必须与被修改密钥的精确匹配。后继密钥的激活日期将被设置为前驱密钥的失效日期。其发布日期被设置为激活日期减去发布前间隔，后者缺省是 30 天。

-i interval 为一个密钥设置发布前间隔。如果设置，则发布日期与激活日期之间必须至少间隔这么多的日期。如果指定了激活日期而没有指定发布日期，则发布日期缺省为激活日期之前这么多时间；相反地，如果指定了发布日期但没有指定激活日期，则激活日期将被设置为在发布日期之后这么多时间。

正在被创建的密钥是另一个密钥的明确后继，则缺省的发布前间隔是 30 天；否则就是零。

与日期偏移量相伴，如果参数后面有后缀‘y’，‘mo’，‘w’，‘d’，‘h’，或‘mi’中的一个，则间隔的单位分别为年，月，周，天，小时，分钟。没有后缀的情况，间隔的单位为秒。

11.15.5 密钥状态选项

已知的密钥状态有 HIDDEN，RUMOURED，OMNIPRESENT 和 UNRETENTIVE。除非用于测试目的，不应该手工修改这些选项。

-s 当设置密钥计时数据时，也更新状态文件。

-g 设置这个密钥的目标状态。必须是 HIDDEN 或 OMNIPRESENT。

-d 设置这个密钥的 DS 状态，以及它的最后修改时间。

-k 设置这个密钥的 DNSKEY 状态，以及它的最后修改时间。

-r 设置这个密钥的 RRSIG（KSK）状态，以及它的最后修改时间。

-z 设置这个密钥的 RRSIG（ZSK）状态，以及它的最后修改时间。

11.15.6 打印选项

dnssec-settime 也能够被用于打印出与一个密钥相关联的定时元数据。

-u 打印 UNIX 纪元格式的时间。

-p C/P/Pds/Psync/A/R/I/D/Dds/Dsync/all 打印一个指定的元数据值或元数据值的集合。-p 选项可以跟随一个或多个下列字符或字符串，以表示要打印哪一个或哪几个值：C 表示创建日期，P 表示发布日期，Pds 表示 DS 发布日期，Psync 表示 CDS 和 CDNSKEY 发布日期，A 表示激活日期，R 表示撤销日期，I 表示失效日期，D 表示删除日期，Dds 表示 DS 删除日期和 Dsync 表示 CDS 和 CDNSKEY 删除日期，使用 all 打印所有的元数据。

11.15.7 参见

dnssec-keygen(8), dnssec-signzone(8), BIND 9 管理员参考手册, RFC 5011.

第 11.16 节 dnssec-importkey - 从外部系统导入 DNSKEY 记录从而可对其进行管理

11.16.1 概要

dnssec-importkey [-K directory] [-L ttl] [-P date/offset] [-P sync date/offset] [-D date/offset] [-D sync date/offset] [-h] [-v level] [-V] {keyfile}

dnssec-importkey {-f filename} [-K directory] [-L ttl] [-P date/offset] [-P sync date/offset] [-D date/offset] [-D sync date/offset] [-h] [-v level] [-V] [dnsname]

11.16.2 描述

dnssec-importkey 读一个公共 DNSKEY 记录并生成一对.key/.private 文件。DNSKEY 记录可以从一个现存的.key 文件中读入，这种情况将会生成一个相关的.private 文件，或者它可以从任何其它文件或者标准输入读入，这时将会生成.key 和.private 文件。

新建立的.private 文件不包含私钥数据，**不能**用于签名。但是，有一个.private 文件使得设置密钥的发布（-P ）和删除（-D ）时间成为可能，这意谓着即使真正的私钥是离线存放，也可以按预计计划将公钥添加到 DNSKEY 资源记录集中，或从中删除。

11.16.3 选项

-f filename 区文件模式：作为公钥文件名的替代，此参数是一个区主文件的域名，它可以从 file 中读入。如果这个域名与 file 相同，这个参数可以忽略。

如果 file 被设置为 "-" ，区数据将从标准输入读入。

-K directory 设置存放密钥文件的目录。

-L ttl 设置本密钥在被转换进一个 DNSKEY 资源记录中时的缺省 TTL 值。如果这个密钥被导入进一个区，这就被用作密钥的 TTL，除非区中已经有一个 DNSKEY 资源记录集，在后者的情况下，已经存在的 TTL 将会优先。将缺省的 TTL 设置为 0 或者 none 来删除它。

-h 输出用法消息并退出。

-v level 设置调试级别。

-V 打印版本信息。

11.16.4 定时选项

日期可以被表示成 YYYYMMDD 或 YYYYMMDDHHMMSS 格式。如果参数以‘+’或‘-’开始，它将会被解释成自当前时间始的偏移量。为方便起见，如果这个偏移量带有这些后缀之一，‘y’，‘mo’，‘w’，‘d’，‘h’或‘mi’，这个偏移量就分别被以年（定义为 365 个 24 小时的天，忽略闰年），月（定义为 30 个 24 小时的天），周，天，小时或分钟计算。没有后缀时，偏移量以秒计算。要显式阻止设置一个日期，使用‘none’或‘never’。

-P date/offset 设置一个密钥被发布到区的日期。在此日期之后，密钥将会被包含到区中，但不会用于对其签名。

-P sync date/offset 设置匹配这个密钥的 CDS 和 CDNSKEY 记录被发布到区的日期。

-D date/offset 设置密钥被删除的日期。在此日期之后，密钥将不再被包含在区中。(然而，它可能仍然保留在密钥仓库中。)

-D sync date/offset 设置匹配这个密钥的 CDS 和 CDNSKEY 记录被删除的日期。

11.16.5 文件

密钥文件可以由密钥标识 Knnnn.+aaa+iiiii 来设计，或者为 dnssec-keygen8 所生成的完整文件名 Knnnn.+aaa+iiiii.key 。

11.16.6 参见

dnssec-keygen(8), dnssec-signzone(8), BIND 9 管理员参考手册, RFC 5011.

第 11.17 节　dnssec-signzone - DNSSEC 区签名工具

11.17.1　概要

dnssec-signzone [-a] [-c class] [-d directory] [-D] [-E engine] [-e end-time] [-f output-file] [-g] [-h] [-i interval] [-I input-format] [-j jitter] [-K directory] [-k key] [-L serial] [-M maxttl] [-N soa-serial-format] [-o origin] [-O output-format] [-P] [-Q] [-q] [-R] [-S] [-s start-time] [-T ttl] [-t] [-u] [-v level] [-V] [-X extended end-time] [-x] [-z] [-3 salt] [-H iterations] [-A] {zonefile} [key…]

11.17.2　描述

dnssec-signzone 签名一个区。它生成 NSEC 和 RRSIG 记录并产生一个区的签名版本。来自这个签名区的授权的安全状态（即，子区是否安全）是由是否存在各个子区的 keyset 文件而决定的。

11.17.3　选项

-a 验证所有生成的签名。

-c class 指定区的 DNS 类。

-C 兼容模式：在对一个区签名时，除了生成 dsset-zonename 之外还生成 keyset-zonename，用于旧版本的 dnssec-signzone 。

-d directory 在 directory 中查找 dsset- 或 keyset- 文件。

-D 输出那些仅由 dnssec-signzone 自动管理的记录类型，即 RRSIG，NSEC，NSEC3 和 NSEC3PARAM 记录。如果使用了智能签名（-S ），也包含 DNSKEY 记录。结果文件可以用 $INCLUDE 包含进原始的区文件中。这个选项不能和 -O raw ， -O map 或序列号更新一起使用。

-E engine 如果适用，指定要使用的加密硬件，例如用于签名的一个安全密钥存储。

当 BIND 使用带 OpenSSL PKCS#11 支持构建时，这个缺省值是字符串“pkcs11”，它标识一个可以驱动一个加密加速器或硬件服务模块的 OpenSSL 引擎，当 BIND 使用带原生 PKCS#11 加密（–enable-native-pkcs11）构建时，它缺省是由 “–with-pkcs11” 指定的 PKCS#11 提供者库的路径。

-g 为来自 dsset- 或 keyset- 文件的子区生成 DS 记录。已经存在的 DS 将被删除。

-K directory 密钥仓库：为搜索 DNSSEC 密钥指定一个目录。如果未指定，缺省为当前目录。

-k key 将指定的密钥当作密钥签名密钥并忽略所有密钥标志。这个选项可以指定多次。

-M maxttl 为签名区设置最大 TTL。输入区中任何超过 maxttl 的 TTL 在输出中将被减小到 maxttl。这为签名区中最大可能的 TTL 提供了确定性，这对知道何时轮转密钥是非常有用的，因为这是被解析器取走的签名能够在解析器缓存中保存的最长可能的时间，使用这个选项签名的区应该配置成在 named.conf 中使用一个一致的 max-zone-ttl 。（注意：这个选项与 -D 不兼容，因为它修改了输出区中的非 DNSSEC 数据。）

-s start-time 指定所生成的 RRSIG 记录生效的日期和时间。这个可以是一个绝对或相对时间。一个绝对开始时间由一个 YYYYMMDDHHMMSS 格式的数所指明；20000530144500 表示 2000 年 5 月 30 日 14:45:00（UTC）。一个相对开始时间由 +N 所指明，N 是从当前时间开始的秒数。如果没有指定 start-time ，就使用当前时间减 1 小时（允许时钟误差）。

-e end-time 指定所生成的 RRSIG 记录过期的日期和时间。与 start-time 一样，一个绝对时间由 YYYYMMDDHHMMSS 格式所指明。一个相对于开始时间的时间由 +N 所指明，即自开始时间之后 N 秒。一个相对于当前时间的时间由 now+N 所指明。如果没有指定 end-time ，就使用开始时间 30 天后作为缺省值。end-time 必须比 start-time 更晚。

-X extended end-time 指定为 DNSKEY 资源记录集而生成的 RRSIG 记录的过期日期和时间。这是用于 DNSKEY 签名的有效时间需要比其它记录签名的有效时间持续更长的情况；例如，当 KSK 的私密部份被离线保存并且需要手动刷新 KSK 签名时。

与 start-time 一样，一个绝对时间由 YYYYMMDDHHMMSS 格式所指明。一个相对于开始时间的时间由 +N 所指明，即自开始时间之后 N 秒。一个相对于当前时间的时间由 now+N 所指明。如果没有指定 extended end-time ，就使用 end-time 的值作为缺省值。（相应地，end-time 的缺省值为开始时间的 30 天后。）extended end-time 必须比 start-time 更晚。

-f output-file 包含签名区的输出文件的名字。缺省是在输入文件名后面添加 .signed 。如果 output-file 被设置成 "-" ，签名区将被写到标准输出，以缺省的输出格式 “full”。

-h 打印 dnssec-signzone 的选项和参数的简短摘要。

-V 打印版本信息。

-i interval 当一个先前已签名的区被作为输入，记录可能被再次签名。interval 选项指定作为自当前时间开始的偏移量（以秒计）的循环间隔。如果一个 RRSIG 记录在这个循环间隔后过期，它会被保留。否则，它被考虑为马上过期，并被替代。

缺省的循环间隔是签名的结束时间和开始时间之差的四分之一，所以如果既不指定 end-time ，也不指定 start-time ，dnssec-signzone 生成的签名在 30 天内有效，并带有 7.5 天的循环间隔。所以，如果任何现存的 RRSIG 记录将在 7.5 天以内过期，它们将会被替代。

-I input-format 输入区文件的格式。可能的格式是 "text" （缺省），"raw" 和 "map" 。这个选项

主要用于动态签名区，这样一个包含动态更新的以非文本格式转储的区文件就可以被直接签名。使用这个选项对非动态区没有意义。

-j jitter 在使用一个固定的签名生存时间对一个区签名时，所有的 RRSIG 记录都分配了几乎是同时的签名过期时间。如果区被增量签名，例如，一个先前签过名的区作为输入传递给签名者，所有过期的签名必须在大致相同的时间被重新生成。jitter 选项指定了一个抖动窗口，用来随机化签名的过期时间，这样就将增量签名的重生成扩展到一个时间段。

签名生存时间抖动通过分散缓存过期时间对验证者和服务器也有某种程度的帮助，例如，如果所有的缓冲中都没有大量 RRSIG 在同一时间过期，就比所有验证者需要在几乎相同的时刻来重新获取记录有更少的拥塞。

-L serial 当以 “raw” 或 “map” 格式输出一个签名区时，在头部中设置 “source serial” 值以指定序列号。（这个功能预期主要用于测试目的。）

-n ncpus 指定要使用的线程个数。缺省时，为每个被检测到的 CPU 绑定一个线程。

-N soa-serial-format 签名区的 SOA 序列号格式。可能的格式有 "keep" （缺省），"increment" ，"unixtime" 和 "date" 。

"keep" 不改变 SOA 序列号。

"increment" 使用 RFC 1982 算术增加 SOA 序列号。

"unixtime" 将 SOA 序列号设置为 UNIX 纪元以来的秒数。

"date" 将 SOA 序列号以 YYYYMMDDNN 的格式设置为今天的日期。

-o origin 区起点。如果未指定，就使用区名作为起点。

-O output-format 包含签名区的输出文件的格式。可能的格式为 "text" （缺省），它是区的标准文本格式；"full" ，它是以文本输出的适合由外部脚本处理的格式，和 "map" ，"raw" 和 "raw=N" ，它是以二进制格式存储区以便 named 快速加载。"raw=N" 指定 raw 区文件的格式版本：如果 N 为 0，raw 区文件可以被任何版本的 named 读取；如果 N 为 1，这个文件则只能被 9.9.0 或更高版本读取。缺省为 1。

-P 关闭签名验证后测试。

签名验证后测试确保对每个用到的算法都有至少一个非撤销自签名的 KSK 密钥，所有撤销的 KSK 都是自签名的，以及区中所有记录都是由这个算法所签名的。这个选项跳过这些测试。

-Q 删除不再活动的密钥的签名。

通常情况，当一个以前已经签名的区被作为输入传递给签名者时，并且一个 DNSKEY 记录被删除且被一个新的所替代时，来自旧密钥的并且仍在其有效期内的签名将被保留。这允许区继续使用缓存中的旧 DNSKEY 资源记录集来作验证。-Q 强制 dnssec-signzone 删除不再活

动的密钥的签名。这使 ZSK 使用 RFC 4641#4.2.1.1 （“Pre-Publish Key Rollover”）中描述的过程进行轮转。

-q 安静模式：拟制不必要的输出。没有这个选项时，运行 dnssec-signzone 将打印在用的密钥数目，用于验证区是否正确签名的算法，其它状态信息，以及包含签名区的最终文件名。使用这个选项时，输出被拟制，只剩下文件名。

-R 删除不再公开的密钥的签名。

这个选项与 -Q 相似，除了它强制 dnssec-signzone 从不再公开的密钥签名之外。这使 ZSK 使用 RFC 4641#4.2.1.2 （“Double Signature Zone Signing Key Rollover”）中描述的过程进行轮转。

-S 智能签名：指示 dnssec-signzone 在密钥仓库中搜索与被签名区匹配的密钥，如果有合适的还要将其包含到区中。

当找到了一个密钥时，就检查其计时元数据以决定如何根据以下的规则来使用它。每个后面的规则优先于其之前的规则：

如果没有为密钥指定计时元数据，密钥被发布在区中并用于对区签名。

如果设置了密钥的发布日期并且已经到了，密钥就被发布到区中。

如果设置了密钥的激活日期并且已经到了，密钥就被发布（忽略发布日期）并用于对区签名。

如果设置了密钥的撤销日期并且已经到了，并且密钥已被发布，就撤销密钥，已撤销的密钥可用于对区签名。

如果设置了密钥的停止公开日期或删除日期之一并且已经到了，密钥不再公开或用于对区签名，而不管任何其它元数据。

如果设置了密钥的同步发布日期并且已经过了，就建立同步记录（类型 CDS 和/或 CDNSKEY）。

如果设置了密钥的同步删除日期并且已经过了，就删除同步记录（类型 CDS 和/或 CDNSKEY）。

-T ttl 为从密钥仓库导入到区中的新 DNSKEY 记录指定一个 TTL。如果未指定，缺省是区的 SOA 记录中的 TTL 值。当不使用 -S 签名时这个选项被忽略，因为在那种情况下，不会从密钥仓库导入 DNSKEY 记录。同样，如果在区顶点存在任何 DNSKEY 记录时，也会忽略这个选项，在这个情况中，新记录的 TTL 值将会被设置成与其匹配，或者如果任何被导入的 DNSKEY 记录有一个缺省的 TTL 值时也会被忽略。在导入密钥中的 TTL 值有冲突的情况下，使用时间最短的一个。

-t 在完成时打印统计结果。

-u 当对之前已签过名的区重新签名时更新 NSEC/NSEC3 链。带有这个选项时，一个使用 NSEC 签名的区可以转换到 NSEC3，或者一个使用 NSEC3 签名的区可以转换为 NSEC 或其它参数的 NSEC3。没有这个选项时，重新签名时，dnssec-signzone 将维持已存在的链。

-v level 设置调试级别。

-x 仅使用密钥签名密钥对 DNSKEY，CDNSKEY 和 CDS 资源记录集签名，并忽略来自区签名密钥的签名。（这与 named 中的 dnssec-dnskey-kskonly yes; 区选项相似。）

-z 在决定要签名什么东西时，忽略密钥中的 KSK 标志。这导致有 KSK 标志的密钥对所有记录签名，而不仅仅是 DNSKEY 资源记录集。（这与 named 中的 update-check-ksk no; 区选项相似。）

-3 salt 使用给定的十六进制编码的干扰值（salt）生成一个 NSEC3 链。在生成 NSEC3 链时，可以使用一个破折号（salt）来指示不使用干扰值（salt）。

-H iterations 在生成一个 NSEC3 链时，使用这个循环次数。缺省是 10。

-A 在生成一个 NSEC3 链时，设置所有 NSEC3 记录的 OPTOUT 标志，并且不为不安全的授权生成 NSEC3 记录。

使用这个选项两次（例如，-AA ）关闭所有记录的 OPTOUT 标志。这在使用 -u 选项修改一个先前具有 OPTOUT 集合的 NSEC3 链时很有用。

zonefile 包含被签名区的文件。

key 指定应该使用那个密钥来签名这个区。如果没有指定密钥，会对区进行检查，在区顶点找 DNSKEY 记录。如果在当前目录找到并与私钥匹配，这个就会用于签名。

11.17.4 例子

下列命令使用由 dnssec-keygen 所生成的 ECDSAP256SHA256 密钥（Kexample.com.+013+17247）对 example.com 区签名。因为没有使用 -S 选项，区的密钥必须在主文件中（db.example.com ）。这个需要在当前目录查找 dsset 文件，这样 DS 记录可以从中导入（-g ）。

```
% dnssec-signzone -g -o example.com db.example.com \
Kexample.com.+013+17247
db.example.com.signed
%
```

在上述例子中，dnssec-signzone 创建文件 db.example.com.signed 。这个文件被 named.conf 文件中的区语句所引用。

这个例子使用缺省参数重新对先前的签名区签名。假定私钥存放在当前目录。

```
% cp db.example.com.signed db.example.com
% dnssec-signzone -o example.com db.example.com
db.example.com.signed
%
```

11.17.5 参见

dnssec-keygen(8), BIND 9 管理员参考手册, RFC 4033, RFC 4641.

第 11.18 节 dnssec-dsfromkey - DNSSEC DS 资源记录生成工具

11.18.1 概要

dnssec-dsfromkey [-1 | -2 | -a alg] [-C] [-T TTL] [-v level] [-K directory] {keyfile}

dnssec-dsfromkey [-1 | -2 | -a alg] [-C] [-T TTL] [-v level] [-c class] [-A] {-f file} [dnsname]

dnssec-dsfromkey [-1 | -2 | -a alg] [-C] [-T TTL] [-v level] [-c class] [-K directory] {-s} {dnsname}

dnssec-dsfromkey [-h | -V]

11.18.2 描述

dnssec-dsfromkey 命令输出 DS（Delegation Signer，授权签名者）资源记录（RRs），或者带有 -C 时它输出 CDS（子 DS）资源记录。

输入密钥可以以数种方式指定：

缺省时，dnssec-dsfromkey 读取一个名字类似 Knnnn.+aaa+iiiii.key 的密钥文件，这是由 dnssec-keygen 生成的。

带有 -f file 选项时，dnssec-dsfromkey 从一个区文件或部份区文件（可以只包含 DNSKEY 记录）中读取密钥。

带有 -s 选项时，dnssec-dsfromkey 读一个 keyset- 文件，这是由 dnssec-keygen -C 生成的。

11.18.3 选项

-1 -a SHA1 的缩写。

-2 -a SHA-256 的缩写。

-a algorithm 指定一个用于转换 DNSKEY 记录到 DS 记录的摘要算法。这个选项可以重复，这样就为每个 DNSKEY 记录生成多个 DS 记录。

algorithm 的值必须是 SHA-1，SHA-256 或 SHA-384 之一。这些值是大小写不敏感的，而且连字符可以省略。如果没有指定算法，缺省是 SHA-256。

-A 当生成 DS 记录时包含 ZSK。没有这个选项时，只有具有 KSK 标志的密钥被转换为 DS 记录并打印。仅用于 -f 区文件模式。

-c class 指定 DNS 类（缺省是 IN），仅用于 -s 密钥集合中或者 -f 区文件模式。

-C 生成 CDS 记录而不是 DS 记录。

-f file 区文件模式：dnssec-dsfromkey 的最终 dnsname 参数是一个其主文件可以从 file 中读取的区的 DNS 域名。如果区名与 file 相同，这个参数可以忽略。

如果 file 为 "-"，区数据将从标准输入读入。这使得使用 dig 命令的输出作为输入成为可能，例如：

dig dnskey example.com | dnssec-dsfromkey -f - example.com

-h 打印用法信息。

-K directory 在 directory 中查找密钥文件或者 keyset- 文件。

-s 密钥集合模式：dnssec-dsfromkey 的最终 dnsname 参数是 DNS 域名，用于定位一个 keyset- 文件。

-T TTL 指定 DS 记录的 TTL。缺省时 TTL 是省略的。

-v level 设置调试级别。

-V 打印版本信息。

11.18.4 例子

要从 Kexample.com.+003+26160 密钥文件（keyfile）名构建 SHA-256 DS 资源记录，你可以执行下列命令：

dnssec-dsfromkey -2 Kexample.com.+003+26160

命令将输出类似下面的内容：

example.com. IN DS 26160 5 2 3A1EADA7A74B8D0BA86726B0C227AA85AB8BBD2B2004F41A868A54F0C5EA0

11.18.5 文件

密钥文件可以由密钥标识 Knnnn.+aaa+iiiii 来指定或者是由 dnssec-keygen8 所生成的完整文件名 Knnnn.+aaa+iiiii.key 。

密钥集合文件名是从 directory ，字符串 keyset- 和 dnsname 中构建的。

11.18.6 注意

即使文件存在，一个密钥文件错误也会给出一个“file not found”消息。

11.18.7 参见

dnssec-keygen(8), dnssec-signzone(8), BIND 9 管理员参考手册, RFC 3658 (DS RRs), RFC 4509 (SHA-256 for DS RRs), RFC 6605 (SHA-384 for DS RRs), RFC 7344 (CDS and CDNSKEY RRs).

第 11.19 节 dnssec-checkds - DNSSEC 授权一致性检查工具

11.19.1 概要

dnssec-checkds [-ddig path] [-Ddsfromkey path] [-ffile] [-ldomain] [-sfile] {zone}

11.19.2 描述

dnssec-checkds 为指定区中的密钥验证授权签名者（DS）资源记录的正确性。

11.19.3 选项

-a algorithm

指定在转换区的 DNSKEY 记录到期待的 DS 记录时的摘要算法。这个选项可以重复，这样，对每个 DNSKEY 记录，将检查多个记录。

algorithm 必须是 SHA-1，SHA-256 或 SHA-384 之一。这些值是大小写不敏感的，并且连字符可以忽略。如果未指定算法，缺省是 SHA-256。

-f file

如果指定了一个 file ，就从那个文件中读入区以查找 DNSKEY 记录。如果没有，就在 DNS 中查找区的 DNSKEY 记录。

-s file

指定一个预先准备的 dsset 文件，例如由 dnssec-signzone 所生成的，用作 DS 资源记录集的来源而不是请求父域。

-d dig path

给一个 dig 程序指定一个路径。用于测试。

-D dsfromkey path

给一个 dnssec-dsfromkey 程序指定一个路径。用于测试。

11.19.4 参见

dnssec-dsfromkey(8), dnssec-keygen(8), dnssec-signzone(8),

第 11.20 节 dnssec-coverage - 检查一个区 DNSKEY 将来的覆盖

11.20.1 概要

dnssec-coverage [-Kdirectory] [-llength] [-ffile] [-dDNSKEY TTL] [-mmax TTL] [-rinterval] [-ccompilezone path] [-k] [-z] [zone…]

11.20.2 描述

dnssec-coverage 验证一个给定的区或一个区集合的 DNSSEC 密钥是正确设置了定时元数据以确保将来没有 DNSSEC 覆盖的失误。

如果指定了 zone ，在密钥仓库中与这个区匹配的密钥都会被扫描，并为那个密钥生成一个事件日程的顺序列表（如，发布，激活，失效，删除）。事件列表以发生顺序遍历。如果任何事件在进行

时，可能导致区进入一个可能发生验证失败的状态时，会生成一个警告。例如，如果一个对给定算法，其发布或激活的密钥数下降到零，或者如果一个密钥在一个新密钥轮转后从其区中被太快地删除，由前一个密钥签名的缓存数据还没时间从解析器的缓存中过期。

如果未指定 zone，在密钥仓库中的所有密钥都会被扫描，所有带密钥的区都会被分析。（注意：这个报告方法只在所有带有给定仓库中密钥的区共享同样的 TTL 参数时才是精确的。）

11.20.3 选项

-K directory

设置能够找到密钥的目录。缺省为当前工作目录。

-f file

如果指定了一个 file，区就从那个文件读取；最大 TTL 和 DNSKEY TTL 就直接从区数据决定，就不需要在命令行指定 -m 和 -d 选项。

-l duration

检查 DNSSEC 覆盖的时间长度。计划在超过 duration 的将来的密钥事件将被忽略，并假设为正确的。

duration 的值可以按秒设置，或通过增加一个后缀设为更大的时间单位：mi 表示分钟，h 表示小时，d 表示天，w 表示周，mo 表示月，y 表示年。

-m maximum TTL

在决定是否存在一个验证失败的可能性时，为一个或多个被分析的区设置最大 TTL 值。当一个区签名密钥失效时，在密钥被剔除出 DNSKEY 资源记录集之前，必须有足够的时间，让区中最大 TTL 的记录在解析器的缓存中过期。如果这个条件不满足，将会产生一个警告。

TTL 长度可以按秒设置，或通过增加一个后缀设为更大的时间单位：mi 表示分钟，h 表示小时，d 表示天，w 表示周，mo 表示月，y 表示年。

如果使用 -f 指定了一个区文件，这个选项是不必要的。如果指定了 -f，仍然可以使用这个选项；它将覆盖在文件中发现的值。

如果没有使用这个选项并且不能从一个区文件提取到最大 TTL，将生成一个警告，并使用 1 周作为缺省值。

-d DNSKEY TTL

在决定是否存在一个验证失败的可能性时，为一个或多个被分析的区设置用作 DNSKEY TTL 的值。当一个密钥被轮转时（即被一个新密钥替代），在新密钥被激活并开始生成签名之前，必须有足够的时间让旧的 DNSKEY 资源记录集在解析器缓存中过期。如果这个条件不满足，将会产生一个警告。

TTL 长度可以按秒设置，或通过增加一个后缀设为更大的时间单位：mi 表示分钟，h 表示小时，d 表示天，w 表示周，mo 表示月，y 表示年。

如果使用 -f 指定了一个区文件来读入 DNSKEY 资源记录集的 TTL，或者使用 dnssec-keygen 的 -L 设定一个缺省的密钥 TTL，这个选项是不必要的。如果上述一项是真，仍然可以使用这个选项；它将覆盖在区文件或密钥文件中发现的值。

如果没有使用这个选项并且不能从区文件或密钥文件提取到密钥 TTL，将生成一个警告，并使用 1 天作为缺省值。

-r resign interval

在决定是否存在一个验证失败的可能性时，为一个或多个被分析的区设置用作放弃间隔（resign interval）的值。这个值缺省为 22.5 天，也是 named 中的缺省值。然而，如果在 named.conf 使用 sig-validity-interval 选项修改了，它应该在这里被修改。

TTL 长度可以按秒设置，或通过增加一个后缀设为更大的时间单位：mi 表示分钟，h 表示小时，d 表示天，w 表示周，mo 表示月，y 表示年。

-k

只检查 KSK 覆盖；忽略 ZSK 事件。不能与 -z 一起使用。

-z

只检查 ZSK 覆盖；忽略 KSK 事件。不能与 -k 一起使用。

-c compilezone path

指定一个 named-compilezone 二进制文件的路径。用于测试。

11.20.4 参见

dnssec-checkds(8), dnssec-dsfromkey(8), dnssec-keygen(8), dnssec-signzone(8)

第 11.21 节 dnssec-keymgr - 为一个基于一个已定义策略的区确保正确的 DNSKEY 覆盖

11.21.1 概要

:program:dnssec-keymgr [-Kdirectory] [-cfile] [-f] [-k] [-q] [-v] [-z] [-gpath] [-spath] [zone…]

11.21.2 描述

dnssec-keymgr 是一个高级 Python 外包装以使 BIND 处理区的密钥轮转进程更加容易。它使用 BIND 命令操纵 DNSSEC 密钥元数据：dnssec-keygen 和 dnssec-settime 。

DNSSEC 策略可以从一个配置文件（缺省是/etc/dnssec-policy.conf）读取，从中任何给定区的密钥参数，发布和轮转时间表，以及期望的覆盖时间段都可以被决定。这个文件可以用于基于每个区定义单独的 DNSSEC 策略，或者为所有区设置一个“default”策略。

当运行 dnssec-keymgr 时，它检查一个或多个区的 DNSSEC 密钥，将这些区的时间元数据与其策略进行对比。如果密钥设置不符合 DNSSEC 策略（例如，由于策略被修改），他们就会被自动纠正。

一个区策略可以指定一段我们想要确保密钥正确性的持续时间（coverage ）。它也可以指定一个轮转周期（roll-period ）。如果策略指示一个密钥应当在覆盖周期结束之前轮转，就会自动创建一个后继密钥并将其添加到密钥串的末尾。

如果在命令行指定了区，dnssec-keymgr 将只检查这些区。如果一个指定的区没有在适当的地方有密钥，就会根据策略为其自动生成密钥。

如果 **没有**在命令行指定区，dnssec-keymgr 将搜索密钥目录（要么是当前工作目录，要么是由 -K 选项设置的目录），并检查出现在目录中的所有区的密钥。

密钥时间在过去的将不会被更新，除非使用了 -f （参见下面）。密钥失活和删除时间小于五分钟之后的将会被延长五分钟。

预期这个工具将会自动地运行并无需人工照看（例如，通过 cron ）。

11.21.3 选项

-c file

> 如果指定了 -c , 就从文件 file 读取 DNSSEC 策略。(如果未指定，就从文件/etc/dnssec-policy.conf 读取策略；如果那个文件不存在，就使用一个内置的全局缺省策略。)

-f

强制：允许密钥事件的更新，即使它们已经过时。不推荐在密钥已经发布的区使用这个。然而，如果一些密钥被生成，所有密钥的发布日期和激活日期都在过去，但是密钥还未在区中发布，这时这个选项可以用于清理它们并使用合适的轮转间隔将其变成一个适当的密钥集合。

-g keygen-path

给 dnssec-keygen 二进制程序指定一个路径。用于测试。参见 -s 选项。

-h

输出 dnssec-keymgr 帮助概要并退出。

-K directory

设置能够找到密钥的目录。缺省是当前工作目录。

-k

仅应用策略到 KSK 密钥。参见 -z 选项。

-q

安静：禁止 dnssec-keygen 和 dnssec-settime 的输出。

-s settime-path

给 dnssec-settime 二进制文件指定一个路径。用于测试。参见 -g 选项。

-v

输出 dnssec-keymgr 版本并退出。

-z

仅应用策略到 ZSK 密钥。参见 -k 选项。

11.21.4 策略配置

dnssec-policy.conf 文件可以指定三种策略：

. **策略类**（policyname{ ... }; ）可以被区策略或者其它策略类继承；这些可以用于建议不同安全档案的集合。例如，一个策略类 normal 可能指定 1024 位的密钥长度，但是一个类 extra 可能指定 2048 位取代它；extra 用于那些有不一般的高安全需求的区。

. **算法策略：**（algorithm-policyalgorithm{ ... }; ）覆盖缺省的按每个算法的设置。例如，缺省时，RSASHA256 密钥对 KSK 和 ZSK 这两者都使用 2048 位密钥长度。这个可以使用 algorithm-policy 修改，并且新的密钥长度就可以被用于任何 RSASHA256 类型的密钥。

. **区策略：**（zonename{ ... }; ）按区名字为单一区设置策略。一个区策略可以通过包含一个 policy 选项继承一个策略类。以数字（即，0-9）开头的区名必须加引号。如果一个区没有自己的策略，就应用 “default” 策略。

可以在策略中指定的选项：

algorithm name;

密钥算法。如果未定义策略，缺省是 RSASHA256。

coverage duration;

确保密钥正确的时间长度；在这个时间之后，将不采取措施建立并激活新密钥。这可以被表示为一个以秒为单位的数，或者使用人可读的单位表示的一段时间（例如：“1y” 或者 “6 months”）。这个选项的一个缺省值可以被设置在算法策略中，和在策略类或区策略中一样。如果未配置策略，缺省是六个月。

directory path;

指定密钥应该存放的目录。

key-size keytype size;

指定用于创建密钥的位数。keytype 要么是 “zsk”，要么是 “ksk”。这个选项的一个缺省值可以设置在算法策略中，和在策略类或区策略中一样。如果没有配置策略，RSA 密钥的缺省值是 2048 位。

keyttl duration;

密钥的 TTL。如果没有定义策略，缺省是一小时。

post-publish keytype duration;

在一个密钥失活后多长时间应将其从区中删除。注意：如果未设置 roll-period ，这个值将被忽略。keytype 要么是 “zsk”，要么是 “ksk”。这个选项的一个缺省持续时间可以设置在算法策略中，和在策略类或区策略中一样。缺省值是一个月。

pre-publish keytype duration;

在一个密钥激活之前多长时间应该发布它。注意：如果未设置 roll-period ，这个值将被忽略。keytype 要么是 “zsk”，要么是 “ksk”。这个选项的一个缺省持续时间可以设置在算法策略中，和在策略类或区策略中一样。缺省值是一个月。

roll-period keytype duration;

密钥应以多大频率被轮转。keytype 要么是 “zsk”，要么是 “ksk”。这个选项的一个缺省持续时间可以设置在算法策略中，和在策略类或区策略中一样。如果没有配置策略，对 ZSK 缺省值是一年。KSK 缺省不轮转。

standby keytype number;

还未实现。

11.21.5 剩余工作

. 通过给 dnssec-keygen 和 dnssec-settime 使用 -P sync 和 -D sync 选项打开 KSK 轮转的调度。检查父区（如同在 dnssec-checkds 中一样）以决定何时轮转密钥是安全的。

. 允许为使用 RFC 5011 语义的密钥配置后备密钥和使用 REVOKE 位。

11.21.6 参见

dnssec-coverage(8), dnssec-keygen(8), dnssec-settime(8), dnssec-checkds(8)

第 11.22 节 filter-aaaa.so - 当 A 记录存在时从 DNS 响应中过滤 AAAA 记录

11.22.1 概要

plugin query “filter-aaaa.so” [{ parameters }];

11.22.2 描述

filter-aaaa.so 是一个 named 的请求插件模块，使 named 能够在给客户端响应时省略某些 IPv6 地址。

在 BIND 9.12 之前，这个特性是在 named 中原生实现的，并使用 filter-aaaa ACL 及 filter-aaaa-on-v4 和 filter-aaaa-on-v6 选项开启。这些选项现在在 named 中已被废弃，但是可以作为参数传送给 filter-aaaa.so 插件，例如：

```
plugin query "/usr/local/lib/filter-aaaa.so" {
    filter-aaaa-on-v4 yes;
    filter-aaaa-on-v6 yes;
    filter-aaaa { 192.0.2.1; 2001:db8:2::1; };
};
```

这个模块旨在协助从 IPv4 到 IPv6 的迁移，当正在查找的名称有一个可用的 IPv4 地址时，阻止将 IPv6 地址返回给没有 IPv6 连接的 DNS 客户端。不推荐使用这个模块，除非绝对需要。

注意：这个机制可能错误地导致其它服务器不对其客户端返回 AAAA 记录。如果一个具有 IPv6 和 IPv4 双栈网络连接的递归服务器使用这个机制通过 IPv4 请求一个权威服务器，它将拒绝 AAAA 记录，即使其客户端使用 IPv6。

11.22.3 选项

filter-aaaa 指定一个客户端地址列表，对这些地址应用 AAAA 过滤。缺省是 any 。

filter-aaaa-on-v4 如果设置为 yes ，DNS 客户端是 filter-aaaa 中的一个 IPv4 地址，如果响应中不包括 DNSSEC 签名，那么响应中的所有 AAAA 记录都被删除。这个过滤动作应用到所有响应，不只是权威响应。

如果设置为 break-dnssec ，即使 DNSSEC 开启，也会删掉 AAAA 记录。如同名字所揭示的，这会导致对响应的验证失败，因为 DNSSEC 协议就是设计用来检测删除的。

这个机制可能错误地导致其它服务器不对其客户端返回 AAAA 记录。一个具有 IPv6 和 IPv4 双栈网络连接的递归服务器使用这个机制通过 IPv4 请求一个权威服务器，它将拒绝 AAAA 记录，即使其客户端使用 IPv6。

filter-aaaa-on-v6 与 filter-aaaa-on-v4 相同，只是它过滤来自 IPv6 客户端而不是 IPv4 客户端查询 AAAA 的响应。要过滤所有响应，将两个选项都设置为 yes 。

11.22.4 参见

BIND 9 管理员参考手册。

第 11.23 节　ddns-confgen - ddns 密钥生成工具

11.23.1　概要

tsig-keygen [-a algorithm] [-h] [-r randomfile] [name]

ddns-confgen [-a algorithm] [-h] [-k keyname] [-q] [-r randomfile] [-s name] [-z zone]

11.23.2　描述

tsig-keygen 和 ddns-confgen 是一个应用程序的调用方法，它为使用 TSIG 签名生成密钥。例如，作为结果的密钥可以被用于加固对一个区的动态 DNS 更新或者用于 rndc 命令通道。

当作为 tsig-keygen 运行时，可以在命令行指定一个域名，它将被用作所生成密钥的名字。如果未指定名字，缺省为 tsig-key 。

当作为 ddns-confgen 运行时，所生成的密钥伴随有设置动态 DNS 时用于 nsupdate 和 named 的配置文件和指令，包括一个 update-policy 语句的例子。（这个用法类似于用 rndc-confgen 命令设置命令通道的安全。）

注意 named 自己可以配置一个本地 DDNS 密钥，并用于 nsupdate -l ：它在区被配置为 update-policy local; 时才这样做。ddns-confgen 只在更复杂的配置才需要：例如，如果 nsupdate 用于来自一个远程系统。

11.23.3　选项

-a algorithm 指定用于 TSIG 密钥的算法。可用的选择为：hmac-md5，hmac-sha1，hmac-sha224，hmac-sha256，hmac-sha384 和 hmac-sha512。缺省为 hmac-sha256。选项是大小写无关的，前缀“hmac-”可以被忽略。

-h 打印选项和参数的一个简短摘要。

-k keyname 指定 DDNS 认证密钥的密钥名。当既没有指定 -s ，也没有指定 -z 选项时，缺省是 ddns-key ；否则，缺省将 ddns-key 作为一个独立的标记，后跟选项的参数，例如，ddns-key.example.com. 。密钥名必须是合法的域名，由字母，数字，连字符和点组成。

-q（仅 ddns-confgen 。）安静模式：只打印密钥，没有解释的文本或用法举例；这与 tsig-keygen 基本相同。

-s name（仅 ddns-confgen 。）给一个允许动态更新的单一主机名生成配置例子。例子 named.conf 文本显示了如何使用“name”名字类型为指定的名字设置一个更新策略。缺省的密钥名字是

ddns-key.name。注意“self”名字类型不再使用，因为要被更新的名字可能与密钥名不同。这个选项不能与 -z 选项同时使用。

-z zone（仅 ddns-confgen 。）给一个允许动态更新的区生成配置例子。例子 named.conf 文本展示了如何使用”zonesub”名字类型为所指定的 zone 设置一个更新策略，允许更新 zone 内所有子域。这个选项不能与 -s 选项同时使用。

11.23.4 参见

nsupdate(1), named.conf(5), named(8), BIND 9 管理员参考手册。

第 11.24 节 rndc-confgen - rndc 密钥生成工具

11.24.1 概要

rndc-confgen [-a] [-A algorithm] [-b keysize] [-c keyfile] [-h] [-k keyname] [-p port] [-s address] [-t chrootdir] [-u user]

11.24.2 描述

rndc-confgen 为 rndc 生成配置文件。它可以用作一个方便手段，用以手工书写 rndc.conf 文件及在 named.conf 中写相应的 controls 和 key 语句。作为选择，它可以带有 -a 选项运行来建立一个 rndc.key 文件，以避免对 rndc.conf 文件和一个 controls 语句的需求。

11.24.3 参数

-a 做自动的 rndc 配置。这会在 /etc （或其它在编译 BIND 时所指定的 sysconfdir ）建立一个文件 rndc.key ，可以被 rndc 和 named 两个在启动时读取。rndc.key 文件定义了一个缺省的命令通道和认证密钥，它允许 rndc 与本机上的 named 通信而不需要更多的配置。

运行 rndc-confgen -a 允许 BIND 9 和 rndc 作为 BIND 8 和 ndc 的简易替代，而不对现存的 BIND 8 的 named.conf 做任何改变。

如果需要一个比 rndc-confgen -a 所生成的更加复杂的配置，例如，如果 rndc 需要远程使用，你应该不使用 -a 选项运行 rndc-confgen 的并按指示设置 rndc.conf 和 named.conf 。

-A algorithm 指定用于 TSIG 密钥的算法。可用的选择有：hmac-md5，hmac-sha1，hmac-sha224，hmac-sha256,hmac-sha384 和 hmac-sha512。缺省是 hmac-sha256。

-b keysize 指定认证密钥的位数。必须在 1 到 512 位之间；缺省是散列的大小。

-c keyfile 与 -a 选项一起使用指定一个 rndc.key 的替代位置。

-h 打印 rndc-confgen 的选项和参数的简短摘要。

-k keyname 指定 rndc 认证密钥的密钥名。这个必须是一个有效域名。缺省是 rndc-key 。

-p port 指定 named 监听 rndc 连接的命令通道的端口。缺省是 953。

-s address 指定 named 监听 rndc 连接的命令通道的 IP 地址。缺省是环回地址 127.0.0.1。

-t chrootdir 与 -a 选项一起使用，指定 named 运行改变根的目录。一个附加的 rndc.key 拷贝会写到相对于这个目录的位置，这样改变了根的 named 才能找到它。

-u user 与 -a 选项一起使用，设置所生成的 rndc.key 文件的拥有者。如果也指定了 -t ，只有改变了根的目录下的文件才改变其拥有者。

11.24.4 例子

允许不用手工配置而使用 rndc ，运行

rndc-confgen -a

要打印一个例子 rndc.conf 文件和相对应的用于手工插入 named.conf 的 controls 和 key 语句，运行

rndc-confgen

11.24.5 参见

rndc(8), rndc.conf(5), named(8), BIND 9 管理员参考手册。

第 11.25 节 delv - DNS 查找和验证工具

11.25.1 概要

delv [@server] [[-4] | [-6]] [-a anchor-file] [-b address] [-c class] [-d level] [-i] [-m] [-p port#] [-q name] [-t type] [-x addr] [name] [type] [class] [queryopt…]

delv [-h]

delv [-v]

delv [queryopt…] [query…]

11.25.2 描述

delv 是一个发送 DNS 请求并验证结果的工具，它使用与 named 同样的内部解析器和验证器逻辑。

delv 将所有需要获取的请求发向一个指定的名字服务器并验证请求到的数据；这包含原始请求，跟随 CNAME 或 DNAME 链的后续请求，以及为建立一个用于 DNSSEC 验证的信任链的对 DNSKEY 和 DS 记录的请求。它不执行迭代解析，但是模仿一个配置为 DNSSEC 验证和转发的名字服务器的行为。

缺省时，响应使用内置的根区（”.”）DNSSEC 信任锚点进行验证。delv 返回的记录要么是完全验证的，要么是没有签名的。如果验证失败，输出中会包含一个对失败的解释；验证过程可被详细跟踪。由于 delv 不依赖一个外部服务器来执行验证，它可以用于本地名字服务器不可信的环境中来检查 DNS 响应的有效性。

除非其被告知去请求一个特定的名字服务器，delv 将试探 /etc/resolv.conf 中列出的每个服务器。如果没有发现可用的服务器地址，delv 将发请求到环回地址（IPv4 为 127.0.0.1，IPv6 为::1）。

当没有给出命令行参数或选项时，delv 将执行对”.”（根区）的 NS 查询。

11.25.3 简单用法

一个典型的 delv 调用看起来像：

```
delv @server name type
```

其中:

server 是要请求的名字服务器的名字或 IP 地址。这可以是一个点分十进制表示的 IPv4 地址或一个冒号分隔表示的 IPv6 地址。当所提供的 server 参数是一个主机名时，delv 在请求那个名字服务器之前先解析那个名字（注意，那个初始查询 **不**被 DNSSEC 验证）。

如果没有提供 server 参数，delv 会查找 /etc/resolv.conf ；如果在其中发现一个地址，它会请求那个地址上的名字服务器。如果使用了 -4 或 -6 选项，则只有相关传输协议的地址才会被试探。如果没有发现可用的地址，delv 将向本地地址（对 IPv4 为 127.0.0.1，对 IPv6 为::1）发送请求。

name 是被查找的域名。

type 指明需要请求那种类型——ANY，A，MX，等。type 可以是任何有效的请求类型。如果没有提供 type 参数，delv 将执行一个对 A 记录的查找。

11.25.4 选项

-a anchor-file 指定从中读取 DNSSEC 信任锚点的文件。缺省为 /etc/bind.keys ，它被包含在 BIND 9 中，并含有一个或多个根区（”.”）的信任锚点。

与根区名字不匹配的密钥会被忽略；一个替换的密钥名可以使用 +root=NAME 选项指定。

注意：在读取信任锚点文件时，delv 同等对待 trust-anchors ，initial-key 和 static-key 。即，对于一个被管理的密钥，受信任的是 **初始的**密钥，RFC 5011 密钥管理是不支持的。delv 不会询问由 named 维护的被管理密钥数据库。这意谓着，如果 /etc/bind.keys 中有一个密钥被撤销和轮转，都有必要更新 /etc/bind.keys 以便在 delv 中使用 DNSSEC 验证。

-b address 设置请求的源 IP 地址为 address 。这必须是一个主机网络接口上的有效地址，或者”0.0.0.0”，或者”::”。可以通过附加”#<port>”指定一个可选的源端口。

-c class 为请求数据设置请求类。当前，在 delv 中只支持类”IN”，任何其它值将被忽略。

-d level 设置系统范围的调试级别为 level 。允许的范围为 0 到 99。缺省是 0 （关闭调试）。调试级别越高，从 delv 调试跟踪得到的信息越多。参见下列 +mtrace ， +rtrace 和 +vtrace 选项以获得关于调试的详细信息。

-h 显示 delv 使用帮助并退出。

-i 非安全模式。这个关闭内部 DNSSEC 验证。（注意，这不会在向上游查询时设置 CD 位。如果被查询的服务器正在执行 DNSSEC 验证，它将会返回无效数据；这导致 delv 超时。当必须检查无效数据以调试一个 DNSSEC 问题时，使用 dig +cd 。）

-m 打开内存使用调试。

-p port# 指定一个用于请求的目的端口，而不是使用标准的 DNS 端口号 53。这个选项用于同一个被配置为在一个非标准端口号监听请求的名字服务器通信时。

-q name 设置请求名为 name 。虽然指定请求名可以不需要使用 -q ，但是某些时候必须使用这个选项来将请求名与类型和类区别开来（例如，在查找名字”ns”时，可能被错误解释为类型 NS，或者查找”ch”，可能被错误解释为类 CH）。

-t type 设置请求类型为 type ，它可以是除区传送类型 AXFR 和 IXFR 之外 BIND 9 所支持的任何有效类型。与 -q 一样，当查询名称类型或类有二义性时，这有助于区分它们。在某些时候必须将名字从类型中区别出来。

缺省请求类型是”A”，除非提供了 -x 指定一个反向查找，这种情况类型是”PTR”。

-v 打印 delv 版本并退出。

-x addr 执行一个反向查找，映射一个地址到一个名字。addr 是一个点分十进制表示的 IPv4 地址，

或者一个冒号分隔的 IPv6 地址。当使用了 -x ，不需要提供 name 或 type 参数。delv 自动执行对一个类似 11.12.13.10.in-addr.arpa 的名字的查找，并设置请求类型为 PTR。IPv6 地址是以半字节格式在 IP6.ARPA 域下查找。

-4 强制 delv 使用 IPv4。

-6 强制 delv 使用 IPv6。

11.25.5 请求选项

delv 提供一些请求选项，它们影响结果的显示方式，在某些情况它们也影响请求执行的方式。

每个请求由一个加号（+）引导的关键字所标识。一些关键字设置或清除一个选项。这些可以由前导的 no 字符串反转关键字的含义。其它关键字给选项赋值，如超时间隔。它们具有 +keyword=value 的形式。请求选项为：

+[no]cdflag 控制是否在由 delv 发出的请求中设置 CD（checking disabled，关闭验证）位。这个可以用于从一个验证解析器后端进行 DNSSEC 问题排查。一个验证解析器将阻塞无效响应，就使获取它们进行分析变得很困难。在请求中设置 CD 标志将使解析器返回无效响应，delv 可以在内部验证并详细报告错误。

+[no]class 控制在打印一个记录时是否显示类。缺省是显示类。

+[no]ttl 控制在打印一个记录时是否显示 TTL。缺省是显示 TTL。

+[no]rtrace 切换解析器取动作的日志。这报告了在执行解析和验证过程中每个由 delv 发送的请求的名字和类型：这包含了原始请求和跟随 CNAME 记录和为 DNSSEC 验证建立信任链的随后请求。

这和在”resolver”日志类别中设置调试级别为 1 是等效的。使用 -d 选项在系统范围设置调试级别为 1 会得到同样的输出（但是也会影响其它日志类别）。

+[no]mtrace 切换消息日志。这产生 delv 在执行解析和验证过程中收到的响应的详细导出结果。

这和在”resolver”日志类别的”packets”模块中设置调试级别为 10 是等效的。使用 -d 选项在系统范围设置调试级别为 10 会得到同样的输出（但是也会影响其它日志类别）。

+[no]vtrace 切换验证日志。这显示验证器的内部进程，它决定一个答复是否是有效签名、未签名或者无效的。

这和在”dnssec”日志类别的”validator”模块中设置调试级别为 3 是等效的。使用 -d 选项在系统范围设置调试级别为 3 会得到同样的输出（但是也会影响其它日志类别）。

+[no]short 提供一个简洁的回答。缺省是以冗长形式输出回答。

+[no]comments 切换在输出中显示注释。缺省是打印注释。

+[no]rrcomments 切换对输出中每个记录注释的显示状态（例如，关于 DNSKEY 的人可读的密钥信息）。缺省是打印每个记录的注释。

+[no]crypto 切换 DNSSEC 记录中加密字段的显示。这些字段的内容对于调试大多数 DNSSEC 验证失败不是必须的，并且去掉它们会使查看通常的失败更容易。缺省是显示这些字段。如果省略，它们被字符串" [omitted]" 所替代，或者在 DNSKEY 的情况下，作为替代，显示密钥的 id，例如 "[key id = value]"。

+[no]trust 控制在打印一个记录时是否显示信任级别。缺省是显示信任级别。

+[no]split[=W] 分割资源记录中的长的 hex-或 base64-格式的字段为 W 个字符大小的块。（这里 W 是最接近的 4 的倍数）。+nosplit 或 +split=0 使字段完全不被分割。缺省是 56 个字符，或者在打开多行模式时为 44 个字符。

+[no]all 设置或清除显示选项 +[no]comments ， +[no]rrcomments 和 +[no]trust 作为一个组。

+[no]multiline 以冗长多行格式并附带人可读的注释打印长记录（诸如 RRSIG，DNSKEY 和 SOA 记录）。缺省是将每条记录打印在一行上，以便 delv 的输出更容易被机器分析。

+[no]dnssec 指示是否在 delv 的输出中显示 RRSIG 记录。缺省是显示。注意（与 dig 不同）这不控制是否请求 DNSSEC 记录或者是否验证它们。总是请求 DNSSEC 记录，并总是进行验证，除非使用 -i 或 +noroot 禁止。

+[no]root[=ROOT] 指示是否执行传统的 DNSSEC 验证，如果是，指定信任锚点的名字。缺省是使用一个 "." （根区）的信任锚点，对此有一个内置密钥。如果指定一个不同的信任锚点，必须使用 -a 指定一个包含这个密钥的文件。

+[no]tcp 控制在发送请求时是否使用 TCP。缺省是使用 UDP，除非收到一个被截断的响应。

+[no]unknownformat 以未知 RR 类型表示格式（RFC 3597 ）打印所有 RDATA。缺省是以类型的表示格式打印已知类型的 RDATA。

+[no]yaml 以 YAML 格式打印响应数据。

11.25.6 文件

/etc/bind.keys

/etc/resolv.conf

11.25.7 参见

dig(1), named(8), RFC 4034, RFC 4035, RFC 4431, RFC 5074, RFC 5155.

第 11.26 节 nsupdate - 动态 DNS 更新工具

11.26.1 概要

nsupdate [-d] [-D] [-i] [-L level] [[-g] | [-o] | [-l] | [-y [hmac:]keyname:secret] | [-k keyfile]] [-t timeout] [-u udptimeout] [-r udpretries] [-v] [-T] [-P] [-V] [[-4] | [-6]] [filename]

11.26.2 描述

nsupdate 是用于提交在 RFC 2136 中所定义的动态 DNS 更新请求给一个名字服务器。这允许在不用手工编辑区文件的情况下增加或删除一个区的资源记录。一个更新请求可以包含增加或删除多个资源记录的请求。

在由 nsupdate 进行动态控制之下的区或者一个 DHCP 服务器不应该由手工编辑。手工编辑可能与动态更新相冲突并导致数据丢失。

使用 nsupdate 动态增加或删除的资源记录必须在同一个区内。请求发给区的主服务器。这由区的 SOA 记录的 MNAME 字段来标识。

事务签名可以被用于认证动态 DNS 更新。这些使用在 RFC 2845 中所描述的 TSIG 资源记录或者在 RFC 2535 和 RFC 2931 中所描述的 SIG(0) 记录或者在 RFC 3645 中所描述的 GSS-TSIG。

TSIG 依赖于一个仅有 nsupdate 和名字服务器所知道一个共享密钥。例如，将合适的 key 和 server 语句添加到 /etc/named.conf 中。这样名字服务器就可以将合适的密钥和算法与将使用 TSIG 认证的客户端应用程序的 IP 地址相关联。你可以使用 ddns-confgen 生成合适的配置片段。nsupdate 使用 -y 或 -k 选项提供 TSIG 共享密码。这些选项是互斥的。

SIG(0) 使用公钥加密算法。要使用一个 SIG(0) 密钥，公钥必须存放在名字服务器所服务的区的一个 KEY 记录中。

GSS-TSIG 使用 Kerberos 凭证。标准的 GSS-TSIG 模式使用 -g 标志打开。Windows 2000 所使用的一个非标准兼容的 GSS-TSIG 变体可以用 -o 标志打开。

11.26.3 选项

-4 只使用 IPv4。

-6 只使用 IPv6。

-d 调试模式。它提供关于所生成的更新请求和从名字服务器收到的回复的跟踪信息。

-D 扩展调试模式。

-i 强制交互模式，即使标准输入不是一个终端。

-k keyfile 这个文件存放 TSIG 认证密钥。密钥文件可以有两种格式：一个包含一个 named.conf-格式的 key 语句的文件，它可以由 ddns-confgen 自动生成，或者一对文件，其文件名格式是 K{name}.+157.+{random}.private 和 K{name}.+157.+{random}.key ，它们可以由 dnssec-keygen 生成。-k 也可以用于指定一个用于认证动态 DNS 更新请求的 SIG(0) 密钥。在这个情况下，所指定的密钥不是一个 HMAC-MD5 密钥。

-l 只 local-host 模式，这将会把服务器地址设置为 localhost（关闭 server，这样服务器地址就不能被覆盖）。到本地服务器的连接将使用在 /var/run/named/session.key 中找到的一个 TSIG 密钥，如果有任何本地主区的 update-policy 设置为 local ，这个密钥由 named 自动生成。这个密钥文件的位置可以使用 -k 选项覆盖。

-L level 设置日志的调试级别。如果为 0，就关掉日志。

-p port 设置用于连接一个名字服务器的缺省端口。缺省为 53。

-P 打印输出私有的 BIND 特定资源记录类型的列表，这些资源记录类型的格式是 nsupdate 所能理解的。参见 -T 选项。

-r udpretries UDP 重试次数。缺省是 3。如果为 0，仅仅会生成一次更新请求。

-t timeout 一个更新请求在其被中断之前可以持续的最大时间。缺省是 300 秒。0 可以用来关掉超时。

-T 打印输出 IANA 标准资源记录类型的列表，这些资源记录类型的格式是 nsupdate 所能理解的。nsupdate 将在打印列表后退出。-T 选项可以和 -P 选项组合。

可以使用 “TYPEXXXXX” 输入其它类型，其中 “XXXXX” 是不以 0 开始的十进制数值。如果出现了 rdata，将会使用 UNKNOWN rdata 格式分析，（<backslash> <hash> <space> <length> <space> <hexstring>）。

-u udptimeout UDP 重试间隔。缺省是 3 秒。如果为 0，这个间隔将会从超时间隔和 UDP 重试次数中计算得到。

-v 即使对小的更新请求也使用 TCP。缺省时，nsupdate 使用 UDP 发送更新请求给名字服务器，除非它们太大不能装进一个 UDP 请求中，这种情况将使用 TCP。当有一批更新请求时，TCP 可能是更优的。

-V 打印版本号并退出。

-y [hmac:]keyname:secret 字面的 TSIG 认证密钥。keyname 是密钥的名字，而 secret 是 base64 编码的共享密钥。hmac 是密钥算法名；有效的选择为 hmac-md5，hmac-sha1，hmac-sha224，hmac-sha256，hmac-sha384 或 hmac-sha512。如果未指定 hmac，缺省是 hmac-md5，或者如果 MD5 被禁止，则是 hmac-sha256。

注意：不鼓励使用 -y 选项，因为共享密钥是以明文形式作为命令行参数提供的。在 ps1 的输出或者在用户的 shell 所维护的历史文件中，这个可能是可见的。

11.26.4 输入格式

nsupdate 从 filename 或标准输入读取输入。每个命令刚好在一个输入行内。一些命令是出于管理的目的。其它的命令要么是更新指令，要么是检查区内容的先决条件。这些检查设置条件，即一些名字或资源记录集要么存在，要么不存在于区中。如果要让整个更新请求成功，这些条件必须被满足。如果对先决条件的测试失败，更新将被拒绝。

每个更新请求由 0 个或多个先决条件以及 0 个或多个更新所组成。如果某些指定的资源记录出现或不出现在区中，这允许一个合适的经过认证的更新请求进行处理。一个空输入行（或 send 命令）导致所有累积的命令被作为一个动态 DNS 更新请求发送给名字服务器。

命令格式及其含义如下：

server servername port 发送所有更新请求给名字服务器 servername。当没有提供 server 语句时，nsupdate 将发送更新请求给正确的区的主服务器。这个区的 SOA 记录中的 MNAME 字段将会标识这个区的主服务器。port 是接收动态更新请求的 servername 上的端口号。如果没有指定端口号，就使用缺省的 DNS 端口号 53。

local address port 使用本地 address 发送所有动态更新请求。当没有提供 local 语句时，nsupdate 将使用系统所选择的一个地址和端口发送更新。port 还可以用在使请求来自一个指定的端口。如果没有指定端口号，系统将会分配一个。

zone zonename 指定所有的更新都发生在区 zonename 上。如果没有提供 zone 语句，nsupdate 会试图基于其余的输入来决定正确的区。

class classname 指定缺省类。如果没有指定 class，缺省类是 IN。

ttl seconds 指定要添加记录的缺省生存期。值 none 将清除缺省生存期。

key hmac:keyname secret 指定所有的更新都用 keyname secret 对进行 TSIG 签名。如果指定了 hmac，它将设置签名使用的算法；缺省是 hmac-md5，或者如果 MD5 被禁止，则是 hmac-sha256。key 命令覆盖任何在命令行由 -y 或 -k 所指定的密钥。

gsstsig 使用 GSS-TSIG 对更新签名。这个等效于在命令行指定 -g。

oldgsstsig 使用 Windows 2000 版的 GSS-TSIG 对更新签名。这个等效于在命令行指定 -o。

realm [realm_name] 在使用 GSS-TSIG 时，用 realm_name 而不是 krb5.conf 中的缺省 realm。如果未指定 realm，则已保存的 realm 将被清除。

check-names [yes_or_no] 在增加记录时打开或者关闭 check-names 处理。check-names 对被删除的先决条件或记录没有影响。缺省时 check-names 处理是打开的。如果 check-names 处理失败，记录将不会被添加到 UPDATE 消息中。

prereq nxdomain domain-name 要求名字 domain-name 没有存在任何类型的资源记录。

prereq yxdomain domain-name 要求 domain-name 存在（至少有一个资源记录，可以是任何类型）。

prereq nxrrset domain-name class type 要求指定的 type，class 和 domain-name 不存在任何资源记录。如果省略 class，就假定为 IN（internet）。

prereq yxrrset domain-name class type 这个要求指定的 type，class 和 domain-name 必须存在一个资源记录。如果省略 class，就假定为 IN（internet）。

prereq yxrrset domain-name class type data 来自每个这种形式的先决条件集合的 data 共享一个共同的 type，class 和 domain-name，并被组合成一个资源记录集合的形式。这个资源记录集合必须精确地匹配区中以 type，class 和 domain-name 给出的已存在的资源记录集合。data 以资源记录 RDATA 的标准文本表示方法书写。

update delete domain-name ttl class type data 删除名为 domain-name 的任何资源记录。如果提供了 type 和 data，只有匹配的资源记录会被删除。如果没有提供 class，就假设是 internet 类。ttl 被忽略，仅为了兼容性而允许之。

update add domain-name ttl class type data 使用指定的 ttl，class 和 data 增添一个新的资源记录。

show 显示当前消息，包含自上次发送以来所指定的所有先决条件和更新。

send 发送当前消息。这等效于输入一个空行。

answer 显示回答。

debug 打开调试。

version 打印版本号。

help 打印命令表。

以分号开始的行为注释，将被忽略。

11.26.5 例子

下面的例子显示 nsupdate 如何被用于对 example.com 区插入和删除资源记录。注意每个例子中的输入包含一个结尾的空行，这样就将一组命令作为一个动态更新请求发送给 example.com 的主名字服务器。

```
# nsupdate
> update delete oldhost.example.com A
> update add newhost.example.com 86400 A 172.16.1.1
> send
```

oldhost.example.com 的任何 A 记录被删除。newhost.example.com 的一个带有 IP 地址 172.16.1.1 的 A 记录被添加。新添加的记录具有一个 1 天的 TTL（86400 秒）。

```
# nsupdate
> prereq nxdomain nickname.example.com
> update add nickname.example.com 86400 CNAME somehost.example.com
> send
```

先决条件是让名字服务器检查没有 nickname.example.com 的任何类型的资源记录。如果有，更新请求失败。如果这个名字不存在，就为它添加一个 CNAME。这就确保了在添加 CNAME 时，不会与 RFC 1034 中的长标准规则相冲突，即如果一个名字存在一个 CNAME，就必须不能存在其它任何记录类型。（这个规则在 RFC 2535 中为 DNSSEC 而被更新，以允许 CNAME 可以有 RRSIG, DNSKEY 和 NSEC 记录。）

11.26.6 文件

/etc/resolv.conf 用于标识缺省的名字服务器。

/var/run/named/session.key 设置用于 local-only 模式的缺省 TSIG 密钥。

K{name}.+157.+{random}.key 由 dnssec-keygen8 所创建的 HMAC-MD5 密钥的 base-64 编码。

K{name}.+157.+{random}.private 由 dnssec-keygen8 所创建的 HMAC-MD5 密钥的 base-64 编码。

11.26.7 参见

RFC 2136, RFC 3007, RFC 2104, RFC 2845, RFC 1034, RFC 2535, RFC 2931, named(8), ddns-confgen(8), dnssec-keygen(8).

11.26.8 缺陷

TSIG 密钥是冗余存放在两个分离的文件中。这是 nsupdate 为其加密操作使用 DST 库的一个后果，在将来的版本中可能会变化。

第 11.27 节 host - DNS 查找工具

11.27.1 概要

host [-aACdlnrsTUwv] [-c class] [-N ndots] [-p port] [-R number] [-t type] [-W wait] [-m flag] [[-4] | [-6]] [-v] [-V] {name} [server]

11.27.2 描述

host 是一个进行 DNS 查找的简单工具。它通常用于转换名字到 IP 地址或相反的操作。在没有给出参数或选项时，host 打印出其命令行参数和选项的简短摘要。

name 是要查找的域名。它也可以是一个点分十进制的 IPv4 地址或者一个冒号分隔的 IPv6 地址，在这种情况下 host 缺省将会对那个地址执行一个反向查找。server 是一个可选参数，可以是名字服务器的名字或 IP 地址，host 应该查询这个服务器，而不是 /etc/resolv.conf 中的服务器或服务器列表。

11.27.3 选项

-4 仅使用 IPv4 传输请求。参见 -6 选项。

-6 仅使用 IPv6 传输请求。参见 -4 选项。

-a “全部”。-a 选项通常等效于 -v -t ANY 。它也影响 -l 列出区名单选项的行为。

-A “几乎全部”。-A 等效于 -a ，除了 RRSIG，NSEC 和 NSEC3 记录在输出时被省略之外。

-c class 请求类：这个可以用于查找 HS（Hesiod）或 CH（Chaosnet）类的资源记录。缺省类是 IN（Internet）。

-C 检查一致性：host 将向从区 name 的所有列出的权威服务器请求 SOA 记录。列出的名字服务器是由区中能找到的 NS 记录定义的。

-d 打印调试跟踪信息。等效于 -v 明细选项。

-l 对区列表：host 对区 name 执行一个区传送，并打印出 NS，PTR 和地址记录（A/AAAA）。

同时，-l -a 选项打印区中全部记录。

-N ndots 出现在被当做完整名字的 name 中的点的数目。缺省值是在 /etc/resolv.conf 中用 ndots 语句定义的值，或者为 1，如果没有使用 ndots 语句。少于这个数目的点的名字会被解释为相对名字，并在 /etc/resolv.conf 的 search 或 domain 指令所列的域名中搜索。

-p port 指定请求去往的服务器的端口。缺省为 53。

-r 非递归请求：设置这个选项清除请求中的 RD（递归期望）位。这意谓着名字服务器在收到这个请求后不会试图去解析 name 。-r 选项使 host 能够模仿一个名字服务器的行为，通过生成非递归请求并期望接收这些请求的回答，这些回答可以是对其它名字服务器指向。

-R number UDP 请求的重试次数：如果 number 是负数或零，重试次数将缺省为 1. 缺省值是 1，或者 /etc/resolv.conf 中 attempts 选项的值，如果设置了这个值。

-s 如果任何服务器响应了一个 SERVFAIL，不发送请求到下一个名字服务器，这与普通的存根解析器行为相反。

-t type 请求类型：type 参数可以是任何可识别的请求类型：CNAME, NS, SOA, TXT, DNSKEY, AXFR 等等。

没有指定请求类型时，host 自动选择一个合适的请求类型。缺省情况，它查找 A，AAAA 和 MX 记录。如果给出 -C 选项，请求将查找 SOA 记录，如果 name 是一个点分十进制 IPv4 地址或冒号分隔的 IPv6 地址，host 将查找 PTR 记录。

如果选择一个 IXFR 请求类型，可以通过附加一个等号和开始序列号来指定开始序列号（类似 -t IXFR=12345678 ）。

-T; -U TCP/UDP：缺省时，host 在生成请求时使用 UDP。-T 选项使其在请求一个名字服务器时使用一个 TCP 连接。对需要的请求，将会自动选择 TCP，例如区传送（AXFR）请求。类型为 ANY 的请求缺省走 TCP，但是可以通过使用 -U 强制使用 UDP。

-m flag 内存使用调试：标志可以为 record ， usage 或 trace 。你可以多次指定 -m 选项以设置多个标志。

-v 明细输出。等效于 -d 调试选项。明细输出也可以在 /etc/resolv.conf 中通过设置 debug 选项打开。

-V 打印版本号并退出。

-w 永远等待：请求超时被设置为最大可能值。参见 -W 选项。

-W wait 超时：等待一个响应最多 wait 秒。如果 wait 小于一，等待间隔被置为一秒。

缺省时，host 将对 UDP 响应等待 5 秒，并对 TCP 连接等待 10 秒。这些缺省值可以被 /etc/resolv.conf 中的 timeout 选项所覆盖。

参见 -w 选项。

11.27.4 IDN 支持

如果编译 host 时带有 IDN（internationalized domain name，国际化域名）支持，它可以接受和显示非 ASCII 域名。host 会在发送一个请求到 DNS 服务器或显示一个来自服务器的回复之前正确地转换域名的字符编码。如果由于某种原因你想关闭 IDN 支持，就定义 IDN_DISABLE 环境变量。在 host 运行时，如果变量已设置，IDN 支持就是关闭的。

11.27.5 文件

/etc/resolv.conf

11.27.6 参见

dig(1), named(8).

第 11.28 节 dig - DNS 查找工具

11.28.1 概要

dig [@server] [-b address] [-c class] [-f filename] [-k filename] [-m] [-p port#] [-q name] [-t type] [-v] [-x addr] [-y [hmac:]name:key] [[-4] | [-6]] [name] [type] [class] [queryopt…]

dig [-h]

dig [global-queryopt…] [query…]

11.28.2 描述

dig 是一个查询 DNS 名字服务器的灵活工具。它执行 DNS 查找并显示从所查找的名字服务器所返回的答案。由于其灵活性，容易使用和整洁的输出，大多数 DNS 管理员使用 dig 来排除 DNS 问题。dig 趋向于比其它查找工具提供更多的功能。

虽然 dig 通常使用命令行参数，它也具有批处理模式的操作，从一个文件读入查找请求。在使用 -h 选项时，会打印出其命令行参数的一个简要总结。与早期的版本不同，dig 的 BIND 9 实现允许从命令行发出多个查找。

dig 将会试探 /etc/resolv.conf 中服务器列表中的每台机器，除非让它查找一个指定的名字服务器。如果没有找到可用的服务器地址，dig 将会把请求发给本地主机。

在没有给出命令行参数或选项时，dig 将会执行一个对“.”（根）的 NS 请求。

通过 ${HOME}/.digrc 文件，可以为每个用户设置 dig 的缺省参数。这个文件将被读入并在命令行参数之前应用其中的所有参数。-r 选项为那些需要可预测结果的脚本关闭这个特性。

IN 和 CH 类名覆盖 IN 和 CH 顶级域名。使用 -t 和 -c 选项指定类型和类，或者使用 -q 指定域名，或者在查找这些顶级域时使用“IN.”和“CH.”。

11.28.3 简单用法

一个典型的 dig 调用看起来是这样的：

```
dig @server name type
```

在这里：

server 是请求发往的名字服务器的名字或者 IP 地址。可以是点分十进制格式的 IPv4 地址或者冒号分隔形式的 IPv6 地址。当所提供的 server 参数是一个主机名，dig 在请求这个名字服务器之前先解析其名字。

如果没有提供 server 参数，dig 查找 /etc/resolv.conf ；如果在其中发现一个地址，它就请求这个地址上的名字服务器。如果使用了 -4 或 -6 选项，就只会试探相关的传输层。如果没有找到可用的地址，dig 就会把请求发到本地主机。显示从名字服务器返回的响应信息。

name 是要查找的资源记录的名字。

type 指明所要的请求类型——ANY，A，MX，SIG，等等。type 可以是任何有效的请求类型。如果没有提供 type 参数，dig 将会执行对 A 记录的查找。

11.28.4 选项

-4 仅使用 IPv4。

-6 仅使用 IPv6。

-b address[#port] 设置请求的源 IP 地址。address 必须是主机的一个网络接口上的有效地址，或者为“0.0.0.0”，或者为“::”。可以通过附加“#<port>”指定一个可选的端口。

-c class 设置请求类。缺省 class 是 IN；其它类是 HS，表示 Hesiod 记录，或 CH，表示 CHAOSNET 记录。

-f file 批处理模式：dig 从文件 file 中读入要查找请求的列表，并进行处理。文件中的每一行应该组织成与使用命令行提供请求给 dig 的同样方式。

-k keyfile 使用 TSIG 签名请求，TSIG 使用一个从给定的文件中读到的一个密钥。密钥文件可以使用 tsig-keygen8 生成。在与 dig 之间使用 TSIG 认证时，被请求的名字服务器需要知道所使用的密钥和算法。在 BIND 中，通过在 named.conf 中指定合适的 key 和 server 语句来完成。

-m 打开内存使用调试。

-p port 发送请求到服务器的一个非标准端口，而不是缺省的 53 端口。这个选项可以用于测试一个名字服务器，将其配置成在一个非标准端口上监听请求。

-q name 要查询的域名。这个用于区别 name 和其它参数。

-r 不从 ${HOME}/.digrc 读取选项。这对需要可预测结果的脚本非常有用。

-t type 请求的资源记录类型。它可以是任何有效的请求类型。如果它是 BIND 9 中所支持的资源记录类型，它可以通过类型助记符（如‘NS’或‘AAAA’）给出。缺省请求类型为“A”，除非设定 -x 选项，它指定一个反向查找。可以通过指定 AXFR 的类型的请求进行区传送。当请求一个增量区传送（IXFR）时，type 被设为 ixfr=N 。增量区传送将包含区的变化，区的 SOA 记录中的序列号为 N 。

所有的资源记录类型都可以表示为“TYPEnn”，这里“nn”是类型编号。如果资源类型是 BIND 9 中所不支持的，结果将会以 RFC 3597 中描述的方式显示。

-u 输出以微秒为单位而不是以毫秒为单位的请求时间。

-v 打印出版本号并退出。

-x addr 简化的反向查找，用于从地址映射到名字。addr 是一个点分十进制形式的 IPv4 地址，或者一个以冒号分隔的 IPv6 地址。当使用 -x 时，不需要提供 name ， class 和 type 参数。dig

自动执行一个类似 94.2.0.192.in-addr.arpa 的查找，并将请求类型和类分别设置为 PTR 和 IN。IPv6 地址使用半字节格式在 IP6.ARPA 域名下面查找。

-y [hmac:]keyname:secret 使用 TSIG 并所给定的认证密钥签名请求。keyname 是密钥的名字，secret 是 base64 编码的共享密码，hmac 是密钥算法的名字；有效的选择是 hmac-md5 ，hmac-sha1 ，hmac-sha224 ，hmac-sha256 ，hmac-sha384 或 hmac-sha512 。如果未指定 hmac ，缺省为 hmac-md5 或者如果 MD5 被禁止，则为 hmac-sha256 。

注解：你应该使用 -k 选项并避免 -y 选项，因为随着 -y 被提供的共享密码是以明文形式被用作一个命令行参数中。这在 ps1 的输出中，或在用户的 shell 中维护的一个历史文件中是可见的。

11.28.5 请求选项

dig 提供许多查询选项，可以影响生成查询和显示结果的方式。其中一些选项设置或清空请求头部的标志位，一些决定打印回答中的哪些部份，其它的决定超时和重试策略。

每个请求选项由一个前导加号（+ ）和一个关键字标识。一些关键字设置或清空一个选项。这些可能由前导字符串 no 来否定关键字的含义。其它关键字给选项赋值，就像超时间隔。他们具有 +keyword=value 的形式。关键字可以是缩写，前提是缩写是无歧义的；例如 +cd 等效于 +cdflag 。请求选项是：

+[no]aaflag +[no]aaonly 的同义词。

+[no]aaonly 在请求中设置“aa”标志。

+[no]additional 显示 [不显示] 回复的附加部份。缺省是显示。

+[no]adflag 设置 [不设置] 请求中的 AD（可靠的数据）位。它要求服务器返回回答和权威部份的所有记录是否都已按照服务器的安全策略验证。AD=1 指示所有记录都已被验证为安全并且回答不是来自于一个 OPT-OUT 范围。AD=0 指示回答中的某些部份是不安全的或者没有验证的。这个位缺省是置位的。

+[no]all 设置或清除所有显示标志。

+[no]answer 显示 [不显示] 回复的回答部份。缺省是显示。

+[no]authority 显示 [不显示] 回复的权威部份。缺省是显示。

+[no]badcookie 如果收到一个 BADCOOKIE 响应，使用新的服务器 cookie 重试查找。

+[no]besteffort 试图显示坏包消息的内容。缺省是不显示坏包回答。

+bufsize=B 这个选项设置使用 EDNS0 公告的 UDP 消息缓冲大小为 B 字节。这个缓冲的最大值和最小值分别为 65535 和 0。+bufsize=0 关闭 EDNS（使用 +bufsize=0 +edns 发送一个带有 0 字节公告大小的 EDNS 消息）。+bufsize 恢复缺省的缓存大小。

+[no]cdflag 设置 [不设置] 请求中的 CD（关闭检查）位。这请求服务器不对响应执行 DNSSEC 验证。

+[no]class 打印记录时显示 [不显示] 类。

+[no]cmd 切换在输出中对初始注释的打印，它标识 dig 的版本和应用的请求选项。这个选项总是具有全局效果；它不能被全局设置并被一个基于每个查询所覆盖。缺省时打印这个注释。

+[no]comments 切换在输出中对某些注释行的显示，包含关于包头部和 OPT 伪部份的信息，以及响应部份的名字。缺省是打印这些注释。

输出中其它类型的注释不受这个选项的影响，但可以使用其它命令行选项进行控制。这些选项包括 +[no]cmd ， +[no]question ， +[no]stats 和 +[no]rrcomments 。

+[no]cookie=#### 带可选值发送一个 COOKIE EDNS 选项。从先前的响应重放一个 COOKIE 将允许服务器标识一个先前的客户端。缺省值是 +cookie 。

当设置了 +trace 时，也设置 +cookie ，这样能更好地模拟来自一个名字服务器的缺省请求。

+[no]crypto 切换对 DNSSEC 记录中加密字段的显示。这些字段在诊断大多数 DNSSEC 验证失败时不是必须的，去掉它们使得查看普通失败更容易。缺省是显示这些字段。当被省略时，它们被字符串” [omitted]” 替代，或者在 DNSKEY 情况，显示密钥标识号作为替代，例如” [key id = value]”。

+[no]defname 废弃，作为 +[no]search 的同义词对待。

+[no]dnssec 通过在请求的附加部份放置 OPT 记录，并设置 DNSSEC OK 位（DO）来请求发送 DNSSEC 记录。

+domain=somename 设置搜索列表使包含唯一域名 somename ，就像在 /etc/resolv.conf 中 domain 命令中指定一样，如果给出 +search 选项，就打开搜索列表处理。

+dscp=value 在发送请求时，设置使用的 DSCP 码点。有效的 DSCP 码点在 [0..63] 的范围。缺省时不显式设定码点。

+[no]edns[=#] 指定请求所带的 EDNS 的版本。有效值为 0 到 255。设置 EDNS 版本会导致发出一个 EDNS 请求。+noedns 清除所记住的 EDNS 版本。缺省时 EDNS 被设置为 0。

+[no]ednsflags[=#] 设置必须为 0 的 EDNS 标志位（Z 位）为指定的值。十进制，十六进制和八进制都是可以的。设置一个命名标志（例如 DO）将被静默地忽略。缺省时，不设置 Z 位。

+[no]ednsnegotiation 打开/关闭 EDNS 版本协商。缺省时 EDNS 版本协商为打开。

+[no]ednsopt[=code[:value]] 使用码点 code 和可选荷载 value 指定 EDNS 选项为一个十六进制字符串。code 可以为一个 EDNS 选项名（例如，NSID 或 ECS ）或一个任意数字值这两者之一。+noednsopt 清除将发送的 EDNS 选项。

+[no]expire 发送一个 EDNS 过期选项。

+[no]fail 如果收到了一个 SERVFAIL 不会重试下一个服务器。缺省是不重试下一个服务器，这与普通的存根解析器行为相反。

+[no]header-only 发送一个带有 DNS 头部但不带问题部分的请求。缺省是要添加一个问题部分。当设置这个选项时，请求类型和请求名被忽略。

+[no]identify 在 +short 选项打开时，显示 [不显示] 用于补充回答的 IP 地址和端口号。如果要求短格式回答，缺省是不显示提供回答的服务器的源地址和端口号。

+[no]idnin 处理 [不处理] 输入中的 IDN 域名。这个要求在编译时打开 IDN SUPPORT。

当标准输出是一个 tty 时，缺省是要处理 IDN 输入。当 dig 输出被重定向到文件，管道以及其它非 tty 文件描述符时，对 IDN 处理是被禁止的。

+[no]idnout 转换 [不转换] 输出上的 puny code。这要求在编译时打开 IDN 支持。

当标准输出是一个 tty 时，缺省是要处理输出的 puny code。当 dig 输出被重定向到文件，管道以及其它非 tty 文件描述符时，对输出的 puny code 处理是被禁止的。

+[no]ignore 忽略 UDP 响应中的截断而不用 TCP 重试。缺省情况要用 TCP 重试。

+[no]keepalive 发送 [或不发送] 一个 EDNS 保活选项。

+[no]keepopen 在两次或多次请求之间保持 TCP 套接字打开，这样可以重用而不是每次查找时都建立一个新的 TCP 套接字。缺省是 +nokeepopen 。

+[no]mapped 允许使用映射 IPv4 到 IPv6 地址。缺省是 +mapped 。

+[no]multiline 以详细的多行格式并附带人所易读的注释打印如 SOA 这样的记录。缺省是将每个记录打印在一行中，以适应机器分析 dig 的输出。

+ndots=D 设置在 name 中必须出现的点的数目为 D 以使其被当成绝对名字。缺省值是在 /etc/resolv.conf 中用 ndots 语句定义的值，或者为 1，如果没有使用 ndots 语句。少于这个数目的点的名字会被解释为相对名字，如果设置了 +search , 就会在 /etc/resolv.conf 中的 search 或 domain 指令所列的域名中搜索。

+[no]nsid 在发送一个请求时包含一个 EDNS 名字服务器 ID 请求。

+[no]nssearch 在设置了这个选项时，dig 试图找到包含所查找名字的区的权威名字服务器并显示这个区的每个名字服务器都有的 SOA 记录。没有响应的服务器的地址也会被打印。

+[no]onesoa 在执行一个 AXFR 时，仅打印一个（开始的）SOA 记录。缺省是打印开始的和结尾的 SOA 记录。

+[no]opcode=value 设置 [恢复]DNS 消息操作码为指定值。缺省值是 QUERY（0）。

+padding=value 使用 EDNS 填充选项将请求包填充到 value 字节对齐的块。例如，+padding=32 将使一个 48 字节的请求被填充到 64 字节。缺省的块大小为 0，即关闭填充。最大是 512。填充值一般是 2 的幂，例如 128；然而，这不是硬性规定。对填充请求的响应也会被填充，但仅当请求使用 TCP 或者 DNS COOKIE 时。

+[no]qr 切换对所发出的请求消息的显示。缺省情况，不打印请求。

+[no]question 切换当一个回答返回时对一个请求的问题部份的显示。缺省是将问题部份作为一个注释打印。

+[no]raflag 设置 [不设置] 请求中的 RA（Recursion Available，递归可用）位。缺省是 +noraflag。对于请求，这个位应当被服务器忽略。

+[no]rdflag 一个 +[no]recurse 的同义词。

+[no]recurse 切换请求中的 RD（期望递归）位设置。这个位缺省是置位的，意谓着 dig 普通情况是发送递归的请求。在使用了 +nssearch 或 +trace 选项时，递归是自动关闭的。

+retry=T 设置向服务器重新进行 UDP 请求的次数为 T 次，取代缺省的 2 次。与 +tries 不同，这个不包括初始请求。

+[no]rrcomments 切换在输出中显示每记录注释的状态（例如，便于人阅读的关于 DNSKEY 记录的密钥信息）。缺省是不打印记录注释，除非多行模式被激活。

+[no]search 使用 [不使用] 在 resolv.conf（如果存在）中由 searchlist 或者 domain 命令所定义的搜索列表。缺省是不使用搜索列表。

resolv.conf 中的‘ndots’（缺省为 1），可以被 +ndots 覆盖，决定名字是否被当成绝对名字以及是否最终执行一个查找。

+[no]short 提供一个简洁的回答。缺省是以明细形式打印回答。这个选项总是具有全局效果；它不能被全局设置并被一个基于每个查询所覆盖。

+[no]showsearch 执行 [不执行] 立即显示结果的搜索。

+[no]sigchase 这个特性现在被废弃并被去掉了；使用 delv 替代。

+split=W 将资源记录中较长的 hex-或 base64-格式的字段分割为 W 个字符的块（W 被向上取整到距其最近的 4 的倍数上）。+nosplit 或 +split=0 导致字段完全不被分割。缺省为 56 个字符，或者在多行模式时为 44 个字符。

+[no]stats 切换对统计的打印：请求完成的时间，响应的大小等等。缺省行为是在每次查询之后以一个注释打印请求统计。

+[no]subnet=addr[/prefix-length] 发送（不发送）一个 EDNS 客户端子网选项，带有指定的 IP 地址或网络前缀。

dig +subnet=0.0.0.0/0，或简写为 dig +subnet=0，发送一个 EDNS client-subnet 选项，附带一个空地址和一个为 0 的源前缀，它发信号给一个解析器，在解析这个请求时，必须不能使用客户端的地址信息。

+[no]tcflag 在请求中设置 [不设置]TC（TrunCation，截断）位。缺省是 +notcflag。对于请求，这个位应当被服务器忽略。

+[no]tcp 在请求名字服务器时使用 [不使用]TCP。缺省行为是使用 UDP，除非一个类型 any 或者 ixfr=N 的查询被请求，这种情况下缺省是 TCP。AXFR 请求总是使用 TCP。

+timeout=T 设置一个请求的超时为 T 秒。缺省超时是 5 秒。试图将 T 设置成小于 1 将会得到请求超时为 1 秒的结果。

+[no]topdown 这个特性与 dig +sigchase 相关，后者已过时并被去掉了。使用 delv 替代。

+[no]trace 切换对从根名字服务器到要查找名字的授权路径的跟踪状态。缺省是关闭跟踪的。当打开跟踪时，dig 迭代发送请求来解析要查找的名字。它会跟随自根服务器起所给出的参考信息，显示出来自每个解析用到的服务器的回答。

如果指定了 @server，它仅影响根区名字服务区的初始请求。

当设置了 +trace 时，也会设置 +dnssec，来更好地模仿来自某个名字服务器的缺省请求。

+tries=T 设置向服务器进行 UDP 请求的重试次数为 T 次，取代缺省的 3 次。如果 T 小于或等于 0，重试次数就静默地回归为 1。

+trusted-key=#### 和 dig +sigchase 一起使用的之前指定的受信任密钥。这个特性现在已过时并被去掉了；使用 delv 替代。

+[no]ttlid 在打印记录时显示 [不显示]TTL。

+[no]ttlunits 显示 [不显示]TTL，以友好地人可读时间单位 “s”，“m”，“h”，“d” 和 “w”，分别代表秒，分，小时，天和周。隐含为 +ttlid。

+[no]unexpected 接受 [不接受] 来自意外源地址的回答。缺省时，dig 将不会接受一个源地址不是其所请求的地址发来的响应。

+[no]unknownformat 以未知 RR 类型表示格式（RFC 3597）打印所有 RDATA。缺省是以类型的表示格式打印已知类型的 RDATA。

+[no]vc 在请求名字服务器时使用 [不使用]TCP。这是为 +[no]tcp 提供向后兼容性而使用的替换语法。“vc” 表示 “virtual circuit”。

+[no]yaml 以一个详细的 YAML 格式打印响应（并且，如果使用了 <option>+qr</option>，也包括发出的请求）。

+[no]zflag 设置 [不设置] 一个 DNS 请求中最后未赋值的 DNS 头部标志。这个标志缺省是关闭。

11.28.6 多个请求

BIND 9 的 dig 实现支持在命令行（另外还支持 -f 批文件选项）指定多个请求。每个这样的请求可以带有自己的标志、选项和请求选项集合。

在这种情况下，每个 query 参数代表一个上述命令行语法中的单独请求。每个都是由标准选项和标志，待查找名字，可选的请求类型和类以及任何应该应用于这个请求的请求选项所组成。

也可以采用一个请求选项的全局集，它将应用到所有请求上。这些全局请求选项必须在命令行中先于第一个名字、类、类型、选项、标志和请求选项的元组之前。任何全局请求选项（+[no]cmd 和 +[no]short 选项除外）都可以被某个请求专用的请求选项所覆盖。例如：

```
dig +qr www.isc.org any -x 127.0.0.1 isc.org ns +noqr
```

显示怎样在命令行使用 dig 完成三个查找：一个对 www.isc.org 的 ANY 的查找，一个对 127.0.0.1 的反向查找和一个对 isc.org 的 NS 记录的查找。应用了一个全局请求选项 +qr ，这样 dig 显示它所进行的每个查找的初始请求。最终的请求有一个局部请求选项 +noqr ，表示 dig 不会打印它在查找 isc.org 的 NS 记录时的初始请求。

11.28.7 IDN 支持

如果编译 dig 时带有 IDN（internationalized domain name，国际化域名）支持，它可以接受和显示非 ASCII 域名。dig 会在发送一个请求到 DNS 服务器或显示一个来自服务器的回复之前正确地转换域名的字符编码。如果由于某种原因你想关闭 IDN 支持，使用参数 +noidnin 和 +noidnout ，或者定义 IDN_DISABLE 环境变量。

11.28.8 文件

/etc/resolv.conf

${HOME}/.digrc

11.28.9 参见

delv(1), host(1), named(8), dnssec-keygen(8), RFC 1035.

11.28.10 缺陷

具有可能是太多的请求选项。

第 11.29 节 nslookup - 交互式请求互联网名字服务器

11.29.1 概要

nslookup [-option] [name | -] [server]

11.29.2 描述

nslookup 是一个请求互联网域名服务器的程序。nslookup 有两种模式：交互式和非交互式。交互式模式允许用户请求名字服务器以获取关于不同主机和域的信息，或者输出一个域中的一个主机列表。非交互模式只是用于输出一个主机或域名的名字和所请求的信息。

11.29.3 参数

在下列情况将会进入交互模式：

a. 当没有给出参数时（将使用缺省名字服务器）

b. 当第一个参数是一个连字符（-）并且第二个参数是主机名或者一个名字服务器的互联网地址时。

当被查找主机的名字或者互联网地址作为第一个参数给出时，就使用非交互模式。可选的第二个参数指定一个名字服务器的主机名或者地址。

选项也可以在命令行中指定，如果它们在参数之前，并且以一个连字符做前缀。例如，要将缺省的请求类型改为主机信息，以及要将初始的超时设置为 10 秒，敲入：

```
nslookup -query=hinfo  -timeout=10
```

-version 选项使得 nslookup 输出版本号并立即退出。

11.29.4 交互命令

host [server] 使用当前缺省的服务器或者，如果设置了，指定的服务器查找主机的信息。如果主机是一个互联网地址并且请求类型是 A 或者 PTR，就返回主机的名字。如果主机是一个名字，并且没有结尾的点，就使用搜索列表来使名字合格（译注：使用搜索列表中的名字作为后缀，将要查找的名字补充为一个完整域名）。

要查找非当前域的一台主机，在名字之后添加一个点。

server domain | lserver domain 将缺省服务器设置为 domain；lserver 使用最初的服务器来查找关于 domain 的信息，而 server 则使用当前缺省的服务器。如果不能发现一个权威的答复，则返回可能具有答复的服务器的名字。

root 未实现

finger 未实现

ls 未实现

view 未实现

help 未实现

? 未实现

exit 退出程序。

set keyword[=value] 这个命令用于改变影响查找的状态信息。有效关键字是：

all 输出 set 频繁使用的选项的当前值。关于当前缺省服务器和主机的信息也会被输出。

class=value 改变请求类为下列之一：

IN Internet 类

CH Chaos 类

HS Hesiod 类

ANY 通配符

类指定信息的协议组。

(缺省 = IN; 缩写 = cl)

nodebug 打开或者关闭在搜索时对完整响应包和任何中间响应包的显示。

(缺省 = nodebug; 缩写 = [no]deb)

nod2 打开或者关闭调试模式。这显示关于 nslookup 正在做什么的更多信息。

(缺省 = nod2)

domain=name 为 name 设置搜索名单。

nosearch 如果查询请求包含至少一个点但是不以点结尾，就在请求的尾部添加域名搜索列表中的域名，直到收到一个回答。

(缺省 = search)

port=value 改变缺省的 TCP/UDP 名字服务器端口为 value。

(缺省 = 53; 缩写 = po)

querytype=value | type=value 改变信息请求的类型。

(缺省 = A，然后是 AAAA; 缩写 = q, ty)

注意：只可能指定一个请求类型，只有在未指定两者之一 时，缺省行为才是查找两种。

norecurse 告诉名字服务器请求其它服务器，如果它没有信息。

(缺省 = recurse; 缩写 = [no]rec)

ndots=number 设置在一个被禁止搜索的域名中的点（标记分隔符）的数量。绝对名字总是停止搜索。

retry=number 设置重试次数为指定的数值。

timeout=number 改变为等待一个回复的初始的超时间隔为所指定的秒数。

novc 在发送请求给服务器时总是使用一个虚电路（译注：virtual circuit，指 TCP）。

(缺省 = novc)

nofail 如果一个服务器的响应为 SERVFAIL，或者是一个指引（nofail），或者是这样一个响应上的中止请求（fail），则试探一个名字服务器。

(缺省 = nofail)

11.29.5 返回值

如果任何请求失败，nslookup 使用退出码 1 返回，否则返回 0。

11.29.6 IDN 支持

如果 nslookup 在编译时带有 IDN（国际化域名，internationalized domain name）支持，它可以接受并显示非 ASCII 域名。nslookup 在发送一个请求到 DNS 服务器之前或者在显示一个来自服务器的响应时会适当的转换域名的字符编码。如果由于某种原因，你希望关闭 IDN 支持，定义 IDN_DISABLE 环境变量即可。当 nslookup 运行时，这个变量被定义，或者当标准输出不是一个终端时，IDN 支持将被关闭。

11.29.7 文件

/etc/resolv.conf

11.29.8 参见

dig(1), host(1), named(8).

第 11.30 节 named - 互联网域名服务器

11.30.1 概要

named [[-4] | [-6]] [-c config-file] [-d debug-level] [-D string] [-E engine-name] [-f] [-g] [-L logfile] [-M option] [-m flag] [-n #cpus] [-p port] [-s] [-S #max-socks] [-t directory] [-U #listeners] [-u user] [-v] [-V] [-X lock-file] [-x cache-file]

11.30.2 描述

named 是一个域名系统（DNS）服务器，是由 ISC 发布的 BIND 9 的一部份。关于更多 DNS 的信息，参考 RFC 1033 ， RFC 1034 和 RFC 1035 。

在没有参数时调用，named 将读缺省的配置文件 /etc/named.conf ，从其中读入所有初始数据，并监听端口以待请求。

11.30.3 选项

-4 即使主机支持 IPv6，也只使用 IPv4。-4 和 -6 是互斥的。

-6 即使主机支持 IPv4，也只使用 IPv6。-4 和 -6 是互斥的。

-c config-file 使用 config-file 作为配置文件，以取代缺省的 /etc/named.conf 。由于配置文件中可能的 directory 选项，服务器改变了其工作目录。要保证重新装入配置文件之后能够继续工作，config-file 应该是一个绝对路径。

-d debug-level 设置服务器守护进程的调试级别为 debug-level。随着调试级别的增加，来自 named 的调试跟踪信息就会更冗长。

-D string 指定一个用于在一个进程列表中标识一个 named 实例的 string。string 的内容是未检查过的。

-E engine-name 如果适用，指定要使用的加密硬件，例如用于签名的安全密钥存储库。

当 BIND 使用带 OpenSSL PKCS#11 支持构建时，这个缺省值是字符串"pkcs11"，它标识一个可以驱动一个加密加速器或硬件服务模块的 OpenSSL 引擎，当 BIND 使用带原生 PKCS#11 加密（–enable-native-pkcs11）构建时，它缺省是由"–with-pkcs11" 指定的 PKCS#11 提供者库的路径。

-f 在前台运行服务器（即，不做守护进程化）。

-g 在前台运行服务器并强制将所有日志写到 stderr 。

-L logfile 写日志到文件 logfile ，替代缺省的系统日志。

-M option 设置缺省的内存上下文选项。如果设置为 external，这就绕过了内部内存管理器，以支持系统所提供的内存申请函数。如果设置为 fill，在分配或释放时，内存块将被填充为标记值，以辅助调试内存问题。（nofill 关闭这个特性，这是缺省值，除非 named 编译时带有开发者选项。）

-m flag 打开内存使用的调试标志。可能的标志是 usage，trace，record，size 和 mctx。这些与 ISC_MEM_DEBUGXXXX 相关的标志在 <isc/mem.h> 中描述。

-n #cpus 创建 #cpus 个工作线程来利用多个 CPU。如果未指定，named 会试图决定 CPU 的个数并为每个 CPU 创建一个线程。如果它不能决定 CPU 的数量，就只创建一个工作线程。

-p port 在端口 port 监听请求。如果未指定，缺省是 53 端口。

-s 在退出时将内存使用统计写到 stdout 。

注解：这个选项主要是对 BIND 9 的开发者有趣，在未来的版本中可能被去掉或改变。

-S #max-socks 允许 named 使用最大数量直到 #max-socks 的套接字。在使用缺省配置选项构建的系统上缺省值为 21000，在使用 "configure –with-tuning=small" 构建的系统上缺省值

为 4096。

警告：这个选项对大量的多数用户而言是不需要的。使用这个选项甚至是有害的，因为所指定的值可能超过下层系统 API 的限制。因此仅仅在缺省配置会耗尽文件描述符并且确认运行环境支持所指定数目的套接字时才设置它。还要注意的是实际的最大数目通常比所指定的值小一点点，因为 named 保留一些文件描述符供其内部使用。

-t directory 在处理命令行参数之后而在读配置文件之前，将根改变为 directory。

警告：这个选项应该与 -u 选项结合使用，因为改变一个以 root 用户运行的进程的根目录在大多数系统上并不增强安全性；定义 chroot(2) 的方式允许一个具有 root 特权的进程逃出一个改变根限制。

-U #listeners 在每个地址上使用 #listeners 个工作线程来监听 UDP 请求。如果未指定，named 将基于检测到的 CPU 个数计算一个缺省值：1 个 CPU 为 1，对超过 1 个 CPU 的机器为检测到的 CPU 个数减一。不能增加到比 CPU 个数更大的值。如果将 -n 设置为比检测到的 CPU 数目更大的值，-U 将会增加到同样的值，但不会超过它。在 Windows 上，UDP 监听器的数目被硬编码为 1，这个选项没有任何效果。

-u user 在完成特权操作后，设置用户 ID(setuid) 为 user，例如创建套接字，使其监听在特权端口上。

注解：在 Linux 上，named 使用内核提供的机制来放弃所有的 root 特权，除 bind(2) 到一个特权端口和设置进程资源限制的能力之外。很遗憾，这意谓着当 named 运行在 2.2.18 之后或 2.3.99-pre3 之后的内核上时，-u 选项才能工作，因为之前的内核不允许 setuid(2) 之后保留特权。

-v 报告版本号并退出。

-V 报告版本号和编译选项，然后退出。

-X lock-file 在运行时获取指定文件的锁；这帮助阻止同时运行重复的 named 实例。使用这个选项覆盖 named.conf 中的 lock-file 选项。如果设置为 none ，就关闭对锁文件的检查。

-x cache-file 从 cache-file 装入数据到缺省视图的缓存中。

警告：这个选项必须不能使用。仅仅是 BIND 9 的开发者对其有兴趣，在未来的版本中可能被去掉或改变。

11.30.4 信号

在常规操作中，信号不应该用于控制名字服务器；应该使用 rndc 来代替。

SIGHUP 强制服务器重新装载。

SIGINT, SIGTERM 关闭服务器。

发送任何其它信号到服务器的结果都未定义。

11.30.5 配置

named 的配置文件太复杂而无法在这里详细描述。完整的描述在 BIND 9 管理员参考手册中提供。

named 从父进程继承 umask（文件创建模式掩码）。如果文件由 named 创建，如日志文件，就需要具有定制的权限，就应当在用于启动 named 进程的脚本中显式地设置 umask 。

11.30.6 文件

/etc/named.conf 缺省配置文件。

/var/run/named/named.pid 缺省进程 ID 文件。

11.30.7 参见

RFC 1033, RFC 1034, RFC 1035, named-checkconf(8), named-checkzone(8), rndc(8), :manpage:`named.conf(5), BIND 9 管理员参考手册。

第 11.31 节 pkcs11-keygen - 在一个 PKCS#11 设备上生成密钥

11.31.1 概要

pkcs11-keygen [-a algorithm] [-b keysize] [-e] [-i id] [-m module] [-P] [-p PIN] [-q] [-S] [-s slot] label

11.31.2 描述

pkcs11-keygen 使 PKCS#11 设备用给定的 label （必须唯一）和 keysize 位素数生成一个新的密钥对。

11.31.3 参数

-a algorithm 指定密钥的算法类：支持的类是 RSA，DSA，DH，ECC 和 ECX。除了这些字符串，algorithm 也可被指定为一个 DNSSEC 签名算法；例如，NSEC3RSASHA1 映射到 RSA，ECDSAP256SHA256 到 ECC，ED25519 到 ECX。缺省类是 “RSA”。

-b keysize 使用 keysize 位的素数建立密钥对。对 ECC 密钥，仅有的有效值是 256 和 384，缺省是 256。对 ECX 密钥，仅有的有效值是 256 和 456，缺省是 256。

-e 仅对 RSA 密钥，使用一个大的指数。

-i id 使用 id 创建密钥对象。id 是一个无符号 2 字节短整数或一个无符号 4 字节长整数之一。

-m module 指定 PKCS#11 提供者模块。这必须是为设备实现 PKCS#11 API 的一个共享库对象的完整路径。

-P 设置新的私钥为不敏感和可提取的。这允许私钥数据可以被 PKCS#11 设备读取。缺省对私钥是敏感和不可提取的。

-p PIN 为设备指定 PIN。如果在命令行没有提供 PIN，pkcs11-keygen 将会提示输入。

-q 安静模式：阻止不必要的输出。

-S 仅对 Diffie-Hellman(DH) 密钥，使用一个特定的 768 位，1024 位或者 1536 位素数和基数（也称为生成器）2。如果未指定，缺省大小为 1024 位。

-s slot 使用给定的 PKCS#11 槽打开会话。缺省是槽 0。

11.31.4 参见

pkcs11-destroy(8), pkcs11-list(8), pkcs11-tokens(8), dnssec-keyfromlabel(8)

第 11.32 节　pkcs11-tokens - 列出 PKCS#11 的可用符号

11.32.1　概要

pkcs11-tokens [-m module] [-v]

11.32.2　描述

pkcs11-tokens 列出 PKCS#11 可用符号，缺省来自应用初始化时所执行的槽/符号扫描。

11.32.3　参数

-m module　指定 PKCS#11 提供者模块。这必须是为设备实现 PKCS#11 API 的一个共享库对象的完整路径。

-v　使 PKCS#11 libisc 初始化可见。

11.32.4　参见

pkcs11-destroy(8), pkcs11-keygen(8), pkcs11-list(8)

第 11.33 节　pkcs11-list - 列出 PKCS#11 对象

pkcs11-list [-P] [-m module] [-s slot] [-i ID] [-l label] [-p PIN]

11.33.1　描述

pkcs11-list 列出带有 ID 或 label 的 PKCS#11 对象，或者缺省的所有对象。对所有密钥，显示对象类，标签和 ID。对私钥，也显示可提取性属性，即 true 、false 或 never 。

11.33.2　参数

-P　仅列出公共对象。（注意在某些 PKCS#11 设备上，所有的对象都是私有的。）

-m module　指定 PKCS#11 提供者模块。这必须是为设备实现 PKCS#11 API 的一个共享库对象的完整路径。

-s slot 使用给定的 PKCS#11 槽打开会话。缺省是槽 0。

-i ID 仅列出带有给定对象 ID 的密钥对象。

-l label 仅列出带有给定标签的密钥对象。

-p PIN 为设备指定 PIN。如果在命令行没有提供 PIN，pkcs11-list 将会提示输入。

11.33.3 参见

pkcs11-destroy(8), pkcs11-keygen(8), pkcs11-tokens(8) pkcs11-destroy - 销毁 PKCS#11 对象

11.33.4 概要

pkcs11-destroy [-m module] [-s slot] [-i ID] [-l label] [-p PIN] [-w seconds]

11.33.5 描述

pkcs11-destroy 销毁存储于 PKCS#11 设备中的密钥，密钥由它们的 ID 或 label 标识。

在销毁之前显示匹配的密钥。缺省时，有五秒的延迟以允许用户在销毁发生之前中断这个过程。

11.33.6 参数

-m module 指定 PKCS#11 提供者模块。这必须是为设备实现 PKCS#11 API 的一个共享库对象的完整路径。

-s slot 使用给定的 PKCS#11 槽打开会话。缺省是槽 0。

-i ID 销毁所给对象 ID 的密钥。

-l label 销毁所给标签的密钥。

-p PIN 为设备指定 PIN。如果在命令行没有提供 PIN，pkcs11-destroy 将会提示输入。

-w seconds 指定在执行密钥销毁之前暂停的时间，缺省是五秒。如果设置为 0 ，销毁将会立即执行。

11.33.7 参见

pkcs11-keygen(8), pkcs11-list(8), pkcs11-tokens(8)

第 11.34 节　named-checkconf - named 配置文件语法检查工具

11.34.1　概要

named-checkconf [-chjlvz] [-p [-x]] [-t directory] {filename}

11.34.2　描述

named-checkconf 检查一个 named 配置文件的语法，但是不检查语义。将会分析配置文件及其所有包含的文件并检查语法错误，缺省是读 /etc/named.conf 。

注意：named 在分离的分析器上下文中所读的文件，如 rndc.key 和 bind.keys ，是不会自动被 named-checkconf 读取的。即使 named-checkconf 成功，这些文件中的错误也可能导致 named 启动失败。然而，named-checkconf 可以显式地检查这些文件。

11.34.3　选项

-h 打印用法摘要并退出。

-j 在装载一个区文件时，如果存在日志文件，就读入。

-l 列出所有配置的区。每行输出包含区名，类（例如，IN），视图，和类型（例如，master 或者 slave）。

-c 只检查“核心”配置。这个禁止装载插件模块，并导致所有针对 plugin 语句的参数被忽略。

-i 忽略在已废弃选项上的警告。

-p 如果没有检测到错误，以正规形式打印 named.conf 和被包含文件。参见 -x 选项。

-t directory 改变根到 directory ，这样在配置文件中包含的指令就象运行在类似的被改变了根的 named 中一样被处理。

-v 打印 named-checkconf 程序的版本并退出。

-x 在以规范形式打印配置文件时，通过替代为问号（‘?’）串的方式隐藏共享密钥。这允许 named.conf 的内容和相关的文件被共享——例如，当提交错误报告时——而不损失私密数据。这个选项在不用 -p 时不能使用。

-z 执行 named.conf 中所有主区的测试装载。

filename 所要检查的配置文件的名字。如果未指定，缺省为 /etc/named.conf 。

11.34.4 返回值

named-checkconf 返回一个退出状态，如果检测到错误为 1，否则为 0。

11.34.5 参见

named(8), named-checkzone(8), BIND 9 管理员参考手册。

第 11.35 节 named-checkzone, named-compilezone - 区文件正确性检查和转换工具

11.35.1 概要

named-checkzone [-d] [-h] [-j] [-q] [-v] [-c class] [-f format] [-F format] [-J filename] [-i mode] [-k mode] [-m mode] [-M mode] [-n mode] [-l ttl] [-L serial] [-o filename] [-r mode] [-s style] [-S mode] [-t directory] [-T mode] [-w directory] [-D] [-W mode] {zonename} {filename}

named-compilezone [-d] [-j] [-q] [-v] [-c class] [-C mode] [-f format] [-F format] [-J filename] [-i mode] [-k mode] [-m mode] [-n mode] [-l ttl] [-L serial] [-r mode] [-s style] [-t directory] [-T mode] [-w directory] [-D] [-W mode] {-o filename} {zonename} {filename}

11.35.2 描述

named-checkzone 检查一个区文件的语法和完整性。它与 named 在装载一个区时执行同样的检查。这使 named-checkzone 能在将区文件配置到一个名字服务器之前对其进行有用的检查。

named-compilezone 与 named-checkzone 相似，但是它总是以一个特殊的格式将区的内容转储到一个特定的文件中。另外，它缺省使用更加严格的检查级别，因为转储的输出将用作一个实际的区文件并由 named 所装载。否则，在手工指定时，必须至少达到 named 配置文件所指定的检查级别。

11.35.3 选项

-d 打开调试。

-h 打印用法摘要并退出。

-q 安静模式 - 仅有退出码。

-v 打印 named-checkzone 程序的版本并退出。

-j 在装载区文件时读日志，如果后者存在。日志文件名是由区文件名后加上字符串 .jnl 。

-J filename 在装载区文件时，从给定的文件读日志，如果后者存在。（隐含-j）。

-c class 指定区的类。如果未指定，就假设为 “IN”。

-i mode 对已装载区执行完整性检查。可能的模式为 "full" （缺省），"full-sibling" ，"local" ，"local-sibling" 和 "none" 。

模式 "full" 检查指向 A 或 AAAA 记录的 MX 记录（包括区内和区外主机名）。模式 "local" 仅仅检查指向区内主机名的 MX 记录。

模式 "full" 检查指向 A 或 AAAA 记录的 SRV 记录（包括区内和区外主机名）。模式 "local" 仅仅检查指向区内主机名的 SRV 记录。

模式 "full" 检查指向 A 或 AAAA 记录的授权 NS 记录（包括区内和区外主机名）。它也检查在区内与子域所广播记录匹配的粘着地址记录。模式 "local" 仅仅检查指向区内主机名的 NS 记录，或者指向要求粘着记录存在，即名字服务器在一个子区中，的 NS 记录。

模式 "full-sibling" 和 "local-sibling" 关闭兄弟粘着记录检查，但是其它方面分别与 "full" 和 "local" 相同。

模式 "none" 关闭所有检查。

-f format 指定区文件格式。可能的格式为 "text" （缺省），"raw" 和 "map" 。

-F format 指定输出文件的格式。对 named-checkzone ，这个不会有任何效果，除非它转储区的内容。

可能的格式为 "text" （缺省），这是区的标准文本表示形式，和 "map" ，"raw" 及 "raw=N" ，将会以二进制格式存放区以使 named 快速装载它。"raw=N" 指定 raw 区文件的格式版本：如果 N 是 0，原始文件可以被任何版本的 named 读取；如果 N 是 1，则文件只能被 9.9.0 或更高版本读取。缺省为 1。

-k mode 使用指定的失败模式执行 "check-names" 检查。可能的模式为 "fail" （named-compilezone 的缺省模式） ，"warn" （named-checkzone 的缺省模式）和 "ignore" 。

-l ttl 为输入文件设定一个允许的最大 TTL。任何一个 TTL 大于这个值的记录都会导致区被拒绝。这类似于在 named.conf 中使用 max-zone-ttl 选项。

-L serial 当将一个区编译成” raw” 或” map” 格式时，将头部中的” source serial” 值设置为指定的序列号。（预料这个功能主要被用于测试目的。）

-m mode 指定是否检查 MX 记录以查看其是否为地址。可能的模式为 "fail"，"warn"（缺省）和 "ignore"。

-M mode 检查一个 MX 记录是否指向一个 CNAME 记录。可能的模式为 "fail"，"warn"（缺省）和 "ignore"。

-n mode 指定是否检查 NS 记录以查看其是否为地址。可能的模式为 "fail"（named-compilezone 的缺省模式），"warn"（named-checkzone 的缺省模式）和 "ignore"。

-o filename 写区的输出到 filename。如果 filename 是 -，就写到标准输出。这个对 named-compilezone 是必须的。

-r mode 检查在 DNSSEC 中被当作不同的，但是在普通 DNS 语义上却是相等的记录。可能的模式为 "fail"，"warn"（缺省）和 "ignore"。

-s style 指定导出的区文件的风格。可能的模式为 "full"（缺省）和 "relative"。full 格式最适合用一个单独的脚本自动进行处理。在另一方面，relative 格式对人来说更易读，因而适合手工编辑。对 named-checkzone，这个不会有任何效果，除非它转储区的内容。如果输出格式不是文本，它也没有任何意义。

-S mode 检查一个 SRV 记录是否指向一个 CNAME 记录。可能的模式为 "fail"，"warn"（缺省）和 "ignore"。

-t directory 改变根到 directory，这样在配置文件中包含的指令就象运行在类似的被改变了根的 named 中一样被处理。

-T mode 检查发送方策略框架（SPF，Sender Policy Framework）记录是否存在并在不存在一个 SPF 格式的 TXT 记录时发出一个警告。可能的模式为 "warn"（缺省），"ignore"。

-w directory 改变目录为 directory，这样在主文件 $INCLUDE 指令中的相对文件名就可以工作。这与 named.conf 中的 directory 子句相似。

-D 以正式格式转储区文件。对 named-compilezone 这总是打开的。

-W mode 指定是否检查非终结通配符。非终结通配符几乎总是对通配符匹配算法（RFC 1034）理解失败的结果。可能的模式为 "warn"（缺省）和 "ignore"。

zonename 要检查的区的域名。

filename 区文件名。

11.35.4 返回值

named-checkzone 返回一个退出状态，如果检测到错误为 1，否则为 0。

11.35.5 参见

named(8), named-checkconf(8), RFC 1035, BIND 9 管理员参考手册。

索引

A

acl_name, 17
address_match_list, 17

D

dialup_option, 18
domain_name, 17
dotted_decimal, 17

F

fixedpoint, 18

I

ip4_addr, 17
ip6_addr, 17
ip_addr, 17
ip_dscp, 17
ip_port, 17
ip_prefix, 18

K

key_id, 18
key_list, 18

M

masters_list, 17

N

namelist, 17
number, 18

P

path_name, 18
port_list, 18

R

RFC
- RFC 821, 59
- RFC 882, 199
- RFC 883, 199
- RFC 920, 199
- RFC 952, 59
- RFC 974, 209
- RFC 1033, 3, 209, 295, 298
- RFC 1034, 3, 30, 31, 59, 63, 125, 126, 199, 202, 233, 280, 281, 295, 298, 305
- RFC 1035, 3, 129, 130, 199, 202, 232, 233, 292, 295, 298, 306
- RFC 1101, 209
- RFC 1123, 53, 59, 202
- RFC 1183, 207
- RFC 1464, 207
- RFC 1521, 209
- RFC 1535, 206
- RFC 1536, 206
- RFC 1537, 209
- RFC 1591, 206
- RFC 1706, 206
- RFC 1712, 207
- RFC 1713, 206
- RFC 1750, 209
- RFC 1794, 206
- RFC 1876, 207
- RFC 1912, 206

RFC 1918, 81, 90, 114, 141
RFC 1982, 202, 255
RFC 1995, 140, 202
RFC 1996, 139, 202
RFC 2010, 209
RFC 2052, 209
RFC 2065, 209
RFC 2104, 281
RFC 2136, 139, 202, 276, 281
RFC 2137, 209
RFC 2163, 202
RFC 2168, 209
RFC 2181, 202
RFC 2219, 208
RFC 2230, 206
RFC 2240, 209
RFC 2308, 130, 202
RFC 2317, 130, 208
RFC 2345, 208
RFC 2352, 206
RFC 2535, 148, 210, 238, 240, 244, 276, 280, 281
RFC 2536, 211
RFC 2537, 210
RFC 2538, 210
RFC 2539, 202, 239, 242
RFC 2540, 208
RFC 2606, 208
RFC 2671, 210
RFC 2672, 210
RFC 2673, 210
RFC 2782, 203
RFC 2825, 206
RFC 2826, 206
RFC 2845, 145, 203, 238, 242, 276, 281
RFC 2874, 209
RFC 2915, 210
RFC 2929, 210
RFC 2930, 148, 203, 238
RFC 2931, 148, 203, 276, 281
RFC 3007, 139, 203, 281
RFC 3008, 210
RFC 3056, 123
RFC 3071, 206
RFC 3090, 210
RFC 3110, 203
RFC 3123, 174, 208
RFC 3152, 210
RFC 3225, 203
RFC 3226, 203
RFC 3258, 206
RFC 3363, 174, 175, 206, 211
RFC 3445, 210
RFC 3490, 210
RFC 3491, 210
RFC 3492, 203
RFC 3493, 191, 206
RFC 3496, 207
RFC 3542, 185, 191
RFC 3548, 13
RFC 3587, 201
RFC 3596, 202
RFC 3597, 203, 232, 275, 285, 290
RFC 3645, 203, 276
RFC 3655, 210
RFC 3658, 210, 260
RFC 3755, 210
RFC 3757, 210
RFC 3833, 207
RFC 3845, 211
RFC 3901, 208
RFC 4025, 203
RFC 4033, 149, 203, 247, 258
RFC 4034, 149, 203, 238, 242, 246, 276

RFC 4035, 149, 203, 276
RFC 4074, 207
RFC 4193, 81
RFC 4255, 203
RFC 4294, 211
RFC 4343, 204
RFC 4398, 204
RFC 4408, 211
RFC 4431, 209, 276
RFC 4470, 204
RFC 4509, 204, 260
RFC 4592, 204
RFC 4635, 204
RFC 4641, 258
RFC 4641#4.2.1.1, 256
RFC 4641#4.2.1.2, 256
RFC 4701, 204
RFC 4892, 207
RFC 4955, 204
RFC 5001, 204
RFC 5011, 22, 100, 101, 150, 151, 157, 158, 202, 220, 234, 245, 251, 252, 273
RFC 5074, 276
RFC 5155, 204, 226, 276
RFC 5452, 204
RFC 5625, 208
RFC 5702, 204
RFC 5737, 81
RFC 5936, 204
RFC 5952, 204
RFC 5966, 211
RFC 6052, 48, 204
RFC 6147, 204
RFC 6303, 208
RFC 6594, 205
RFC 6598, 81
RFC 6604, 205
RFC 6605, 205, 260
RFC 6672, 205
RFC 6698, 205
RFC 6725, 205
RFC 6742, 208
RFC 6781, 207
RFC 6840, 205
RFC 6844, 211
RFC 6891, 97, 202
RFC 6944, 211, 212
RFC 7043, 207
RFC 7129, 207
RFC 7216, 205
RFC 7314, 208
RFC 7344, 205, 235, 238, 260
RFC 7477, 205
RFC 7553, 207
RFC 7583, 207
RFC 7706, 114
RFC 7766, 205
RFC 7793, 208
RFC 7816, 45
RFC 7828, 205
RFC 7830, 205
RFC 7929, 208
RFC 8080, 205
RFC 8482, 205
RFC 8490, 205
RFC 8624, 205
RFC 8749, 206
RFC 8906, 208

S

size_or_percent, 18
size_spec, 18

Y

yes_or_no, 18