

BIND 9管理员参考手册

BIND版本9.10.2



Copyright © 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015 Internet Systems Consortium, Inc. ("ISC")

Copyright © 2000, 2001, 2002, 2003 Internet Software Consortium.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND ISC DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL ISC BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Internet System Consortium
950 Charter Street
Redwood City, California
USA
<http://www.isc.org/>

Copyright © 2008-2015 本文档中文版权归sunguonian@gmail.com所有

允许任何个人、任何组织以任何目的、任何形式付费或免费使用，拷贝，修改和分发本文档，前提是需要
在所有拷贝中出现上述的版权声明和本许可声明。

Contents

1	介绍	1
1.1	文档范围	1
1.2	本文档的组织	1
1.3	本文档中的惯例	1
1.4	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	2
1.4.4	权威名字服务器	3
1.4.4.1	主服务器	3
1.4.4.2	辅服务器	3
1.4.4.3	隐藏服务器	3
1.4.5	缓存名字服务器	3
1.4.5.1	转发	3
1.4.6	名字服务器的多个角色	4
2	BIND资源要求	5
2.1	硬件要求	5
2.2	CPU要求	5
2.3	内存要求	5
2.4	名字服务器增强环境问题	5
2.5	所支持的操作系统	5
3	名字服务器配置	7
3.1	样例配置	7
3.1.1	一个只缓存名字服务器	7
3.1.2	一个只权威名字服务器	7
3.2	负载均衡	8
3.3	名字服务器操作	9
3.3.1	用于名字服务器后台进程的工具	9
3.3.1.1	诊断工具	9
3.3.1.2	管理工具	9
3.3.2	信号	11
4	高级DNS特征	13
4.1	通知 (Notify)	13
4.2	动态更新 (Dynamic Update)	13
4.2.1	日志文件	13
4.3	增量区传送 (IXFR)	14
4.4	分割DNS	14
4.4.1	分割DNS设置的例子	14
4.5	TSIG	17
4.5.1	对每对主机生成共享密钥	17
4.5.1.1	自动生成	17
4.5.1.2	手工生成	18
4.5.2	拷贝共享密钥到两台主机	18
4.5.3	告知服务器密钥的存在	18
4.5.4	指示服务器使用密钥	18
4.5.5	基于TSIG密钥的访问控制	18
4.5.6	错误	19
4.6	TKEY	19
4.7	SIG(0)	19
4.8	DNSSEC	19

4.8.1	生成密钥	20
4.8.2	对区签名	20
4.8.3	配置服务器	20
4.9	DNSSEC, 动态区, 和自动化签名	22
4.9.1	从不安全转换到安全	22
4.9.2	动态DNS更新方法	23
4.9.3	完全自动化区签名	23
4.9.4	私有类型记录	24
4.9.5	DNSKEY轮转	24
4.9.6	动态DNS更新方法	24
4.9.7	自动密钥轮转	24
4.9.8	通过UPDATE轮转NSEC3PARAM	25
4.9.9	从NSEC转换到NSEC3	25
4.9.10	从NSEC3转换到NSEC	25
4.9.11	从安全转换为不安全	25
4.9.12	周期性的重签名	25
4.9.13	NSEC3 and OPTOUT	25
4.10	动态的信任锚点管理	25
4.10.1	验证解析器	25
4.10.2	权威服务器	26
4.11	PKCS#11 (Cryptoki)支持	26
4.11.1	先决条件	27
4.11.2	原生PKCS#11	27
4.11.2.1	构建SoftHSMv2	27
4.11.3	基于OpenSSL的PKCS#11	27
4.11.3.1	给OpenSSL打补丁	28
4.11.3.2	在Linux上为AEP Keyper构建OpenSSL	28
4.11.3.3	为Solaris上的SCA 6000构建OpenSSL	29
4.11.3.4	为SoftHSM构建OpenSSL	29
4.11.3.5	为Linux配置带AEP Keyper的BIND 9	30
4.11.3.6	为Solaris配置带SCA 6000的BIND 9	30
4.11.3.7	为SoftHSM配置BIND 9	31
4.11.4	PKCS#11工具	31
4.11.5	使用HSM	31
4.11.6	在命令行指定引擎	32
4.11.7	以自动区重签的方式运行named	33
4.12	DLZ (Dynamically Loadable Zones, 动态加载区)	33
4.12.1	配置DLZ	33
4.12.2	样板DLZ驱动	34
4.13	BIND 9对IPv6的支持	35
4.13.1	使用AAAA记录查找地址	35
4.13.2	使用半字节格式从地址查名字	35
5	BIND 9的轻量级解析器	37
5.1	轻量级解析器库	37
5.2	运行一个解析器看守进程	37
6	BIND 9配置参考	39
6.1	配置文件元素	39
6.1.1	地址匹配表	40
6.1.1.1	语法	40
6.1.1.2	定义和用法	40
6.1.2	注释语法	41
6.1.2.1	语法	41
6.1.2.2	定义和用法	41
6.2	配置文件语法	42
6.2.1	acl语句语法	42
6.2.2	acl语句定义和用法	43
6.2.3	controls语句语法	43

6.2.4	controls 语句定义和用法	44
6.2.5	include 语句语法	44
6.2.6	include 语句定义和用法	44
6.2.7	key 语句语法	44
6.2.8	key 语句定义和用法	45
6.2.9	logging 语句语法	45
6.2.10	logging 语句定义和用法	45
6.2.10.1	channel 短语	46
6.2.10.2	category 短语	47
6.2.10.3	query-errors 类别	49
6.2.11	lwres 语句语法	50
6.2.12	lwres 语句定义和用法	50
6.2.13	masters 语句语法	51
6.2.14	masters 语句定义和用法	51
6.2.15	options 语句语法	51
6.2.16	options 语句定义和用法	55
6.2.16.1	布尔选项	60
6.2.16.2	转发	65
6.2.16.3	双栈服务器	65
6.2.16.4	访问控制	66
6.2.16.5	接口	67
6.2.16.6	请求地址	68
6.2.16.7	区传送	69
6.2.16.8	UDP端口列表	71
6.2.16.9	操作系统资源限制	72
6.2.16.10	服务器资源限制	72
6.2.16.11	周期的内部任务	73
6.2.16.12	拓扑	73
6.2.16.13	sortlist 语句	74
6.2.16.14	资源记录集排序	75
6.2.16.15	调优	75
6.2.16.16	服务器内置信息区	78
6.2.16.17	内置空区	78
6.2.16.18	附加部份缓存	82
6.2.16.19	内容过滤	82
6.2.16.20	响应策略区 (RPZ) 重写	83
6.2.16.21	响应比率限制	86
6.2.17	server 语句语法	87
6.2.18	server 语句定义和用法	88
6.2.19	statistics-channels 语句语法	89
6.2.20	statistics-channels 语句定义和用法	89
6.2.21	trusted-keys 语句语法	90
6.2.22	trusted-keys 语句定义和用法	90
6.2.23	managed-keys Statement Grammar	90
6.2.24	managed-keys Statement Definition and Usage	91
6.2.25	view 语句语法	91
6.2.26	view 语句定义和用法	92
6.2.27	zone 语句语法	93
6.2.28	zone 语句定义和用法	96
6.2.28.1	区类型	96
6.2.28.2	类	98
6.2.28.3	区选项	98
6.2.28.4	动态更新策略	102
6.2.28.5	多视图	104
6.3	区文件	105
6.3.1	资源记录的类型及使用时机	105
6.3.1.1	资源记录	105
6.3.1.2	文本形式的资源记录表示	107
6.3.2	对MX记录的讨论	108

6.3.3	设置TTL	108
6.3.4	IPv4中的反向映射	108
6.3.5	其它区文件指令	109
6.3.5.1	The @ (at-sign)	109
6.3.5.2	\$ORIGIN指令	109
6.3.5.3	\$INCLUDE指令	109
6.3.5.4	\$TTL指令	110
6.3.6	BIND主文件扩展: \$GENERATE指令	110
6.3.7	附加文件格式	111
6.4	BIND9统计	111
6.4.0.1	统计文件	112
6.4.1	统计计数器	112
6.4.1.1	名字服务器统计计数器	112
6.4.1.2	区维护统计计数器	114
6.4.1.3	解析器统计计数器	114
6.4.1.4	套接字I/O统计计数器	115
6.4.1.5	对BIND 8计数器的兼容性	115
7	BIND 9安全考虑	117
7.1	访问控制表	117
7.2	Chroot和Setuid	118
7.2.1	chroot环境	118
7.2.2	使用setuid函数	118
7.3	动态更新的安全	118
8	排除故障	119
8.1	通常问题	119
8.1.1	它不工作; 我如何判定哪里出错了?	119
8.2	增加和修改序列号	119
8.3	从哪里获得帮助?	119
A	发行注记	121
A.1	BIND版本9.10.2的发行注记	121
A.1.1	介绍	121
A.1.2	Download	121
A.1.3	安全修补	121
A.1.4	新特性	121
A.1.5	有变化的特性	122
A.1.6	Bug Fixes	122
A.1.7	End of Life	123
A.1.8	Thank You	123
B	DNS和BIND的简要历史	125
B.1	Section	125
C	通用DNS参考信息	127
C.1	IPv6地址 (AAAA)	127
C.2	参考书目 (和建议读物)	127
C.2.1	注释请求 (RFC)	127
C.2.2	互联网草案	131
C.2.3	其它关于BIND的文档	131
D	BIND 9 DNS库支持	133
D.1	BIND 9 DNS Library Support	133
D.1.1	先决条件	133
D.1.2	编译	133
D.1.3	安装	133
D.1.4	已知的缺陷/限制	134
D.1.5	dns.conf文件	134
D.1.6	例子应用	134

D.1.6.1	sample: 一个简单的存根解析器应用	134
D.1.6.2	sample-async: 一个例子存根解析器, 异步工作	135
D.1.6.3	sample-request: 一个简单的DNS事物客户端	135
D.1.6.4	sample-gai: getaddrinfo()和getnameinfo()测试代码	136
D.1.6.5	sample-update: 一个简单的动态更新客户端程序	136
D.1.6.6	nsprobe: domain/name server checker in terms of RFC 4074	137
D.1.7	库参考	137
E	手册页	139
E.1	dig	139
E.2	host	144
E.3	delv	146
E.4	dnssec-checkds	149
E.5	dnssec-coverage	150
E.6	dnssec-dsfromkey	152
E.7	dnssec-importkey	154
E.8	dnssec-keyfromlabel	155
E.9	dnssec-keygen	158
E.10	dnssec-revoke	162
E.11	dnssec-settime	163
E.12	dnssec-signzone	165
E.13	dnssec-verify	169
E.14	named-checkconf	171
E.15	named-checkzone	172
E.16	named	174
E.17	named-journalprint	177
E.18	named-rrchecker	178
E.19	nsupdate	178
E.20	rndc	182
E.21	rndc.conf	186
E.22	rndc-confgen	188
E.23	ddns-confgen	190
E.24	arpaname	191
E.25	genrandom	192
E.26	isc-hmac-fixup	192
E.27	nsec3hash	193

Chapter 1

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。

1.1 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集团（Internet Systems Consortium，ISC）的BIND版本9软件包的基本信息。

这个版本的手册对应于BIND的9.10版本。

1.2 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BIND9软件的安装过程提供帮助。接下来的第四章也是面向任务的部份，它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。第五章描述BIND 9的轻量级解析器。第六章的内容按照参考手册组织，以帮助正在进行的软件维护。第七章指明安全考虑，第八章包含排错帮助。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。

1.3 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 固定宽度
用户输入	Fixed Width Bold 固定宽度的黑体
程序输出	Fixed Width 固定宽度

下列排版惯例用于描述BIND配置文件：

要描述的内容：	我们使用的风格：
关键字	Fixed Width 固定宽度
变量	Fixed Width 固定宽度

可选输入	[Text is enclosed in square brackets 文本在方括号之内]
------	--

1.4 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**, 和一个解析器库, **liblwres**。旧的**libbind**解析器库也可以作为一个单独的下载包从ISC得到。

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, 被成为一个域。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所属于的顶级域, example是com下的一个子域, 而ourhost是主机名。

为管理考虑, 名字空间划分为名叫区的单位, 每个区以节点开始, 向下扩展到树叶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[第6.3.1节](#)中描述。

关于DNS设计和DNS协议的更详细的信息, 请参见[第C.2.1节](#)中所列的标准文档。

1.4.3 区

为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 记录表示, 它应该与被授权区的根的NS 记录匹配。

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, 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、1034 和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。当你询问一些其它站点是否愿意充当你的域的辅服务器时, 你实际上是想要其对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。

权威服务器的应答的数据包中包括“权威回答”（AA）位。这在使用象**dig**（第3.3.1.1节）这样的工具来调试DNS配置时容易鉴别。

1.4.4.1 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。

在某些情况下，主文件是完全无法由手工编辑而成，只能是以动态更新操作的结果来代替。

1.4.4.2 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**）的复制过程从其它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，但是也可能从其它辅服务器传送。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。

1.4.4.3 隐藏服务器

通常区的所有权威服务器都在其上级区中有NS记录。这些NS记录组成了上级对这个区的授权。权威服务器也在其自己的区文件中列出，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仍然使区可用。

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能直接同外面的世界通信。

1.4.5 缓存名字服务器

由大多数操作系统所提供的解析库叫做存根解析器，意思是它们没有能够通过直接与权威服务器通信而执行完全的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。这样的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。

1.4.5.1 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。

可以有一个或多个转发服务器，它们轮流被请求，直到名单被遍历或者找到答案为止。使用转发服务器的典型情况是，当你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。一个典型的景象会涉及到一些内部DNS服务器和一个互联网防火墙。不能穿过防火墙的服务器就向能够穿过防火墙的服务器转发请求，后者代表内部的服务器向互联网上的DNS发请求。

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。

Chapter 2

BIND资源要求

2.1 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。

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。

2.2 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i486级机器变化到企业级的机器，如果你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。

2.3 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。另外，如果附加部份的缓存（[第6.2.16.18节](#)）是打开的，**max-acache-size**可以用于限制这个机制所使用的内存总量。最好的实践是保持足够的内存以装载所有的区并在内存中缓存数据—不幸地，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。

2.4 名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，不需要某台主机有象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。

2.5 所支持的操作系统

ISC BIND 9可以在大多数类UNIX操作系统和微软Windows Server 2003和2008，以及Windows XP和Vista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README文件。

Chapter 3

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。

3.1 样例配置

3.1.1 一个只缓存名字服务器

下列配置样例合适用于一个只缓存名字服务器，用于一个公司的内部客户端。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作权威服务器的，在这里服务器作“example.com”的主服务器和其子域“eng.example.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; };
};

```

3.2 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WWW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，象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](#)。

3.3 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。

3.3.1.1 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 域名信息探索者（domain information groper, **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。

用法

```
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**的帮助页。

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。

用法

```
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**的帮助页。

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。

用法

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

在没有给出参数（将会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。

3.3.1.2 管理工具

管理工具在一个服务器的管理中扮演一个主要角色。

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

用法

```
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。

用法

```
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 与**named-checkzone**相似，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。自BIND 9.2以来，**rndc**支持BIND 8的**ndc**工具中除了**ndc start**和**ndc restart**之外的所有命令，而这两个命令在**rndc**的通道模式中也是不支持的。如果你不带任何参数运行**rndc**，它将显示出以下的用法信息：

用法

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

关于可用的**rndc**命令细节，参见**rndc(8)**。

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构建时的由sysconfdir所定义的其它目录）。rndc.key文件是由**rndc-confgen -a**所生成，如第6.2.4节所描述。

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。这个密钥是一个由RFC 3548所指定的base-64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
        "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEd29tYW4K";
};
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，将允许以下命令：

\$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。

运行**rndc-confgen**程序将会方便地为你创建一个rndc.conf文件，并且会显示你所需要添加到named.conf中的相关的**controls**语句。另外一个选择是，你可以运行**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进行特定的动作，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。

Chapter 4

高级DNS特征

4.1 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的相应，辅服务器会检查其区的版本是否当前版本，如果不是，就发起区传送。

更多的关于DNS NOTIFY的信息，参见第6.2.16.1节对notify选项的描述和第6.2.16.7节对区选项also-notify的描述。NOTIFY协议由RFC 1996规定。

注意



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。

4.2 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、替换和删除其上的记录的方法。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。更多信息参见第6.2.28.4节。

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，要么是设置**tkey-gssapi-credential**和**tkey-domain**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。

4.2.1 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写到 (“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 (dump) 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 .jnk 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象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**。

如果你必须手工修改动态区，需要按照下列步骤：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” 文件中的变化更新区的主文件。编辑区文件。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，使用 **rndc sync -clean**。

4.3 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 IXFR 协议，为区提供必要的变化历史的信息。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。

4.4 分割DNS

对 DNS 空间的内部和外部解析器设置不同的视图，或可见度，通常被称为一个分割的 DNS 设置。有几个原因使某个单位想要这样设置其 DNS。

一个以此方式设置 DNS 系统的共同的原因是对 “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，一个单位可以选择使用一个分割 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。

4.4.1 分割DNS设置的例子

让我们假设一个名叫 *Example* 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 “对外” 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。为了这样的机制正常运转, 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。

为了让以上这些正常工作, 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。
- 查询互联网上的所有主机。
- 同内部网和外部网上的人员交换电子邮件。

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, 参见第3.1节。

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; };

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 master 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 slave 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" {
    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 slave zone
zone "site1.example.com" {
    type master;
    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

```

4.5 TSIG

这是一个在BIND中设置基于事务安全的事务签名 (Transaction SIGNatures, TSIG) 的简短指导。它描述了配置文件需要作的变化, 以及不同特征所要求的变化, 包括创建事务密钥的过程和在BIND 中使用事务签名。

BIND基本支持TSIG, 用于服务器到服务器的通信。包括区传送, 通知和递归查询的信息。基于BIND 8较新版本的解析器对TSIG提供有限的支持。

TSIG也可以用于动态更新。使用动态区的主服务器应该控制对动态更新服务的访问, 但是基于IP 的访问控制是不够的。由TSIG所提供的加密访问控制要好得多。**nsupdate**程序通过-k和-y命令行选项或使用内置的key语句来支持TSIG。

4.5.1 对每对主机生成共享密钥

生成一个共享密钥, 由host1和host2共享。可以任意选择一个密钥名: “host1-host2.”。两台主机上的密钥名必须一致。

4.5.1.1 自动生成

下面的命令将生成一个上面所描述的128位 (16字节) HMAC-SHA256密钥。密钥越长越好, 但是更短的密钥更易读。注意最大的密钥长度是摘要长度, 在这里是256位。长度超过512位的密钥将被MD5算法摘要成为一个128位的密钥。

```
dnssec-keygen -a hmac-sha256 -b 128 -n HOST host1-host2.
```

密钥在文件khost1-host2.+163+00000.private中。并不直接使用这个文件, 但是可以从中将“Key:”后面base-64编码的字符串抽取出来, 用作一个共享密钥。

```
Key: La/E5CjG90+os1jq0a2jdA==
```

字符串“La/E5CjG9O+os1jq0a2jdA==”可以用作共享密钥。

4.5.1.2 手工生成

共享密钥只是一个简单的以base-64编码的随机二进制位序列。大多数ASCII字符串都是有效的base-64字符串（假设其长度是4的倍数，并且只含有有效字符），所以共享密钥可以手工生成。

同样，一个字符串也可以通过使用**mmencode**或类似的程序来生成base-64编码的数据。

4.5.2 拷贝共享密钥到两台主机

这已经超出了DNS的范围。应该使用一个安全的传输机制。可以通过安全的FTP，ssh，电话等等。

4.5.3 告知服务器密钥的存在

设想host1和host2为两台服务器。将下列语句添加到各自的named.conf文件中：

```
key host1-host2. {  
    algorithm hmac-sha256;  
    secret "La/E5CjG9O+os1jq0a2jdA==";  
};
```

密钥是由以上方式所生成的。因为这是一个秘密，推荐将named.conf设置成并非所有人可读的，或者使用key指令将一个并非所有人可读的文件包含进named.conf中。

在这一点，密钥是可识别的。这意味着如果服务器收到一个由这个密钥所签名的消息，它可以验证这个签名。如果成功验证签名，响应就会使用同样的密钥签名。

4.5.4 指示服务器使用密钥

由于密钥仅仅在两台主机之间共享，服务器必须被告知去使用它。下列指令被添加到host1的named.conf文件中，假定host2的IP地址是10.1.2.3:

```
server 10.1.2.3 {  
    keys { host1-host2. ; };  
};
```

可以提供多个密钥，但只能使用第一个。这条指令不包含任何秘密，所以它可以放在一个所有人可读的文件中。

如果host1发出的消息是一个到那个地址的请求，这个消息将被指定的密钥签名。host1将会期望对这个被签名的消息的回应都被同一个密钥签名。

一个类似的语句必须出现在host2的配置文件（使用host1的IP地址）中，供host2将发向host1的消息进行签名。

4.5.5 基于TSIG密钥的访问控制

BIND允许在ACL定义和allow-{ query | transfer | update }指令中指定IP地址和网络范围。这个也被扩展到允许TSIG密钥。上述密钥被表示成key host1-host2.。

一个allow-update指令的例子是：

```
allow-update { key host1-host2. ;};
```

这仅允许被名为“**host1-host2.**”的密钥所签名的动态更新请求能够成功。

关于**update-policy**语句更灵活的讨论，参见[第6.2.28.4节](#)。

4.5.6 错误

TSIG签名消息的过程可能产生一些错误。如果一个签名消息被放到一个不知道TSIG的服务器，将返回一个FORMERR（格式错误）错误，因为服务器不能够理解这个记录。这个是错误配置的结果，因为服务器必须被显式地配置成发一个TSIG签名的消息到一个指定的服务器。

如果一个知道TSIG的服务器收到一个由其不知道的密钥所签名的消息，它将返回一个未被签名的消息，并且TSIG扩展的错误码被置为BADKEY。如果一个知道TSIG的服务器收到一个带有无效签名的消息，它将返回一个未被签名的消息，并且TSIG扩展的错误码被置为BADSIG。如果一个知道TSIG的服务器收到一个在允许的时间范围之外的消息，它将返回一个被签名的消息，并且TSIG扩展的错误码被置为BADTIME，并调整时间值，以使回应可以被成功验证。在所有这些情况，消息的rcode（响应码）都被置为NOTAUTH（未被认证）。

4.6 TKEY

TKEY是一个在两台主机之间自动生成共享密钥的机制。TKEY有几种“模式”来指定如何生成和分派密钥。BIND 9只实现了这些模式中的一种，即Diffie-Hellman 密钥交换。两台主机都需要具有Diffie-Hellman KEY记录（虽然并不要求这个记录出现在一个区中）。TKEY过程必须使用签名消息，可以使用TSIG或SIG(0)签名。TKEY的结果是一个共享秘密，可以用于在TSIG中签名消息。TKEY也可以用来删除它先前生成的共享密钥。

TKEY过程是由一个客户端或者服务器向一个知道TKEY的服务器发送一个签名的TKEY请求（包括任何合适的KEY）来发起的。服务器的响应，如果它指示是成功的，将包含一个TKEY记录和任何合适的密钥。完成这次交换后，两方的参与者都有足够的信息来决定共享密钥；精确的过程依赖于TKEY的模式。当使用Diffie-Hellman TKEY模式时，互相交换Diffie-Hellman密钥，两个参与方就得到了共享密钥。

4.7 SIG(0)

BIND 9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；服务器不会试图定位和（或）证实密钥。

多个消息的TCP流的SIG(0)签名还不支持。

跟随BIND 9发行的生成SIG(0)签名消息的唯一工具是**nsupdate**。

4.8 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工具不兼容。

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。

对其它相信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。

4.8.1 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，并且必须可以用于认证。推荐区密钥使用由IETF作为“强制实现”所指定的加密算法；当前唯一的是RSASHA1。

下列命令为child.example区生成一个768位的RSASHA1密钥：

```
dnssec-keygen -a RSASHA1 -b 768 -n ZONE child.example.
```

将产生两个输出文件：Kchild.example.+005+12345.key和Kchild.example.+005+12345.private（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（3表示DSA，1表示RSAMD5，5表示RSASH1，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。

公钥应该使用**\$INCLUDE**语句来包含.key文件的方式插入到区文件中。

4.8.2 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‘-d’选项时为子区生成的DS记录一样。如果未指定‘-d’选项，就必须为安全的子区手工添加DS资源记录集。

下列命令对区签名，假设它是一个名为zone.child.example的文件。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。

```
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，以及一个可选的dlvset文件。他们用于向上级区管理员提供DNSKEY（或者与之相关的DS记录），后者作为区的安全入口点。

4.8.3 配置服务器

为了使named正确响应知道DNSSEC的客户端所发出的DNS请求，必须将**dnssec-enable**设置为yes。（这是缺省设置。）

为了使named验证其它服务器来的响应，必须将**dnssec-enable**选项设置为yes，并且将**dnssec-validation**选项设置为yes或auto。

如果**dnssec-validation**选项被设置为auto，就会使用DNS根区的一个缺省信任锚。如果它被设置为yes，必须通过在named.conf中使用一个**trusted-keys**或**managed-keys**语句配置至少一个信任锚点，否则就不会进行DNSSEC验证。缺省设置为yes。

trusted-keys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。所有在**trusted-keys**（及相关的区）中的密钥列表都被认为是存在的，并且只有所列出的密钥可以用于验证与其来源一致的DNSKEY资源记录集。

managed-keys是通过RFC 5011信任锚点维护自动保持更新的受信任密钥。

在本文档的后面有更多关于**trusted-keys**和**managed-keys**的详细描述。

与BIND 8不同, 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。

```
managed-keys {
    /* Root Key */
    "." initial-key 257 3 3 "BNY4wrWMlnCfJ+CXd0rVXyYmobt7sEEfK3clRbGaTwS
        JxrGkxJWoZu6I7PzJu/E9gx4UC1zGAHlXKdE4zYIpRh
        aBKncvC2U9mZhkdUpd1Vso/HAdjNe8LmMlnzY3zy2Xy
        4klWOADTPzSv9eamj8V18PHGjBLaVtYvk/ln5ZApjYg
        hf+6fElrmLkdaz MQ2OCnACR8l7DF4BBa7UR/beDHyp
        5iWTXWSi6XmoJLbG9Scqc7l70KDqlvXR3M/1UUVRbke
        glIPJSidmK3ZyCllh4XSKbje/45SKucHgnwU5jefMtq
        66gKodQj+MiA2lAfUVe7u99WzTLzY3qlxDhxYQQ20FQ
        97S+LKUTpQcq27R7AT3/V5hRQxScINqwcZ4jYqZD2fQ
        dgxbcDTClU0CRBdiieyLMNzXG3";

};

trusted-keys {
    /* Key for our organization's forward zone */
    example.com. 257 3 5 "AwEAAaxPMcR2x0HbQV4WeZB6oEDX+r0QM6
        5KbhTjrWlZaARmPhEZZe3Y9ifgEuq7vZ/z
        GZUdEGNWy+JZzus0lUptwgjGwhUSl558Hb
        4JKUbbOTcM8pwXlJ0EiX3oDFVmJHO444gL
        kBOUKUf/mC7HvfwYH/Be22GnClrinKJp1O
        g4yWzO9WglMk7jbfW33gUKvirTHr25GL7S
        TQUzBb5Usxt8lgnyTUHs1t3JwCY5hKZ6Cq
        FxmAVZP20igTixin/1LcrgX/KMEGd/biuv
        F4qJCYduieHukuY3H4XMacR+xia2nIUPvm
        /oyWR8BW/hWdzOvnSCThlHf3xiYleDbt/o
        1OTQ09A0=";

    /* Key for our reverse zone. */
    2.0.192.IN-ADDRPA.NET. 257 3 5 "AQOnS4xn/IgOUpBPJ3bogzwc
        xOdNax07lL18QqZnQQQAVVr+i
        LhGTnNGp3HoWQLUIzKrJVZ3zg
        gy3WwNT6kZo6c0tszYqbtvchm
        gQC8CzKojM/Wl6i6MG/eafGU3
        siaOdS0yOI6BgPsw+YZdzlYMa
        IJGf4M4dyoKIhzdZyQ2bYQrjy
        Q4LB0lC7aOnsMyYKHHYeRvPxj
        IQXmdqgOJGq+vsevG06zW+1xg
        YJh9rCIfnm1GX/KMgxLPG2vXT
        D/RnLX+D3T3UL7HJYHJhAZD5L
        59VvjSPsZJHeDCUyWYrvPZesZ
        DIRvhDD52SKvbheeTJUmeEhkz
        ytNN2SN96QRk8j/iI8ib";

};

options {
    ...
    dnssec-enable yes;
    dnssec-validation yes;
};
```

注意

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签名的安全区的未通过验证的响应，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。

注意

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”。（在BIND 9.7之前，日志记录的错误为“not insecure”。这是指区，而不是指响应。）

4.9 DNSSEC, 动态区, 和自动化签名

作为BIND 9.7.0的一个特性，可以将一个动态区从不安全转变为签名的区并备份。一个安全的区可以使用NSEC或NSEC3链。

4.9.1 从不安全转换到安全

有两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，或者**auto-dnssec**区选项。

对这两种方法，你都需要配置**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。你可以将密钥文件放到在**named.conf**中所指名的密钥目录中：

```
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。

4.9.2 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 AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EY
> update add example.net DNSKEY 257 3 7 AwEAAAd/7odU/64o2LGsifbLtQmtO8dFDtTAZX
> send
```

虽然更新请求会几乎立即完成, 但是只有**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。

如果你希望使用NSEC3来取代NSEC作签名, 你应该添加NSEC3PARAM 记录到初始更新请求中。如果你希望NSEC3链具有OPTOUT位, 就在NSEC3PARAM记录的标志字段中设置它。

```
% nsupdate
> ttl 3600
> update add example.net DNSKEY 256 3 7 AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EY
> update add example.net DNSKEY 257 3 7 AwEAAAd/7odU/64o2LGsifbLtQmtO8dFDtTAZX
> 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**有集合建立/强调相关的链。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。

4.9.3 完全自动化区签名

要打开自动签名, 需要在**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。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(8)**和**dnssec-settime(8)**。)

named将会周期性地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。(然而, 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。

如果你希望使用NSEC3来代替NSEC给区签名, 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。如果你希望NSEC3链设置OPTOUT位, 就在NSEC3PARAM记录的标志字段中设置它。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。

4.9.4 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，这些记录将会在最后一个字节有一个非零值（这些记录都有一个初始的非零值）。

私有类型记录的格式：如果第一个字节不为0，这个记录表明区需要由与记录匹配的密钥来签名，或者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默默地忽略掉。

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
```

4.9.5 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。

4.9.6 动态DNS更新方法

为通过动态更新执行密钥轮转，你需要为新密钥添加K*文件，这样named就能够找到它们。然后你可以通过动态更新添加新的DNSKEY资源记录集。然后将引起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，你需要将新KSK通知上级域和所有的信任锚仓库。

然后，你应该在删除旧DNSKEY之前等待区中最大TTL的时间。如果更新一个KSK，你还需要等待上级区中的DS资源记录集更新和其TTL失效。这就确保当你删除旧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。

4.9.7 自动密钥轮转

当一个新密钥到达其激活日期（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，如果替代现在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。

4.9.8 通过UPDATE轮转NSEC3PARAM

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，你可以删除旧的NSEC3PARAM记录。旧的链将会在更新请求完成之后被删除。

4.9.9 从NSEC转换到NSEC3

要做这个，你只需要添加一条NSEC3PARAM记录。在转换完成后，NSEC链将被移除，NSEC3PARAM记录有一个0标志域。NSEC3链将在NSEC链被去掉之前生成。

4.9.10 从NSEC3转换到NSEC

要做这件事，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。

4.9.11 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`（或者重签名）。

4.9.12 周期性的重签名

在任何支持动态更新的安全区中，`named`会周期性地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。

4.9.13 NSEC3 and 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UPDATES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。

4.10 动态的信任锚点管理

BIND 9.7.0引入了对RFC 5011，动态信任锚点管理，的支持。使用这个特征允许`named`跟踪关键的DNSSEC密钥，而不需要操作员改变任何配置文件。

4.10.1 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`managed-keys`语句来配置信任锚点。关于这个的信息可以在第6.2.23节中找到。

4.10.2 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“活动的”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。解析器将会周期性地重新检查区，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。

在撤销一个活动的密钥之后，必须使用被撤销的KSK和新的活动的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，可以使用新的活动密钥（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，将会行使警告。

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。

4.11 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项目产生的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。

4.11.1 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。

4.11.2 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。）

要使用原生PKCS#11构建BIND，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。）

4.11.2.1 构建SoftHSMv2

SoftHSMv2，最新开发版SoftHSM，可在<https://github.com/opendnssec/SoftHSMv2> <<https://github.com/opendnssec/SoftHSMv2>>得到。它是由OpenDNSSEC项目所开发的软件库（<http://www.opendnssec.org> <<http://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 --enable-gost
$ make
$ make install
$ /opt/pkcs11/usr/bin/softhsm-util --init-token 0 --slot 0 --label softhsmv2
```

4.11.3 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“美味”，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和1.0.1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和1.0.1。

注意



这篇文档（2015年1月）的最新OpenSSL版本是0.9.8zc, 1.0.0o和1.0.1j。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。

4.11.3.1 给OpenSSL打补丁

```
$ wget http://www.openssl.org/source/openssl-0.9.8zc.tar.gz
```

解压tar包:

```
$ tar xzf 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
```

注意



注意，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，用户构建OpenSSL库的位置需要通过configure指定。

4.11.3.2 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’选项。

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
```

这个库仅仅通过32位二进制提供给Linux的。如果我们要在一台64位Linux系统上编译，需要强制进行一个32位的构建，在构建时指定-m32选项。

最后，Keyper库要求线程，所以我们必须指定-pthread。

```
$ cd openssl-0.9.8zc
$ ./Configure linux-generic32 -m32 -pthread \
    --pk11-libname=/opt/pkcs11/usr/lib/libpkcs11.so \
    --pk11-flavor=sign-only \
    --prefix=/opt/pkcs11/usr
```

在配置之后，运行“make”和“make test”。如果“make test”失败并输出“pthread_atfork() not found”，你可能忘记加上前面提到的-pthread了。

4.11.3.3 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。

4.11.3.4 为SoftHSM构建OpenSSL

SoftHSM（版本1）是一个由OpenDNSSEC项目（<http://www.opendnssec.org> <<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KCS#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 -pthread \
  --pk11-libname=/opt/pkcs11/usr/lib/libsoftsm.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。

4.11.3.5 为Linux配置带AEP Keyper的BIND 9

为了链接到PKCS#11提供者，在构建BIND 9时必须打开对线程的支持。

AEP Keyper的PKCS#11库当前只作为一个32位二进制提供。如果我们要在一个64位的主机上构建，我们必须通过在“configure”命令行的CC选项中添加“-m32”来强制进行一个32位的构建。

```
$ cd ../bind9
$ ./configure CC="gcc -m32" --enable-threads \
  --with-openssl=/opt/pkcs11/usr \
  --with-pkcs11=/opt/pkcs11/usr/lib/libpkcs11.so
```

4.11.3.6 为Solaris配置带SCA 6000的BIND 9

为链接到PKCS#11提供者，必须在BIND 9构建时打开对线程的支持。

```
$ cd ../bind9
$ ./configure CC="cc -xarch=amd64" --enable-threads \
  --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位混合搭配的体系结构。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）。

4.11.3.7 为SoftHSM配置BIND 9

```
$ cd ../bind9
$ ./configure --enable-threads \
    --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。）

4.11.4 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。

4.11.5 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。

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**。

现在我们可以再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\(8\)](#)。）

```
$ 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。如果你想，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，使用硬盘上的密钥没有速度优势，因为加密操作是由HSM完成，并不使用硬盘上的密钥。）

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
```

4.11.6 在命令行指定引擎

在使用基于OpenSSL的PKCS#11时，OpenSSL所使用的“引擎”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，“引擎”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。

4.11.7 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文件的位置可以再运行**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进入权限就能够访问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/hsm-pin**，就可以从文件**/etc/hsm-pin**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。

4.12 DLZ (Dynamically Loadable Zones, 动态加载区)

DLZ (Dynamically Loadable Zones, 动态加载区)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，仍然以同样方式链接。way.

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模块给**named**提供数据时，它使用文本格式。响应由**named**转换为DNS线上格式。这个转换没有任何内部缓存，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。）

4.12.1 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;
};

zone "." {
    type redirect;
    dlz other;
};
```

4.12.2 样板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
```

样板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。

4.13 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，参见第C.1节。

4.13.1 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。

4.13.2 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.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0  14400 IN      PTR      (
                                         host.example.com. )
```


Chapter 5

BIND 9的轻量级解析器

5.1 轻量级解析器库

传统的应用链接到一个存根解析器库，通过后者向本地缓存名字服务器发送递归的DNS请求。

IPv6给解析过程引入了新的复杂性，如：根据A6链和DNNAME记录，并且同时查找IPv4和IPv6地址。虽然大多数复杂性已经被去掉，还是很难或不可能实现一个传统的存根解析器。

因而BIND 9就通过一个运行在本地主机上的轻量级解析器库和一个解析器后台进程来为本地客户端提供解析服务。这些通信使用一个基于UDP的简单协议，“轻量级解析器协议”与完全的DNS协议区别开来，并比后者更简单。

5.2 运行一个解析器看守进程

要使用轻量级解析器接口，系统必须运行解析器后台进程**lwresd**，或者一个配置了**lwres** 语句的本地名字服务器。

缺省情况下，使用轻量级解析器库的应用向IPv4的环回地址（127.0.0.1）的921端口发送UDP请求。这个地址可以由/etc/resolv.conf中的**lwserver** 行重新指定。

当前，后台进程只在DNS中查找，但是将来它可能使用到其它资源，如/etc/hosts，NIS等等。

lwresd后台进程本质上是一个只缓存名字服务器，它使用轻量级解析器协议而不是DNS协议来响应请求。因为它需要在每台主机上运行，所有它被设计成不需要或只需要最小的配置。除非被配置成其它方式，它使用/etc/resolv.conf中的**nameserver**行中列出的名字服务器作转发者，如果没有任何指定，它也能够自动做解析。

lwresd也可以被配置成具有一个named.conf风格的配置文件，缺省为/etc/lwresd.conf。一个名字服务器也可以通过在named.conf中使用**lwres**语句而被配置成充当一个轻量级解析器后台进程。

Chapter 6

BIND 9配置参考

BIND 9的配置在很大程度上与BIND 8 相似；然而，也有一些新的配置领域，例如视图（view）。BIND 8的配置文件只需很少改动就可以工作在BIND 9中，虽然更复杂的配置应该被重新审核以检查他们是否可以用只有BIND 9才有的新特征来更有效率地实现。

BIND 4的配置文件可以使用shell脚本contrib/named-bootconf/named-bootconf.sh转换成新的格式。

6.1 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语句所定义的一个地址匹配表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，参见第6.1.1节。
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	必须用作一个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联系标识符。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，用于给去往支持差分服务代码点（DSCP）的操作系统的流量选择一个DSCP值。
ip_port	一个IP端口号。其范围为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。
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的端口号，含。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不意味着字面上的零；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。
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 ， 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。

6.1.1 地址匹配表

6.1.1.1 语法

```
address_match_list = address_match_list_element ;
[ address_match_list_element; ... ]
address_match_list_element = [ ! ] ( ip_address [/length] |
    key key_id | acl_name | { address_match_list } )
```

6.1.1.2 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**和**blackhole**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，因为算法会将所有查找1.2.3.13的匹配到元素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。

6.1.2 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。

6.1.2.1 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
```

6.1.2.2 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，只要你愿意）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



你不能象在区文件中一样，使用分号（‘;’）字符来开始一个注释。分号表示一个配置语句的结束。

6.2 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。
include	包含一个文件。
key	指定在使用TSIG时，用于的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。
lwres	配置named，使其也充当轻量级解析器看守进程（lwresd）。
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。
trusted-keys	定义信任的DNSSEC密钥。
managed-keys	列出通过RFC 5011信任锚点维护来保持更新的DNSSEC密钥。
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。

6.2.1 acl语句语法

```
acl acl-name {
    address_match_list
};
```

6.2.2 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。

当BIND 9是带GeoIP支持构建时，ACL也可用于地理访问限制。通过指定一个如下形式的ACL元素可以达成：**geoip [db database] field value**

*field*指示在一个匹配时搜索哪个字段。可用的字段为“country”，“region”，“city”，“continent”，“postal”（邮政编码），“metro”（城区代码），“area”（地区编码），“tz”（时区），“isp”，“org”，“asnum”，“domain”和“netspeed”。

*value*是要在数据库中搜索的值。一个包含空格或其它特殊字符的字符串必须使用引号。如果是一个“asnum”搜索，需要使用“ASN”开始的字符串，否则必须使用完整描述（例如，“ASN 某某公司名”）。如果是一个“country”搜索并且字符串是两个字符长度，它必须是一个标准的ISO-3166-1双字符国家代码，如果是三个字符长度，它必须是一个ISO-3166-1三字符国家代码；否则就应该是国家的全称。类似地，如果是一个“region”搜索并且字符串是两个字符长度，它必须是一个标准的双字符州或省的缩写；否则它就是州或省的全称。

*database*字段指示匹配时要搜索哪个GeoIP 数据库。大多数情况这不是必须的，因为大多数搜索字段只能在单个数据库中找到。然而，对一个国家的搜索可以由“city”，“region”或“country”数据库回答，对区域的搜索（例如，州或省）可以由“city”或“region”数据库回答。对这些搜索类型，指定一个*database*将强制请求由那个数据库而不是别的数据库回答。如果未指定*database*，这些请求将会按照顺序由“city”数据库回答，如果安装了这个数据库，或者由“region”数据库回答，如果安装了这个数据库，或者由“country”数据库回答。

一些GeoIP ACL的例子：

```
geoip country US;
geoip country JAP;
geoip db country country Canada;
geoip db region region WA;
geoip city "San Francisco";
geoip region Oklahoma;
geoip postal 95062;
geoip tz "America/Los_Angeles";
geoip org "Internet Systems Consortium";
```

6.2.3 controls语句语法

```
controls {
    [ inet ( ip_addr | * ) [ port ip_port ]
        allow { address_match_list }
        keys { key_list }; ]
    [ inet ...; ]
    [ unix path perm number owner number group number
        keys { key_list }; ]
```

```
[ unix ...; ]
};
```

6.2.4 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**。如果你只想对本机使用**rndc**，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，参见第3.3.1.2节中的[Remote Name Daemon Control application]。

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**。

建立**rndc.key**特性是为了减少从BIND 8的改变，后者在其命令通道上没有提供数字签名，因而没有**keys**子句。这使BIND 9可以不加改变地使用现成的BIND 8配置文件，并使**rndc**能够用与BIND 8中**ndc**同样的方式工作，只是在BIND 9安装完成之后简单地执行命令**rndc-confgen -a**。

由于**rndc.key**特性只是想要允许向后兼容BIND 8配置文件的用法，这个特征没有高度的配置能力。你不能很容易地改变密钥名或秘密的大小，所以，如果你希望更改这些东西，你应该用你自己的密钥建立一个**rndc.conf**。**rndc.key**文件有其自身的权限设置，以使其只能由其所有者（运行**named**的用户）才能访问。如果你期望更大的灵活性以使其它用户可以访问**rndc**命令，你需要建立一个**rndc.conf**并将其设为要访问它的用户所在的组具有组可读的权限。

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

6.2.5 include语句语法

```
include filename;
```

6.2.6 include语句定义和用法

include语句将指定的文件插入到**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。

6.2.7 key语句语法

```
key key_id {
    algorithm string;
    secret string;
```

```
};
```

6.2.8 key语句定义和用法

key语句定义了一个共享密钥，用于TSIG（参见第4.5节）或命令通道（参见第6.2.4节）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（参见第6.2.4节）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**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base-64编码的字符串。

6.2.9 logging语句语法

```
logging {
    [ channel channel_name {
        ( file path_name
          [ versions ( number | unlimited ) ]
          [ size size_spec ]
          | syslog syslog_facility
          | stderr
          | null );
        [ severity (critical | error | warning | notice |
                   info | debug [ level ] | dynamic ); ]
        [ print-category yes or no; ]
        [ print-severity yes or no; ]
        [ print-time yes or no; ]
    }; ]
    [ category category_name {
        channel_name ; [ channel_name ; ... ]
    }; ]
    ...
};
```

6.2.10 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; };
};
```

在BIND 9中，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BIND 8中，日志配置是在**logging**语句被分析时建立的。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。

6.2.10.1 channel短语

所有的日志输出都写到一个或多个**channels**；你可以按照需要建立多个通道。

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。还有些可选的选项，限定此通道所接受的消息严重级别（缺省为**info**），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目标子句将定向到一个磁盘文件。它可以包括两个指标的限制，一个是允许多大的文件，一个是在文件打开时允许多少个版本的文件存在。

如果你使用**versions**日志文件选项，**named**将通过在打开状态更改名字来保存多个备份版本。例如，如果你选择保持文件**lamers.log**的三个旧版本，那么在打开它之前，**lamers.log.1**被更名为**lamers.log.2**，**lamers.log.0**被更名为**lamers.log.1**，**lamers.log**被更名为**lamers.log.0**。你可以设置**versions unlimited**不限制版本数。如果将**size**选项附加到日志文件，只有在被打开的文件超过了所指定的大小时才会更名。缺省情况下，没有备份版本；所有现存的日志文件只是简单地追加。

size选项用于限制日志的增长，如果文件超过了这个大小，**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，文件将按照上述方式轮转，并开始一个新的。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输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。

size和**versions**选项用法例子：

```
channel an_example_channel {
    file "example.log" versions 3 size 20m;
    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。如果你的系统使用非常旧的**syslog**版本，它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（Event Viewer）。

severity子句工作起来象**syslog**的“优先级”，只有一点不同，就是如果你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。

如果你使用**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**服务器时带**-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;
};
```

将会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。

如果**print-time**被打开，日期和时间将会被记入日志。**print-time**可以用于**syslog**通道，但是通常不这样用，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
```

有四个预定义的通道供**named**缺省日志使用，具体如下：如何使用它们在[第6.2.10.2节](#)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 '-f' 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;
};

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

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。如果你需要捕捉这些输出，你必须使用“-g”选项运行服务器并将标准错误重定向到一个文件。

一旦一个通道被定义后，它不能被重定义。这样你不能直接修改内建的通道，但是你可以修改缺省的日志，即将类别指向你所定义的通道。

6.2.10.2 category短语

存在许多类别，这样你可以发送你想要看的日志，而不看你不想要的日志。如果你不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。如果你不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; };
```

作为一个例子，我们假设你想将安全事件记录到一个文件，但是你也想保持缺省的日志行为。你会如下设定：

```
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。

default	缺省类别，为没有被明确定义的配置的类别指定的日志选项。
general	捕捉所有。许多未被分类到某个类别中，都被记入此类。
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。
security	同意和拒绝请求。
config	配置文件分析及处理。
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。
xfer-in	服务器所接受的区传送。
xfer-out	服务器所发出的区传送。
notify	NOTIFY协议。
client	对客户端请求的处理。
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。
network	网络操作。
update	动态更新。
update-security	同意和拒绝更新请求。
queries	指定记录请求日志的地方。 在启动时，指定 queries 类别将启动对请求的日志，除非设定 querylog 选项。 请求日志条目报告了客户端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（E），是否使用TCP（T），DO位（DNSSEC Ok）是否被设置（D），或者CD位（Checking Disables）是否被设置（C）。在这之后，报告请求被发送到的目标地址。 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。
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。
lame-servers	跛服务器。这是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。

delegation-only	只授权。记录被强制成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。
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。
RPZ	关于响应策略区文件中错误，重写响应以及最高级 debug 中试图重写响应等的信息。
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，这种情况下以一个星号(*)开头。各种内部事件以调试级别1或更高级记录日志。
cname	对单个请求的比率限制记录在 query-errors 类别中。 记录因为一个CNAME记录而非A/AAAA记录而被跳过的名字服务器。

6.2.10.3 query-errors类别

query-errors类别是专门用于调试目的：是为了标识特定的请求为什么以及怎样导致错误响应。因此，这个类别的信息仅仅用**debug**级别记录日志。

在1或更高的调试级别时，每个返回码（rcode）为SERVFAIL的响应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2或更高的调试级别时，记录了导致SERVFAIL的递归解析的详细上下文信息。日志消息将像如下这样：

```
fetch completed at resolver.c:2970 for www.example.com/A
in 30.000183: timed out/success [domain:example.com,
referral:2,restart:7,qrysent:8,timeout:5,lame: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30.000183内完成并最终结果导致一个SERVFAIL，它是在源文件resolver.c的第2970行找到的。

接下来的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，这个请求导致SERVFAIL，可能是因为所有的名字服务器都宕机或不可达，并产生了一个30秒的超时。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信息。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。在上面的例子中是2，很可能是com和example.com。
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BIND9的地址数据库（ADB）中的查找结果，前者缓存了跛服务器。
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3或更高的调试级别时，记录那些与在调试级别1相同的消息，但只对其它错误，而不记录SERVFAIL。注意，在这里，象NXDOMAIN这样的否定响应并不被当作错误。

在4或更高的调试级别时，记录那些与在调试级别2相同的消息，但只对其它错误，而不记录SERVFAIL。与上面的级别3的情况不同，否定响应的消息要被记录。这是因为在递归的情况下，调试任何未预期的结果都可能很难。

6.2.11 lwres语句语法

这是named.conf文件中lwres语句的语法：

```
lwres {
    [ listen-on { ip_addr [port ip_port] [dscp ip_dscp] ;
      [ ip_addr [port ip_port] [dscp ip_dscp] ; ... ] } ; ]
    [ view view_name ; ]
    [ search { domain_name ; [ domain_name ; ... ] } ; ]
    [ ndots number ; ]
};
```

6.2.12 lwres语句定义和用法

lwres语句把名字服务器配置成为一个轻量级的解析服务器。（参见第5.2节。）可以有多个lwres语句配置轻量级解析服务器的不同属性。

listen-on语句指定一个轻量级解析服务能够接受请求的IPv4地址（和端口）的名单。如果没有指定端口，就使用921端口。如果这个语句被省略，请求将在地址127.0.0.1的921端口被接受。

view语句将一个轻量级解析服务绑定到DNS名字空间的视图，这样响应就会以与这个视图所匹配的普通DNS请求同样的方式被构造。如果这个语句被省略，就使用缺省视图，如果没有缺省视图，就触发一个错误。

search语句与/etc/resolv.conf 中的**search**语句等效。它提供一个追加到请求中的相对名字的域名列表。

ndots语句与/etc/resolv.conf中的**ndots**语句等效。它指示在一个相对域名中的点的最小数目，将在追加查找路径元素之前作一个精确查找。

6.2.13 masters语句语法

```
masters name [port ip_port] [dscp ip_dscp] { ( masters_list |
    ip_addr [port ip_port] [key key] ) ; [...] };
```

6.2.14 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。

6.2.15 options语句语法

这是named.conf文件中**options**语句的语法:

```
options {
    [ attach-cache cache_name; ]
    [ version version_string; ]
    [ hostname hostname_string; ]
    [ server-id server_id_string; ]
    [ directory path_name; ]
    [ geoip-directory path_name; ]
    [ key-directory path_name; ]
    [ managed-keys-directory path_name; ]
    [ named-xfer path_name; ]
    [ tkey-gssapi-keytab path_name; ]
    [ tkey-gssapi-credential principal; ]
    [ tkey-domain domainname; ]
    [ tkey-dhkey key_name key_tag; ]
    [ cache-file path_name; ]
    [ dump-file path_name; ]
    [ bindkeys-file path_name; ]
    [ secroots-file path_name; ]
    [ session-keyfile path_name; ]
    [ session-keyname key_name; ]
    [ session-keyalg algorithm_id; ]
    [ memstatistics yes_or_no; ]
    [ memstatistics-file path_name; ]
    [ pid-file path_name; ]
    [ recursing-file path_name; ]
    [ request-sit yes_or_no; ]
    [ statistics-file path_name; ]
    [ zone-statistics full | terse | none; ]
    [ auth-nxdomain yes_or_no; ]
    [ deallocate-on-exit yes_or_no; ]
    [ dialup dialup_option; ]
```

```

[ fake-iquery yes_or_no; ]
[ fetch-glue yes_or_no; ]
[ flush-zones-on-shutdown yes_or_no; ]
[ has-old-clients yes_or_no; ]
[ host-statistics yes_or_no; ]
[ host-statistics-max number; ]
[ minimal-responses yes_or_no; ]
[ multiple-cnames yes_or_no; ]
[ notify yes_or_no | explicit | master-only; ]
[ recursion yes_or_no; ]
[ request-nsid yes_or_no; ]
[ rfc2308-type1 yes_or_no; ]
[ use-id-pool yes_or_no; ]
[ maintain-ixfr-base yes_or_no; ]
[ ixfr-from-differences (yes_or_no | master | slave); ]
[ dnssec-enable yes_or_no; ]
[ dnssec-validation (yes_or_no | auto); ]
[ dnssec-lookaside ( auto |
                    no |
                    domain trust-anchor domain ); ]
[ dnssec-must-be-secure domain yes_or_no; ]
[ dnssec-accept-expired yes_or_no; ]
[ forward ( only | first ); ]
[ forwarders { [ ip_addr [port ip_port] [dscp ip_dscp] ; ... ] }; ]
[ dual-stack-servers [port ip_port] [dscp ip_dscp] {
    ( domain_name [port ip_port] [dscp ip_dscp] |
      ip_addr [port ip_port] [dscp ip_dscp]) ;
  ... }; ]
[ check-names ( master | slave | response )
  ( warn | fail | ignore ); ]
[ check-dup-records ( warn | fail | ignore ); ]
[ check-mx ( warn | fail | ignore ); ]
[ check-wildcard yes_or_no; ]
[ check-integrity yes_or_no; ]
[ check-mx-cname ( warn | fail | ignore ); ]
[ check-srv-cname ( warn | fail | ignore ); ]
[ check-sibling yes_or_no; ]
[ check-spf ( warn | ignore ); ]
[ allow-new-zones { yes_or_no }; ]
[ allow-notify { address_match_list }; ]
[ allow-query { address_match_list }; ]
[ allow-query-on { address_match_list }; ]
[ allow-query-cache { address_match_list }; ]
[ allow-query-cache-on { address_match_list }; ]
[ allow-transfer { address_match_list }; ]
[ allow-recursion { address_match_list }; ]
[ allow-recursion-on { address_match_list }; ]
[ allow-update { address_match_list }; ]
[ allow-update-forwarding { address_match_list }; ]
[ update-check-ksk yes_or_no; ]
[ dnssec-update-mode ( maintain | no-resign ); ]
[ dnssec-dnskey-kskonly yes_or_no; ]
[ dnssec-loadkeys-interval number; ]
[ dnssec-secure-to-insecure yes_or_no ;]
[ try-tcp-refresh yes_or_no; ]
[ allow-v6-synthesis { address_match_list }; ]
[ blackhole { address_match_list }; ]
[ no-case-compress { address_match_list }; ]
[ use-v4-udp-ports { port_list }; ]

```

```

[ avoid-v4-udp-ports { port_list }; ]
[ use-v6-udp-ports { port_list }; ]
[ avoid-v6-udp-ports { port_list }; ]
[ listen-on [ port ip_port ] [dscp ip_dscp] { address_match_list }; ]
[ listen-on-v6 [ port ip_port] [dscp ip_dscp]
{ address_match_list }; ]
[ query-source ( ( ip4_addr | * )
  [ port ( ip_port | * ) ]
  [ dscp ip_dscp] |
  [ address ( ip4_addr | * ) ]
  [ port ( ip_port | * ) ] )
  [ dscp ip_dscp] ; ]
[ query-source-v6 ( ( ip6_addr | * )
  [ port ( ip_port | * ) ]
  [ dscp ip_dscp] |
  [ address ( ip6_addr | * ) ]
  [ port ( ip_port | * ) ] )
  [ dscp ip_dscp] ; ]
[ use-queryport-pool yes_or_no; ]
[ queryport-pool-ports number; ]
[ queryport-pool-updateinterval number; ]
[ max-transfer-time-in number; ]
[ max-transfer-time-out number; ]
[ max-transfer-idle-in number; ]
[ max-transfer-idle-out number; ]
[ tcp-clients number; ]
[ reserved-sockets number; ]
[ recursive-clients number; ]
[ serial-query-rate number; ]
[ serial-queries number; ]
[ tcp-listen-queue number; ]
[ transfer-format ( one-answer | many-answers ); ]
[ transfers-in number; ]
[ transfers-out number; ]
[ transfers-per-ns number; ]
[ transfer-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ transfer-source-v6 (ip6_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ alt-transfer-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ alt-transfer-source-v6 (ip6_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ use-alt-transfer-source yes_or_no; ]
[ notify-delay seconds ; ]
[ notify-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ notify-source-v6 (ip6_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ notify-to-soa yes_or_no ; ]
[ also-notify { ip_addr
  [port ip_port] [dscp ip_dscp] [key keyname] ;
  [ ip_addr [port ip_port] [dscp ip_dscp] [key keyname] ; ... ] }; ]
[ max-ixfr-log-size number; ]
[ max-journal-size size_spec; ]
[ coresize size_spec ; ]
[ datasize size_spec ; ]
[ files size_spec ; ]
[ stacksize size_spec ; ]
[ cleaning-interval number; ]
[ heartbeat-interval number; ]
[ interface-interval number; ]
[ statistics-interval number; ]
[ topology { address_match_list }];
[ sortlist { address_match_list }];

```

```

[ rrset-order { order_spec ; [ order_spec ; ... ] } ];
[ lame-ttl number; ]
[ max-ncache-ttl number; ]
[ max-cache-ttl number; ]
[ max-zone-ttl number ; ]
[ sig-validity-interval number [number] ; ]
[ sig-signing-nodes number ; ]
[ sig-signing-signatures number ; ]
[ sig-signing-type number ; ]
[ min-roots number; ]
[ use-ixfr yes_or_no ; ]
[ provide-ixfr yes_or_no; ]
[ request-ixfr yes_or_no; ]
[ treat-cr-as-space yes_or_no ; ]
[ min-refresh-time number ; ]
[ max-refresh-time number ; ]
[ min-retry-time number ; ]
[ max-retry-time number ; ]
[ port ip_port; ]
[ dscp ip_dscp] ;
[ additional-from-auth yes_or_no ; ]
[ additional-from-cache yes_or_no ; ]
[ random-device path_name ; ]
[ max-cache-size size_spec ; ]
[ match-mapped-addresses yes_or_no; ]
[ filter-aaaa-on-v4 ( yes_or_no | break-dnssec ); ]
[ filter-aaaa-on-v6 ( yes_or_no | break-dnssec ); ]
[ filter-aaaa { address_match_list }; ]
[ dns64 ipv6-prefix {
    [ clients { address_match_list }; ]
    [ mapped { address_match_list }; ]
    [ exclude { address_match_list }; ]
    [ suffix IPv6-address; ]
    [ recursive-only yes_or_no; ]
    [ break-dnssec yes_or_no; ]
}; ];
[ dns64-server name ]
[ dns64-contact name ]
[ preferred-glue ( A | AAAA | NONE ); ]
[ edns-udp-size number; ]
[ max-udp-size number; ]
[ max-rsa-exponent-size number; ]
[ root-delegation-only [ exclude { namelist } ] ; ]
[ querylog yes_or_no ; ]
[ disable-algorithms domain { algorithm;
                                [ algorithm; ] }; ]
[ disable-ds-digests domain { digest_type;
                              [ digest_type; ] }; ]
[ acache-enable yes_or_no ; ]
[ acache-cleaning-interval number; ]
[ max-achache-size size_spec ; ]
[ clients-per-query number ; ]
[ max-clients-per-query number ; ]
[ max-recursion-depth number ; ]
[ max-recursion-queries number ; ]
[ masterfile-format
    (text|raw|map) ; ]
[ empty-server name ; ]
[ empty-contact name ; ]

```

```

[ empty-zones-enable yes_or_no ; ]
[ disable-empty-zone zone_name ; ]
[ zero-no-soa-ttl yes_or_no ; ]
[ zero-no-soa-ttl-cache yes_or_no ; ]
[ resolver-query-timeout number ; ]
[ deny-answer-addresses { address_match_list } [ except-from { namelist } ]; ]
[ deny-answer-aliases { namelist } [ except-from { namelist } ]; ]
[ prefetch number [number] ; ]

[ rate-limit {
  [ responses-per-second number ; ]
  [ referrals-per-second number ; ]
  [ nodata-per-second number ; ]
  [ nxdomains-per-second number ; ]
  [ errors-per-second number ; ]
  [ all-per-second number ; ]
  [ window number ; ]
  [ log-only yes_or_no ; ]
  [ qps-scale number ; ]
  [ ipv4-prefix-length number ; ]
  [ ipv6-prefix-length number ; ]
  [ slip number ; ]
  [ exempt-clients { address_match_list } ; ]
  [ max-table-size number ; ]
  [ min-table-size number ; ]
} ; ]

[ response-policy {
  zone zone_name
  [ policy (given | disabled | passthru | drop |
            nxdomain | nodata | cname domain) ]
  [ recursive-only yes_or_no ]
  [ max-policy-ttl number ]
  [ break-dnssec yes_or_no ]
  [ min-ns-dots number ]
  [ qname-wait-recurse yes_or_no ]
  ; [...]
} ; ]
};

```

6.2.16 options语句定义和用法

options语句设置BIND所使用的全局选项。这个语句只能在一个配置文件中出现一次。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 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**, **cleaning-interval**, **dnssec-accept-expired**, **dnssec-validation**, **max-cache-ttl**, **max-ncache-ttl**, **max-cache-size**, and **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 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将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。

geoip-directory 为GeoIP初始化指定包含GeoIP.dat 数据库文件的目录。缺省时，这个选项未设置，GeoIP支持会使用libGeoIP的内建目录。（关于**geoip** ACL更详细的内容，参见第6.2.2节。）

key-directory 当进行安全区的动态更新时，应该从这个目录能够找到DNSSEC公钥和私钥文件，它与当前工作目录不同。（注意这个选项对如同**bind.keys**, **rndc.key**或**session.key**这样包含非DNSSEC密钥的文件的文件的路径时无效。）

managed-keys-directory 指定目录，其中保存着跟踪被管理DNSSEC密钥的文件。缺省时，它就是工作目录。

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，受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SHA256 HASH的值，再加上后缀**.mkeys**。

named-xfer 本选项已被废除。它是用于BIND 8中指定**named-xfer**程序的路径。在BIND 9中，不再需要单独的**named-xfer**程序；其功能已经被内建在名字服务器中。

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**，将允许使用与指定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”。如果你使用了GSS-TSIG，必须定义这个变量，除非你使用tkey-gssapi-keytab指定了一个特定的keytab。

tkey-dhkey 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，密钥名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 指定服务器在收到rndc dumpdb命令时，转储数据到文件的路径。如果未指定，缺省为named_dump.db。

memstatistics-file 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。

pid-file 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。

recurring-file 指定服务器在通过rndc recurring命令指定转储当前递归请求到文件的路径。如果未指定，缺省为named.recurring文件。

statistics-file 指定服务器在收到rndc stats命令时，追加统计数据文件的路径。如果未指定，缺省为服务器当前目录下的named.stats文件。这个文件的格式由第6.4.0.1节描述。

bindkeys-file 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-lookaside和dnssec-validation的讨论。如果未指定，缺省是/etc/bind.keys。

secreots-file 在收到rndc secreots指令后，服务器转储安全根的目的文件的路径名。如果未指定，缺省为named.secreots。

session-keyfile 存放一个TSIG会话密钥的文件的路径名，这个密钥是由named所生成，用于nsupdate -l。如果未指定，缺省是/var/run/named/session.key。（参见第6.2.28.4节，特别的，对update-policy语句的local选项的讨论有关于这个特征的更多信息。）

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

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

random-device 服务器所用的熵源。熵的主要需求是DNSSEC操作，例如TKEY事务和签名区的动态更新。这个选项指定一个读出熵的设备（或文件）。如果是一个文件，当要求熵的操作在文件耗尽时将会失败。如果未指定，缺省值为/dev/random（或等效物），如果后者存在的话，如果不存在则为none。**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。缺省是不优先任何类型（NONE）。

root-delegation-only 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，并且如果请求的名字没有找到一个CNAME记录时不能转换为NXDOMAIN响应。

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。

与之类似的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 关掉在指定名下的指定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。

disable-ds-digests 在指定的名字及之下的名字关闭指定的DS/DLV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-lookaside 当其被设置后，**dnssec-lookaside**提供一个校验器，后者带有一个可选的方法来校验一个区的顶端的DNSKEY记录。当一个DNSKEY位于一个由最深的**dnssec-lookaside**所指定的域中或之下，并且普通的DNSSEC校验留下了不可信的密钥，则信任锚点将会被添加到密钥名字之后，同时一个DLV记录将会被查找以检查其是否可以校验这个密钥。如果DLV记录校验了一个DNSKEY（与一个DS记录所做的方式相似），DNSKEY资源记录被视作可信任的。

如果**dnssec-lookaside**被设置为**auto**，就会使用内置DLV域和信任锚点的缺省值，同时用一个内置密钥来验证。

如果**dnssec-lookaside**被设置为**no**，就不使用**dnssec-lookaside**。

缺省的DLV密钥存放在文件bind.keys中；如果**dnssec-lookaside**被设为**auto**，**named**在启动时会装载这个密钥。在BIND 9安装时，会同时安装这个文件的一个拷贝，并且通常为发行日期。如果DLV密钥过期，可以从<https://www.isc.org/solutions/dlv/> <<https://www.isc.org/solutions/dlv/>>下载一个新的bind.keys拷贝。

（为避免找不到bind.keys带来的问题，当前的密钥也可以被编译进**named**。依赖于这个措施是不推荐的，因为，当DLV密钥过期时，它要求使用一个新的密钥重新编译**named**。）

注意: **named** 仅仅装载 `bind.keys` 中的某些特定的密钥: 即DLV区和DNS根区的密钥。这个文件不能用来存放其它区的密钥。

dnssec-must-be-secure 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**, 允许不安全回答的普通的DNSSEC校验申请将会被接受。指定的域必须在一个**trusted-key**或**managed-keys**语句之下, 或者**dnssec-lookaside**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。

使用合成的CNAME为前缀创建一个附加的反向IP6.ARPA区, 提供一个从IP6.ARPA名字到对应的IN-ADDR.ARPA名字的映射。CNAMEs. **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。这些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**none**。

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-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 (参见第6.2.28.4节), 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, **named**, 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。)

max-zone-ttl 指定一个最大可能的TTL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。)

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）。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。参见第6.4.0.1节。

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。

6.2.16.1 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**。

当前要支持**automatic-interface-scan**，操作系统需要支持路由套接字（routing sockets）。

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

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**；这是从BIND 8 开始改变的。如果你使用非常旧的DNS软件，你可能需要将其设为**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 在退出时写内存统计到**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，拨号连接是由服务器的流量所引发。根据区类型的不同，这种方式有不同的效果，并且缩短了区维护，以使所有动作发生在一个短的周期之内，每一个**heartbeat-interval**进行一次，并希望在一次连接中完成。它也抑制了一些通常的区维护流量。缺省为**no**。

dialup选项也可在**view**和**zone**语句中指定，在此情况它将覆盖全局**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**仅仅发送NOTIFY消息，使用**notify-passive**发送NOTIFY消息并禁止普通刷新请求，使用**refresh**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**禁止普通刷新处理来进行细粒度控制。

dialup模式	普通刷新	心跳刷新	心跳通知
no (缺省)	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。

fake-iquery 在BIND 8中，这个选项打开对作废的DNS请求类型IQUERY的模拟。BIND 9不再支持IQUERY模拟。

fetch-glue 这个选项被废弃了。在BIND 8中，**fetch-glue yes**使服务器在一个响应中构造附加数据部份时试图去取得它所没有的粘着记录。这个在现在被认为是一个坏主意，BIND 9不再这样做了。

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**。

has-old-clients 这个选项在BIND 8中被错误实现了，并在BIND 9中被忽略了。要取得**has-old-clients yes**的预想效果，使用两个分开的选项**auth-nxdomain yes**和**rfc2308-type1 no**来替代。

host-statistics 在BIND 8中，这个选项打开一个统计开关，它允许对与此名字服务器相互影响的所有主机进行统计。在BIND 9中没有实现。

maintain-ixfr-base 这个选项被废弃了。它用于BIND 8中，以决定是否为增量区传送保留一个事务日志。BIND 9只要可能，随时保留着一个事务日志。如果你需要关掉对外的增量区传送，使用**provide-ixfr no**。

minimal-responses 如果为**yes**，在产生响应时，服务器将只将被请求的记录添加到权威和附加数据部份（例如：授权，否定响应）。这可以增加服务器的性能。缺省为**no**。

multiple-cnames 这个选项是用于BIND 8中，以允许一个域名可以含有与DNS标准冲突的多个CNAME记录。BIND 9.2 之后总是按照标准严格地对主文件和动态更新增强了CNAME规则。

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，参见第4.1节。消息发向区的NS记录（除SOA 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，你可能想要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。参见上面的**fetch-glue**。

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**resolver**分类，级别为**info**。缺省是**no**。

request-sit 如果为**yes**，请求中会设置一个SIT（Source Identity Token，源特征符号）EDNS选项。如果解析器之前与服务器通信过，之前返回的SIT会被发送。这个用于让服务器判断这个解析器之前是否来访问过。一个发送正确SIT的解析器会被假设不是发送一个伪造源地址请求的**off-path**攻击者；这个请求就不太可能被当成一个反射/放大攻击，因此发送一个正确SIT选项的解析器就不会遭遇到响应率限制（RRL）。不发送一个正确SIT选项的解析器可能被通过**nosit-udp-size**选项限制成接收更小的响应。

sit-secret 如果设置，这是一个用于在一个任意播集群内生成和验证源特征符号EDNS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AES128为128位，对SHA1为160位，对SHA256为256位。

rfc2308-type1 将这个设置为**yes**将导致服务器发送NS记录连同SOA记录作为否定答案。缺省为**no**。

注意



BIND 9还没有实现。

use-id-pool 这个选项被废弃了。BIND 9总是从一个池中申请请求的ID。

use-ixfr 此选项被废除了。如果你需要关掉到一个或多个服务器的IXFR，参见第6.2.18节中**provide-ixfr**选项的相关信息。也可参见第4.3节。

provide-ixfr 参见第6.2.18节中对**provide-ixfr**的描述。

request-ixfr 参见第6.2.18节中对**request-ixfr**的描述。

treat-cr-as-space 这个选项用于BIND 8中，使服务器将回车（“**\r**”）字符以空格或tab字符一样的方式对待，以帮助在UNIX系统上装载在NT或DOS主机上生成的区文件。在BIND 9中，UNIX的“**\n**”和NT/DOS的“**\r\n**”换行都会被接受，这个选项就被忽略了。

additional-from-auth, additional-from-cache 这些选项控制权威服务器在回答有附加数据的请求，或者在跟随CNAME和DNAME链时的行为。

当这两个选项都被设置为**yes**（缺省值）并且请求由权威数据（配置在服务器中的一个区）所应答时，响应的附加数据部份将会填充来自其它权威区和缓存的数据。在某些情况下这是不受欢迎的，例如，在关心缓存的正确性时，或者在服务器的辅区可能被不信任的第三方所添加和修改时。另外，在可能浪费额外的请求来解析那些可能被提供在附加部份的名字时，避免对这些附加的数据的搜索将会加速服务器的操作。

例如，如果一个请求问到主机foo.example.com的MX记录，而找到的记录是“MX 10 mail.example.net”，通常mail.example.net的地址记录（A和AAAA）也会被提供，如果服务器知道的话，即使它们并不在example.com区中。将这些选项设置为**no**会关掉这个行为，并使服务器只在它要回应的区中查找附加记录。

这些选项目的是应用于只作权威的服务器，或只作权威的视图。如果企图将它们设为**no**而不同时指定**recursion no**将会导致服务器忽略这个选项并在日志中记录一个警告消息。

指定**additional-from-cache no**实际上不止是在查找附加数据部份时关掉使用缓存，而且在查找回答部份时也不使用缓存。这通常是只作权威服务器的所期望的行为，其中的缓存数据的正确性是有争议的。

当一个名字服务器收到一个请求，要求查询不在其所服务的顶点之下的一个名字时，它通常回答一个“向上引用”，指到根服务器，或者指到被请求名字的众所周知的上级服务器。由于在向上引用中的数据来自于缓存，服务器在**additional-from-cache no**被设定时将不能够提供向上引用。作为替代，它将对这样的请求以REFUSED进行响应。这不会带来任何问题，因为向上引用并不是解析过程所要求的。

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。

filter-aaaa-on-v4 这个选项仅在BIND 9用“configure”命令行中的**--with-filter-aaaa**选项编译时可用。其目的是通过不将IPv6地址返回给没有IPv6接入的DNS客户端来帮助从IPv4到IPv6的迁移。除非特别需要，这个是不推荐的。缺省是**no**。**filter-aaaa-on-v4**选项也可在**view**语句中指定，用于覆盖全局**filter-aaaa-on-v4**选项。

如果为**yes**，DNS客户端在**filter-aaaa**中的一个IPv4地址，并且如果响应中不包含DNSSEC签名，这时所有的AAAA记录都会从响应中删除。这个过滤应用在所有响应中而不仅仅是权威响应。

如果为**break-dnssec**，即使打开了DNSSEC也会删除AAAA记录。如同名字所暗示的，这会地址不校验响应，因为DNSSEC协议被设计成能检测删除。

这个机制可能会错误地导致其它服务器不向其客户端给出AAAA记录。一个具有IPv6和IPv4双栈地址的递归服务器通过IPv4向具有这个机制的权威服务器发起请求，即使其客户端使用IPv6，也有可能被删去AAAA记录。

这个机制应用于权威记录，也应用于非权威记录。一个使用IPv4并且不允许递归请求的客户端可能会错误地收到AAAA记录，因为其服务器不被允许查找A记录。

有些AAAA记录会作为粘着（glue）记录返回给IPv4客户端。同时也做服务器的IPv4客户端可能错误地回应通过IPv4接收到的对AAAA记录的请求。

filter-aaaa-on-v6 与**filter-aaaa-on-v4**相同，除了它过滤来自IPv6的客户端而不是IPv4的客户端的AAAA响应。要过滤所有响应，将两个选项都设置为**yes**。

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**和**slave**，这将导致**ixfr-from-differences**分别在所有的**master**或**slave**区被打开。缺省是关闭的。

multi-master 当你的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**。

dnssec-enable 在**named**中打开对DNSSEC的支持。除非设为**yes**，否则**named**的行为就象其不支持DNSSEC。缺省为**yes**。

dnssec-validation 在**named**中打开DNSSEC正确性检查。注意**dnssec-enable**也需要被设为**yes**以使此选项生效。如果设置为**no**，DNSSEC验证将被关闭。如果设置为**auto**，DNSSEC验证将被开启，

并为根区使用一个缺省的信任锚。如果设置为**yes**，DNSSEC验证将被开启，但是必须使用一个**trusted-keys**或**managed-keys**语句手工配置一个信任锚。缺省为**yes**。

dnssec-accept-expired 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 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。

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**master**区，缺省为**fail**。对**slave**区，缺省是**warn**。对从网络接收的回答（**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 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 检查MX记录是否与一个IP地址一起出现。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导致理解通配符匹配算法（RFC 1034）的失败。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 对主区执行加载后的区完整性检查。这将检查MX和SRV记录所提到的地址（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记录做发送者策略框架的发布已被弃用，即从使用TXT记录到SPF记录的迁移一样被放弃一样。打开这个选项在已存在一个SPF记录时，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**抑制。

check-mx-cname 如果设置了**check-integrity**，遇到指向CNAME名字的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名字的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 当给SOA请求返回权威的消极响应时，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 当为SOA请求返回缓存的消极响应时，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。

dnssec-loadkeys-interval 当一个区配置有**auto-dnssec maintain;**时，将会周期性地检查其密钥库，以查看是否有新的密钥添加或者任何现存密钥的计时元数据被修改（参见**dnssec-keygen(8)**和**dnssec-settime(8)**）。**dnssec-loadkeys-interval**选项设置自动检查库的频率，以分钟计。缺省为60（1小时），最小为1（1分钟），最大为1440（24小时）；任何更大的值将被静默地减小。

try-tcp-refresh 如果UDP请求失败，试图使用TCP刷新区。为BIND 8兼容性，缺省是yes。

dnssec-secure-to-insecure 通过删除所有DNSKEY记录，允许一个区从安全状态迁移到不安全状态。缺省为no。

6.2.16.2 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。值为first时，即其缺省值，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，服务器再自行查找回答。如果指定only，服务器只请求转发服务器。

forwarders 指定用于转发的IP地址。缺省时空列表（不转发）。

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。你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，参见第6.2.27节。

6.2.16.3 双栈服务器

双栈服务器是服务器的最后手段，用于解决这样的问题，即由于主机缺乏IPv4或IPv6可达性。

dual-stack-servers 指定机器的主机名或者地址，它可以通过IPv4和IPv6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方式上能够被解析到。如果机器是双栈的，**dual-stack-servers**就没有效，除非到某个传输方式的访问被命令行禁止掉（如**named -4**）。

6.2.16.4 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。参见第6.1.1节以获得如何指定IP地址列表的详细内容。

allow-notify 指定允许哪些主机可以通知本服务器，作为一个辅服务器，除了区主服务器之外关于区改变的信息。**allow-notify**也可以在**zone**语句中指定，这时它会覆盖**options allow-notify**语句。它仅对一个辅区有意义。如果未指定，缺省是仅处理来自一个区的主服务器的通知消息。

allow-query 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。

注意



allow-query-cache现在用于指定对缓存的访问。

allow-query-on 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。

注意



allow-query-cache用于指定对缓存的访问。

allow-query-cache 指定允许哪些主机可以从缓存中获取答案。如果未设置**allow-query-cache**而设置了**allow-recursion**就使用它，否则如果设置了**allow-query**就使用它，除非设置了**recursion no;**，在这种情况下使用**none;**，否则使用缺省的 (**localnets; localhost;**)。

allow-query-cache-on 指定哪些本机地址可以从缓存给出答案，如果未指定，缺省允许在任何地址上的对缓存的请求，**localnets**和**localhost**。

allow-recursion 指定允许哪些主机可以通过本服务器进行递归查询。如果未设置**allow-recursion**而设置了**allow-query-cache**就使用它，否则如果设置了**allow-query**就使用它，否则使用缺省的 (**localnets;localhost;**)。

allow-recursion-on 指定哪些本机地址可以接受递归请求。如果未指定，缺省是允许在所有地址上的递归请求。

allow-update 指定允许哪些主机可以对主区提交动态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；更详细的内容参见第7.3节。

allow-update-forwarding 指定允许哪些主机可以对能够转发到主区的辅区提交动态DNS更新。缺省是{ none; }, 表示不执行更新转发操作。要允许更新转发, 指定**allow-update-forwarding { any; };**。指定{ none; }或{ any; }之外的值通常是相反的意义, 因为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依赖基于不安全IP地址的访问控制暴露给攻击者; 更详细的内容参见第7.3节。

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 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**options allow-transfer**语句。如果未指定, 缺省是允许所有主机进行传送。

blackhole 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**。

filter-aaaa 指定一个**filter-aaaa-on-v4**所应用到的地址的列表。缺省为**any**。

no-case-compress 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, 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: 如果一个其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, 重复客户端匹配这个ACL。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时间。缺省值和最小值是10, 最大值是30。将其设置为0将会导致使用缺省值。

6.2.16.5 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; };
```

将使服务器监听在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。

当指定

```
{ any; }
```

作为**listen-on-v6**选项的**address-match-list**时，如果操作系统对IPv6有足够的API支持，服务器不会象对IPv4地址一样将一个单独的套接字绑定到其每个IPv6接口地址上（特别是在其遵从RFC 3493和RFC 3542时）。作为替代，它会监听在IPv6的通配地址上。如果系统仅仅对IPv6有不完全的API支持，其行为就与IPv4一样。

可以指定一个特殊的IPv6地址列表，在这样的情况下，服务器监听在每个特定地址的单独的套接字上，而不用考虑系统是否有足够的API支持。在**listen-on-v6**中指定的IPv4地址将被忽略，并在日志中记录一条警告。

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; };
```

将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; };
```

6.2.16.6 请求地址

如果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**将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; };
```

注意：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。

注意：**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 {};
```

注意：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。

注意



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。

注意



Solaris 2.5.1 及之前的版本不支持设置TCP套接字的源地址。

注意



参见**transfer-source**和**notify-source**。

6.2.16.7 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 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 进来数据的区传送如果超过这么多分钟将被终止。缺省为120分钟（2小时）。最大值为28天（40320分钟）。

max-transfer-idle-in 进来数据的区传送在这么多分钟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 出去数据的区传送如果超过这么多分钟将被终止。缺省为120分钟（2小时）。最大值为28天（40320分钟）。

max-transfer-idle-out 出去数据的区传送在这么多分钟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）。

serial-query-rate 辅服务器将会周期性地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。

除了控制发出SOA刷新请求的速率，**serial-query-rate**还控制从主区和辅区发出的NOTIFY消息的速率。

serial-queries 在BIND 8中，**serial-queries**选项设置在任何时间允许未完成的最大并发序列号请求的数目。BIND 9不限制未完成的序列号请求数并忽略**serial-queries**选项。取而代之的是，它限制由**serial-query-rate**选项所定义的请求发送的速率。

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，但是仅在相对新的辅服务器中支持，如BIND 9，BIND 8.x和BIND 4.9.5之后的版本。最近的Microsoft Windows名字服务器也支持**many-answers**格式。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。

transfers-in 可以同时运行的进入的区传送的最大数目。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 可以同时运行的出去的区传送的最大数目。超过限制的区传送请求将会被拒绝。缺省值是10。

transfers-per-ns 可以同时运行的来自某个给定远程名字服务器的进入的区传送的最大数目。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**可以被覆盖成以每个服务器为基础。

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的**allow-transfer**选项以便能够进行区传送，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。

注意



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 与**transfer-source**相似，除了区传送是使用IPv6之外。

alt-transfer-source 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**。

注意



如果你不希望使用一个可替换的传送源，你应该正确设置**use-alt-transfer-source**，并且你不应该依赖于从第一个更新请求获得一个回复。

alt-transfer-source-v6 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 是否使用可替换的传送源，如果使用视图，这个选项的省为**no**，否则其缺省为**yes**（为了BIND 8的兼容性）。

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。

注意



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 与**notify-source**相似，但是以IPv6地址发送通知消息。

6.2.16.8 UDP端口列表

use-v4-udp-ports, **avoid-v4-udp-ports**, **use-v6-udp-ports**和**avoid-v6-udp-ports**指定可以用于或不能用于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，参见第6.2.16.6节。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**-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。

6.2.16.9 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。参见第6.1节的关于**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 内核转储的最大大小。缺省为**default**。

datasize 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。如果你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 服务器可以同时打开的文件的最大数目。缺省为**unlimited**。

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

6.2.16.10 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-ixfr-log-size 本选项被废除了；它被接受并忽略只是为了BIND 8的兼容性。在BIND 9中，选项**max-journal-size**执行类似的功能。

max-journal-size 设置每个日志文件（参见第4.2.1节）的最大大小。当日志文件达到指定的大小时，其中一些最旧的事物会被自动删除。允许的最大值为2G字节。缺省是**unlimited**，也表示2G字节。这也可以基于每个区来设置。

host-statistics-max 在BIND 8中，指定所要保持的主机统计条目的最大数目。在BIND 9中未实现。

recursive-clients 执行并行递归查询的最大数目，这是服务器所提供给客户端的行为。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。

tcp-clients 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100。

reserved-sockets 为TCP，标准输入输出等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，以及用于提供向外TCP请求和进入的区传送的**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 用于服务器缓存的最大内存量，以字节为单位。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**unlimited**。

tcp-listen-queue 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。

6.2.16.11 周期的内部任务

cleaning-interval 这个间隔事实上被废除了。先前，服务器将会每**cleaning-interval**分钟清理一次缓存，从中删除过期的资源记录。BIND 9现在以一个更复杂的方式管理缓存，并不再依赖于周期性地清理。因此指定这个选项对服务器的行为没有任何影响。

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，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），并且不会停止已经不存在的监听端口。

statistics-interval 名字服务器统计信息在每**statistics-interval**分钟被写入日志。缺省是60分钟。最大值是28天（40320分钟）。如果设置为0，就不将统计写入日志。

注意



BIND 9中还未实现。

6.2.16.12 拓扑

在所有其它方面都是相同的情况下，当一个服务器在一个名字服务器的列表中选择来发送请求时，它会优先选择拓扑上离它最近的。**topology**语句使用**address-match-list**，并以一个特殊的方式解释它。每个顶级列表元素被指定一个距离。非否定元素获得一个基于它们在列表中位置的距离，离表的开始处越近，在其与服务器之间的距离越短。否定匹配将被指定离服务器最大的距离。如果没有匹配，地址将会获得一个比任何非否定列表元素更远，而比任何否定元素更近的距离。例如，

```
topology {
    10/8;
    !1.2.3/24;
    { 1.2/16; 3/8; };
};
```

将最先选择网络10中的服务器，然后是网络1.2.0.0（掩码255.255.0.0）和网络3上的主机，并且不包括网络1.2.3（掩码255.255.255.0）上主机，后者是最小的优先级。

缺省拓扑是

```
topology { localhost; localnets; };
```

注意



BIND 9中还未实现**topology** 选项。

6.2.16.13 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（但可参见第6.2.16.14节中的**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）以一个**address_match_list**作为参数并且比**topology**语句更有效率地解释它（第6.2.16.12节）。每个**sortlist**中的顶级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。每个顶级列表的第一个元素（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。

一旦请求的源地址匹配，如果顶级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，第二个元素被当成**topology**语句中的**address_match_list**。每个顶级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, 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。它与BIND 4.9.x中的地址排序行为很相似。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; };
};
```

6.2.16.14 资源记录集排序

在一个回答中有多个记录时，配置响应中记录的顺序可能非常有用。**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，[第6.2.16.13节](#)。

order_spec按下列格式定义：

[class *class_name*] [type *type_name*] [name "*domain_name*"] order *ordering*

如果没有指定类，缺省为**ANY**。如果没有指定类型，缺省是**ANY**。如果没有指定名字，缺省是“*”（星号）。

合法的**ordering**值是：

fixed	以区文件中定义的顺序返回记录。
random	以随机的顺序返回记录。
cyclic	以循环轮转的顺序返回记录。
	如果BIND在编译时使用了“ -enable-fixed-rrset ”选项，资源记录集的初始顺序会与区文件中指定的一致。

例如：

```
rrset-order {
    class IN type A name "host.example.com" order random;
    order cyclic;
};
```

将使响应中任何类为IN类型为A并有“host.example.com”后缀的记录以随机的顺序返回。所有其它的记录都以轮转的顺序返回。

如果出现了多个**rrset-order**语句，它们不会组合——只有最后一个起作用。

缺省时，所有记录以随机顺序返回。

注意



在这个版本的BIND 9中，缺省时**rrset-order**语句不支持“固定”顺序。可以在编译时在“configure”命令行指定“**-enable-fixed-rrset**”打开固定顺序。

6.2.16.15 调优

lame-ttl 设置跛服务器指示的缓存秒数。0关掉缓存。（这是不推荐的。）缺省为值600（10分钟），最大值为1800（30分钟）。

Lame-ttl也控制DNSSEC验证失败被缓存的时间。如果**lame-ttl** 被设置为小于30秒，则被应用到否定缓存条目的最小值为30秒。

max-ncache-ttl 为减少网络流量和提高性能，服务器会存储“记录不存在”的响应。**max-ncache-ttl**用来设置这些响应在服务器中最大保持时间的秒数。缺省的**max-ncache-ttl**值为10800（3小时）。**max-ncache-ttl** 不能超过7天，如果超过，将会静默地被截为7天。

max-cache-ttl 设置服务器的缓存普通（正）响应的最大时间。缺省值是一周（7天）。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。

min-roots 根服务器的最小个数，一个需要查询根服务器的请求所能接受。缺省值为2。

注意



BIND 9中还未实现。

sig-validity-interval 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（第4.2节）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。The **sig-validity-interval**

sig-signing-nodes 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 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 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，但是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 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4096。

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。

如果这个初始响应超时，**named**会以普通DNS重试，如果重试成功，它就会得出这个服务器不支持EDNS的结论。在使用EDNS时足够的失败和使用普通DNS时足够的成功之后，**named**在将来同这台服务器通信时缺省使用普通DNS。（**named**将会周期性地发送EDNS请求来查看情况是否有改善。）

另外，如果初始的广告带512字节缓存的EDNS的请求成功后，**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 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4096。

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，就将得到越多的到名字服务器的TCP流量。

masterfile-format 指定区文件的格式（参见第6.3.7节）。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。这就意味着raw格式的区文件必须以**named**配置文件中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服务器为基础。

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。

max-recursion-depth 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 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。对诸如“com”和“net”等顶级域和DNS根区进行查找的请求被豁免在这个限制之外。缺省是75。

notify-delay 在发送一个区的通知消息集之间的延迟，单位为秒。缺省是五（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 在验证时接受的最大RSA指数的位数。有效值是从35到4096位。缺省值零（0）也可以接受，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“**trigger**” 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。设置一个触发TTL为零（0）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“**eligibility**” TTL：对于一个适宜预取的记录可接受的最小**original** TTL值。适宜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。缺省的适宜TTL是9。

6.2.16.16 服务器内置信息区

服务器通过在伪顶级域bind的**CHAOS**类中的一些内置区提供了一些有用的诊断信息。这些区是类**CHAOS**中内置视图（参见第6.2.25节）的一部份，与类**IN**的缺省视图是分开的。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**被设置为允许三个响应每秒。

如果你需要关掉这些区，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 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。

hostname 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，并且可以通过**gethostname()** 函数查到。这个请求的主要目的是判定在一组**anycast**服务器中是哪一台实际回应你的请求。指定**hostname none**；关掉对这个请求的处理。

server-id 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**的请求时应该报告其ID。这类请求的主要目的是判定在一组**anycast**服务器中是哪一台实际回应你的请求。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**。

6.2.16.17 内置空区

named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。这些也包含IPv6本地地址（本地分配的）的反向名字空间，IPv6链路本地地址，IPv6环回地址和IPv6未知地址。

named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

6.2.16.18 附加部份缓存

附加部份缓存，也叫做**acache**，是一个为了提高BIND 9 响应性能的内部缓存。当打开附加部份缓存时，BIND 9将为每个答案部份的资源记录而在附加部份的内容中缓存一个内部缩写。注意**acache**是一个BIND 9的内部缓存机制，与DNS的缓存服务器功能没有关系。

附加部份缓存不改变响应内容（附加部份的资源记录集的顺序除外，参见下文），但是可以极大地提高响应的性能。特别是在BIND 9作为一个带有许多授权，并且每个授权都有多个粘连资源记录的区的权威服务器的情况下。

为了从附加部份缓存获得最大的性能提升，推荐将**additional-from-cache**设置为**no**，因为当前的**acache**实现没有缩小来自DNS缓存数据的附加部份信息。

acache的一个明显的缺点是它需要更多的内存空间来存放内部缓存数据。然而，如果响应性能不明显并且内存消耗更关键时，可以通过将**acache-enable**设置为**no**来关掉**acache**机制。也可以通过使用**max-acache-size**来指定内存消耗的上限。

附加部份缓存对附加部份中的资源记录集的顺序也有一点点影响。没有**acache**的情况下，**cyclic**顺序对附加部份也有效，就象其对答案和权威部份一样。然而，附加部份缓存将会在它第一次缓存一个资源记录集时固定其顺序，在后续的响应中将保持同样的顺序，而忽略**rrset-order**的设置。这样做的影响应该较小，另外，由于附加部份的一个资源记录集典型情况下只包含很小数目的资源记录（在许多情况它只包含一个资源记录），这样情况下顺序就没有什么影响了。

以下是与**acache**相关的选项的概括。

acache-enable 如果设为**yes**，打开附加部份缓存。缺省值是**no**。

acache-cleaning-interval 服务器将去掉陈旧的缓存条目，这个操作基于LRU算法，每**acache-cleaning-interval**分钟一次。缺省是60 分钟。如果设为0，不做定期清理。

max-acache-size 服务器用于**acache**的内存的最大字节数。当**acache**中的数据量达到这个限制时，服务器将会主动地进行清理以保证不超过限额。在有多视图的服务器上，**acache**的限额单独应用在各个视图上。缺省是16M。

6.2.16.19 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**所给出的address_match_list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**选项所给出的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将会被默默地忽略。

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，返回一个你自己网络内的IP地址或者一个你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，会让攻击者访问你的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<http://portal.acm.org/citation.cfm?id=1315245.1315298> <<http://portal.acm.org/citation.cfm?id=1315245.1315298>> 处的论文。

例如，如果你拥有一个域名“example.net”并且你的内部网络使用192.0.2.0/24的IPv4前缀，你可能设定如下规则：

```
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你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”名字映射到你“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就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，除非你很清楚没有其它的选择，并且攻击对你的应用而言是一个现实的威胁。

如果你想在地址段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。过滤包含这类地址的DNS记录可能会破坏这类应用。

6.2.16.20 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黑名单。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; };**

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32；超过那个数就是一个配置错误。

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-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-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，但是如同IN-ADDR.ARPA中的反向表示的。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64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触发器用A和AAAA资源记录集中的IP地址匹配域名，后者可以与NSDNAME策略记录比对。

RPZ-NSIP NSIP触发器被编码成类似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。对非顶级域，**min-ns-dots**的缺省值是1。

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：

- 选择区中最先在**response-policy**选项中出现的触发器记录。
- 在单个区中，按CLIENT-IP, QNAME, IP, NSDNAME, NSIP的顺序使用触发器。
- 在NSDNAME触发器中，优先使用匹配DNSSEC顺序下最小名字的触发器。
- 在IP或者NSIP触发器之间，优先选择具有最长前缀的触发器。
- 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 白名单策略由一个目标为**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 黑名单策略由一个目标为**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 “slip”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 资源记录的空集由目标为通配顶级域（*）的CNAME指定。它重写响应为NODATA或ANCOUNT=1。

Local Data 一个普通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。这个特殊形式的目的是在围墙花园（原文：walled garden）内的权威DNS服务器上作请求日志。

一个使用由另一个组织所提供的RPZ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。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 站位符策略表示“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，但是记录如果策略区未被关闭时可能做的事情。对DNS请求的应答将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 使用对应的每记录策略覆盖。

CNAME domain 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，这个缺省行为可能花费大量时间。当递归查询不能改变一个非错响应时，**qname-wait-recurse no**选项覆盖缺省行为。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。

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改变那个最大值。

例如，你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      .

; blacklist and whitelist some DNS clients
112.zz.2001.rpz-client-ip CNAME      rpz-drop.
8.0.0.0.127.rpz-client-ip CNAME      rpz-drop.

; 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。

6.2.16.21 响应比率限制

过多的几乎同样的UDPreponses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。可以通过发送短截断（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（缺省基于**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），例如来自反垃圾黑名单。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。这个用于控制random.wild.example.com（译注：随机通配符域名）的泛洪。

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小截断（TC=1）响应。“slipped”响应的大小较小且缩小了频率，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“slip”：因比率限制而不发出被截断的响应，所有响应都不丢弃。值为1使每个响应都slip；值为2到10之间使每N个响应slip一次。一些错误响应包括REFUSE和SERVFAIL不能被替代成截断的响应，and are instead leaked以slip比率。

（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。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子句免出比率限制。

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 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。Attacks that justify ignoring the contents of DNS responses are likely to be attacks on the DNS server itself. 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。

6.2.17 server语句语法

```
server ip_addr[/prefixlen] {
    [ bogus yes_or_no ; ]
    [ provide-ixfr yes_or_no ; ]
    [ request-ixfr yes_or_no ; ]
```



```
[ request-nsid yes_or_no ; ]
[ request-sit yes_or_no ; ]
[ edns yes_or_no ; ]
[ edns-udp-size number ; ]
[ nosit-udp-size number ; ]
[ max-udp-size number ; ]
[ transfers number ; ]
[ transfer-format ( one-answer | many-answers ) ; ]
[ keys { string ; [ string ; [...]] } ; ]
[ transfer-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ transfer-source-v6 (ip6_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ notify-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ notify-source-v6 (ip6_addr | *) [port ip_port] [dscp ip_dscp] ; ]
[ query-source [ address ( ip_addr | * ) ]
                [ port ( ip_port | * ) ] [dscp ip_dscp] ; ]
[ query-source-v6 [ address ( ip_addr | * ) ]
                  [ port ( ip_port | * ) ] [dscp ip_dscp] ; ]
[ use-queryport-pool yes_or_no; ]
[ queryport-pool-ports number; ]
[ queryport-pool-updateinterval number; ]
};
```

6.2.18 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。

如果你发现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，将会自动降格为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。

edns子句决定在与远程服务器通信时，是否企图使用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。）

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）。这个选项是用于当你知道有防火墙阻止来自**named**的大回复包时。

nosit-udp-size选项设置发送给没有有效源身份特征请求方的UDP响应的最大大小。**max-udp-size**选项可以更进一步地限制响应的大小。

服务器支持两种区传送方法。第一种，**one-answer**，使用一个DNS消息传送一个资源记录。**many-answers**将尽可能多的资源记录封装在一个消息中。**many-answers**更有效率，但它只能被BIND 9，BIND 8.X和打补丁的BIND 4.9.5 所识别。你可以在**transfer-format**选项中指定对一个服务器使用哪种方法。如果**transfer-format**未指定，将使用**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**keys**语句所定义的**key_id**，在与远程服务器通讯时，用于事务安全（TSIG，第4.5节）。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。

虽然**keys**子句的语法允许多个**key**，但是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，仅需要指定**transfer-source**。类似的，对一个IPv6远程服务器，仅需要指定**transfer-source-v6**。更多细节，参见第6.2.16.7节中对**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，仅需要指定**notify-source**。类似的，对一个IPv6远程服务器，仅需要指定**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，仅需要指定**query-source**。类似的，对一个IPv6远程服务器，仅需要指定**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。

request-sit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SIT EDNS选项。这覆盖视图或全局选项中的**request-sit**设置。**named**可能决定SIT不被远程服务器支持并且不在请求中添加一个SIT EDNS选项。

6.2.19 statistics-channels语句语法

```
statistics-channels {
    [ inet ( ip_addr | * ) [ port ip_port ]
      [allow { address_match_list } ]];
    [ inet ...; ]
};
```

6.2.20 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，给HTTP通道使用端口80。星号“*”不能用作**ip_port**。

打开一个统计通道的企图被可选的**allow**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。

依赖某一特定XML模式的应用可以请求<<http://127.0.0.1:8888/xml/v2>>获得版本2的统计XML模式或者<<http://127.0.0.1:8888/xml/v3>>获得版本3的。如果服务器支持所请求的模式，就会有响应；如果不支持，就会返回一个“page not found”错误。

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）。

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）。

6.2.21 trusted-keys语句语法

```
trusted-keys {
    string number number number string ;
    [ string number number number string ; [...]]
};
```

6.2.22 trusted-keys语句定义和用法

trusted-keys语句定义DNSSEC安全根。DNSSEC在[第4.8节](#)中描述。一个安全根被定义，当一个非授权区的公钥被公开，但是不能通过DNS获得安全，要么因为它是DNS根区，要么因为它的父区为签名。一旦一个密钥被配置成可信任密钥，它就被当作已验证和被证实为安全。解析器试图DNSSEC验证一个安全根子域的所有DNS数据。

所有**trusted-keys**中列出的密钥（和相应的区）都被认为是存在，并与父区所说无关。所有**trusted-keys**中列出的密钥的相似地是，仅仅这些密钥在验证DNSKEY资源记录集时被用到。父区的DS资源记录集不会被用到。

trusted-keys语句可以包含多个密钥条目，每个条目由密钥域名，标志，协议，算法和Base-64表示的密钥数据组成。密钥数据中的空格，制表符，换行和回车被忽略，所以配置可以分割放入多行中。

trusted-keys可以被设置在named.conf的顶级或者在一个视图内。如果它在两个地方都被设置了，它们都会被添加：在顶级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。

6.2.23 managed-keys Statement Grammar

```
managed-keys {
    name initial-key flags protocol algorithm key-data ;
    [ name initial-key flags protocol algorithm key-data ; [...]]
};
```

6.2.24 managed-keys Statement Definition and Usage

于**trusted-keys**相似，**managed-keys**语句定义DNSSEC安全根。差别是**managed-keys**可以自动保持更新，无须解析器管理员干预。

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。在**trusted-keys**语句中拥有一个旧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。这会一直持续下去，直到解析器管理员使用新的密钥更新**trusted-keys**语句。

然而，如果区被列入到**managed-keys**中，区的所有者可以预先给区增加一个“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。

一个**managed-keys**语句包含了要管理的密钥的列表，以及这些密钥在开始的时候如何被初始化的信息。当前所支持的（即对于BIND 9.7.0）唯一初始化方法是**initial-key**。这表示**managed-keys**语句必须包含一个初始化密钥的拷贝。（将来的版本可能允许密钥以其它方式初始化，从而取消这项要求。）

所以，一个**managed-keys**语句表面上与一个**trusted-keys**相似，区别出现在第二个字段，前者包含了关键字**initial-key**。区别是，在**trusted-keys**中列出的密钥可以一直被信任，直到其从**named.conf**中删除掉，而在**managed-keys**语句中列出的一个初始化密钥只被信任一次：因为它装载被管理的密钥数据库并开始RFC 5011密钥维护进程。

当**named**第一次带有一个配置在**named.conf**中的被管理密钥运行时，它直接从区顶点取得DNSKEY资源记录集，并使用**managed-keys**语句中所指定的密钥验证它。如果这个DNSKEY资源记录集是有效签名的，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，它看见了**managed-keys**语句，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**managed-keys**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。

在一个名字从**managed-keys**语句中删除之后下一次运行**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。

named只维持一个被管理的密钥数据库；所以，与**trusted-keys**不同，**managed-keys**可能仅仅被设置在**named.conf**的顶级，而不放在视图中。

在当前的实现中，被管理的密钥数据库是以一个主区文件格式存放的，文件名为**managed-keys.bind**。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，**managed-keys.bind.jnl**。随后它们会尽快被提交到主文件；在被管理的密钥数据库的情况下，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，这两个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被管理密钥。同样地，如果**dnssec-lookaside**选项被设为**auto**，**named**将会自动为区**dlv.isc.org**初始化一个被管理密钥。在两种情况下，用于初始化密钥维护进程的密钥是内建于**named**，并且能够被**bindkeys-file**中的密钥所覆盖。

6.2.25 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; ...]
    };
};
```

6.2.26 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`匹配并且它的目的地址与视图的**match-destinations**子句中的`address_match_list`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**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，缺省试验**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;
        file "example-external.db";
    };
};
```

6.2.27 zone语句语法

```

zone zone_name [class] {
    type master;
    [ allow-query { address_match_list }; ]
    [ allow-query-on { address_match_list }; ]
    [ allow-transfer { address_match_list }; ]
    [ allow-update { address_match_list }; ]
    [ update-check-ksk yes_or_no; ]
    [ dnssec-dnskey-kskonly yes_or_no; ]
    [ dnssec-loadkeys-interval number; ]
    [ update-policy local | { update_policy_rule [...] }; ]
    [ also-notify { ip_addr [port ip_port] [dscp ip_dscp] ;
                  [ ip_addr [port ip_port] [dscp ip_dscp] ; ... ] }; ]
    [ check-names (warn|fail|ignore) ; ]
    [ check-mx (warn|fail|ignore) ; ]
    [ check-wildcard yes_or_no; ]
    [ check-spf ( warn | ignore ); ]
    [ check-integrity yes_or_no ; ]
    [ dialup dialup_option ; ]
    [ file string ; ]
    [ masterfile-format (text|raw|map) ; ]
    [ journal string ; ]
    [ max-journal-size size_spec; ]
    [ forward (only|first) ; ]
    [ forwarders { [ ip_addr [port ip_port] [dscp ip_dscp] ; ... ] }; ]
    [ ixfr-base string ; ]
    [ ixfr-from-differences yes_or_no; ]
    [ ixfr-tmp-file string ; ]
    [ request-ixfr yes_or_no ; ]
    [ maintain-ixfr-base yes_or_no ; ]
    [ max-ixfr-log-size number ; ]
    [ max-transfer-idle-out number ; ]
    [ max-transfer-time-out number ; ]
    [ notify yes_or_no | explicit | master-only ; ]
    [ notify-delay seconds ; ]
    [ notify-to-soa yes_or_no; ]
    [ pubkey number number number string ; ]
    [ notify-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
    [ notify-source-v6 (ip6_addr | *) [port ip_port] [dscp ip_dscp] ; ]
    [ zone-statistics full | terse | none; ]
    [ sig-validity-interval number [number] ; ]
    [ sig-signing-nodes number ; ]
    [ sig-signing-signatures number ; ]
    [ sig-signing-type number ; ]
    [ database string ; ]
    [ min-refresh-time number ; ]
    [ max-refresh-time number ; ]
    [ min-retry-time number ; ]
    [ max-retry-time number ; ]
    [ key-directory path_name; ]
    [ auto-dnssec allow|maintain|off; ]
    [ inline-signing yes_or_no; ]
    [ zero-no-soa-ttl yes_or_no ; ]
    [ serial-update-method increment|unixtime; ]
    [ max-zone-ttl number ; ]
};

```

```

zone zone_name [class] {
    type slave;
    [ allow-notify { address_match_list }; ]
    [ allow-query { address_match_list }; ]
    [ allow-query-on { address_match_list }; ]
    [ allow-transfer { address_match_list }; ]
    [ allow-update-forwarding { address_match_list }; ]
    [ dnssec-update-mode ( maintain | no-resign ); ]
    [ update-check-ksk yes_or_no; ]
    [ dnssec-dnskey-kskonly yes_or_no; ]
    [ dnssec-loadkeys-interval number; ]
    [ dnssec-secure-to-insecure yes_or_no ; ]
    [ try-tcp-refresh yes_or_no; ]
    [ also-notify [port ip_port] [dscp ip_dscp] { ( masters_list | ip_addr
                                                [port ip_port]
                                                [dscp ip_dscp]
                                                [key key] ) ; [...] }; ]
    [ check-names (warn|fail|ignore) ; ]
    [ dialup dialup_option ; ]
    [ file string ; ]
    [ masterfile-format (text|raw|map) ; ]
    [ journal string ; ]
    [ max-journal-size size_spec; ]
    [ forward (only|first) ; ]
    [ forwarders { [ ip_addr [port ip_port] [dscp ip_dscp] ; ... ] }; ]
    [ ixfr-base string ; ]
    [ ixfr-from-differences yes_or_no; ]
    [ ixfr-tmp-file string ; ]
    [ maintain-ixfr-base yes_or_no ; ]
    [ masters [port ip_port] [dscp ip_dscp] { ( masters_list | ip_addr
                                                [port ip_port]
                                                [dscp ip_dscp]
                                                [key key] ) ; [...] }; ]
    [ max-ixfr-log-size number ; ]
    [ max-transfer-idle-in number ; ]
    [ max-transfer-idle-out number ; ]
    [ max-transfer-time-in number ; ]
    [ max-transfer-time-out number ; ]
    [ notify yes_or_no | explicit | master-only ; ]
    [ notify-delay seconds ; ]
    [ notify-to-soa yes_or_no; ]
    [ pubkey number number number string ; ]
    [ transfer-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
    [ transfer-source-v6 (ip6_addr | *) [port ip_port] [dscp ip_dscp] ; ]
    [ alt-transfer-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
    [ alt-transfer-source-v6 (ip6_addr | *)
                            [port ip_port]
                            [dscp ip_dscp] ; ]
    [ use-alt-transfer-source yes_or_no; ]
    [ notify-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
    [ notify-source-v6 (ip6_addr | *) [port ip_port] [dscp ip_dscp] ; ]
    [ zone-statistics full | terse | none; ]
    [ sig-validity-interval number [number] ; ]
    [ sig-signing-nodes number ; ]
    [ sig-signing-signatures number ; ]
    [ sig-signing-type number ; ]
    [ database string ; ]
    [ min-refresh-time number ; ]
    [ max-refresh-time number ; ]

```

```

[ min-retry-time number ; ]
[ max-retry-time number ; ]
[ key-directory path_name; ]
[ auto-dnssec allow|maintain|off; ]
[ inline-signing yes_or_no; ]
[ multi-master yes_or_no ; ]
[ zero-no-soa-ttl yes_or_no ; ]
};

zone zone_name [class] {
    type hint;
    file string ;
    [ delegation-only yes_or_no ; ]
    [ check-names (warn|fail|ignore) ; ] // Not Implemented.
};

zone zone_name [class] {
    type stub;
    [ allow-query { address_match_list }; ]
    [ allow-query-on { address_match_list }; ]
    [ check-names (warn|fail|ignore) ; ]
    [ dialup dialup_option ; ]
    [ delegation-only yes_or_no ; ]
    [ file string ; ]
    [ masterfile-format (text|raw|map) ; ]
    [ forward (only|first) ; ]
    [ forwarders { [ ip_addr [port ip_port] [dscp ip_dscp] ; ... ] }; ]
    [ masters [port ip_port] [dscp ip_dscp] { ( masters_list | ip_addr
        [port ip_port]
        [dscp ip_dscp]
        [key key] ) ; [...] }; ]
    [ max-transfer-idle-in number ; ]
    [ max-transfer-time-in number ; ]
    [ pubkey number number number string ; ]
    [ transfer-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
    [ transfer-source-v6 (ip6_addr | *)
        [port ip_port] [dscp ip_dscp] ; ]
    [ alt-transfer-source (ip4_addr | *) [port ip_port] [dscp ip_dscp] ; ]
    [ alt-transfer-source-v6 (ip6_addr | *)
        [port ip_port] [dscp ip_dscp] ; ]
    [ use-alt-transfer-source yes_or_no; ]
    [ zone-statistics yes_or_no ; ]
    [ database string ; ]
    [ min-refresh-time number ; ]
    [ max-refresh-time number ; ]
    [ min-retry-time number ; ]
    [ max-retry-time number ; ]
    [ multi-master yes_or_no ; ]
};

zone zone_name [class] {
    type static-stub;
    [ allow-query { address_match_list }; ]
    [ server-addresses { [ ip_addr ; ... ] }; ]
    [ server-names { [ namelist ] }; ]
    [ zone-statistics yes_or_no ; ]
};

zone zone_name [class] {

```



```

type forward;
[ forward (only|first) ; ]
[ forwarders { [ ip_addr [port ip_port] [dscp ip_dscp] ; ... ] }; ]
[ delegation-only yes_or_no ; ]
};

zone "." [class] {
    type redirect;
    file string ;
    [ masterfile-format (text|raw|map) ; ]
    [ allow-query { address_match_list }; ]
    [ max-zone-ttl number ; ]
};

zone zone_name [class] {
    type delegation-only;
};

zone zone_name [class] {
    [ in-view string ; ]
};

```

6.2.28 zone语句定义和用法

6.2.28.1 区类型

master
slave

服务器拥有区数据的主拷贝，并能够提供关于这个区的权威回答。一个辅区是一个主区的一份复制。**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入这个文件。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。（如果你将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	<p>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。</p> <p>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。在BIND 4/8中, 对父区的区传送会包含这个区中存根孩子的NS记录。这意味着, 在某些情况下, 用户可以不用配置孩子的存根, 而仅在主服务器中为父区配置。BIND 9从来不以这样的方式混淆不同区中的区数据。所以, 如果一个BIND 9配置成父区的主服务器, 并配置了子区的存根区, 那么所以父区的辅服务器也都需要配置同样的子区的存根区。</p> <p>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RFC1918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。</p>
static-stub	<p>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。</p> <p>区数据使用server-addresses和server-names区选项来配置。</p> <p>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 将区数据库转储为可读的格式。</p> <p>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, 没有周期性的刷新, 收到的通知消息将会被拒绝, 并带有一个NOTAUTH的响应码 (rcode)。</p> <p>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。</p>
forward	<p>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。这样如果你想使用这种类型的区来改变全局forward选项 (即, “forward first”, 然后“forward only”, 或者相反, 但是想要使用同样的服务器作为全局设置), 你需要重新指定全局转发者。</p>
hint	<p>使用一个“hint zone”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。</p>

redirect	<p>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。</p> <p>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。</p> <p>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“.”的典型(type)重定向区，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”。</p> <p>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“*。”。要重定向所有商业西班牙名字（在.COM.ES下面），需要使用通配符条目“*.COM.ES.”。</p> <p>注意重定向区支持所有可能的类型；它不限于A和AAAA记录。</p> <p>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rndc reload zonename来重载一个重定向区。然而，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。</p>
delegation-only	<p>这个用于强制基础区（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。这不应该应用在叶子区。</p> <p>delegation-only对从转发服务器来的回答没有效果。</p> <p>参见root-delegation-only中的说明。</p>

6.2.28.2 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。

6.2.28.3 区选项

allow-notify 参见第6.2.16.4节中对**allow-notify**的描述。

allow-query 参见第6.2.16.4节中对**allow-query**的描述。

allow-query-on 参见第6.2.16.4节中对**allow-query-on**的描述。

allow-transfer 参见第6.2.16.4节中对**allow-transfer**的描述。

allow-update 参见第6.2.16.4节中对**allow-update**的描述。

update-policy 指定一个“简单安全更新”策略。参见第6.2.28.4节。

allow-update-forwarding 参见第6.2.16.4节中对**allow-update-forwarding**的描述。

also-notify 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列出的区名字服务器（主服务器除外）和**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，让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**master**区缺省值是**fail**。对**slave**区缺省值是**warn**。对于**hint**区没有实现。

check-mx 参见第6.2.16.1节中对**check-mx**的描述。

check-spf 参见第6.2.16.1节中对**check-spf**的描述。

check-wildcard 参见第6.2.16.1节中对**check-wildcard**的描述。

check-integrity 参见第6.2.16.1节中对**check-integrity**的描述。

check-sibling 参见第6.2.16.1节中对**check-sibling**的描述。

zero-no-soa-ttl 参见第6.2.16.1节中对**zero-no-soa-ttl**的描述。

update-check-ksk 参见第6.2.16.1节中对**update-check-ksk**的描述。

dnssec-update-mode 参见第6.2.16节中对**dnssec-update-mode**的描述。

dnssec-dnskey-kskonly 参见第6.2.16.1节中对**dnssec-dnskey-kskonly**的描述。

try-tcp-refresh 参见第6.2.16.1节中对**try-tcp-refresh**的描述。

database 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解释。

缺省是“**rbt**”，BIND 9的原生内存红黑树数据库。这个数据库不需要参数。

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 参见第6.2.16.1节中对**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。

forward 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但没有回答之后导致查找失败，而**first**值将会允许进行一个普通查找的试探。

forwarders 用于重载全局转发者列表。如果没有指定区类型为**forward**，对此区就没有转发，也不使用全局选项。

ixfr-base 用于BIND 8中，为动态更新和IXFR指定事务日志（journal）文件的名字。BIND 9忽略这个选项，它通过在区文件的名字后面增加“.jnl”来构造日志文件名。

ixfr-tmp-file BIND 8中一个未列入文档的选项。在BIND 9中被忽略。

journal 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**master**和**slave**区中。

max-journal-size 参见第6.2.16.10节中对**max-journal-size**的描述。

max-transfer-time-in 参见第6.2.16.7节中对**max-transfer-time-in**的描述。

max-transfer-idle-in 参见第6.2.16.7节中对**max-transfer-idle-in**的描述。

max-transfer-time-out 参见第6.2.16.7节中对**max-transfer-time-out**的描述。

max-transfer-idle-out 参见第6.2.16.7节中对**max-transfer-idle-out**的描述。

notify 参见第6.2.16.1节中对**notify**的描述。

notify-delay 参见第6.2.16.15节中对**notify-delay**的描述。

notify-to-soa 参见第6.2.16.1节中对**notify-to-soa**的描述。

pubkey 在BIND 8中，这个选项用于指定一个公共区密钥，它是为了在从磁盘中装载DNSSEC签名的区时校验区的签名的。BIND 9在装载时不校验签名，并忽略此选项。

zone-statistics 如果为**yes**，服务器将会保持这个区的统计信息，这些信息可以被导出到服务器选项**statistics-file**所指定的文件中。

server-addresses 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。为这个选项配置一个非空列表，将会在内部配置顶点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，如果服务器受到一个带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 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，如果服务器受到一个带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（一个或多个）这些地址。

sig-validity-interval 参见第6.2.16.15节中对sig-validity-interval的描述。

sig-signing-nodes 参见第6.2.16.15节中对sig-signing-nodes的描述。

sig-signing-signatures 参见第6.2.16.15节中对sig-signing-signatures的描述。

sig-signing-type 参见第6.2.16.15节中对sig-signing-type的描述。

transfer-source 参见第6.2.16.7节中对transfer-source的描述。

transfer-source-v6 参见第6.2.16.7节中对transfer-source-v6的描述。

alt-transfer-source 参见第6.2.16.7节中对alt-transfer-source的描述。

alt-transfer-source-v6 参见第6.2.16.7节中对alt-transfer-source-v6的描述。

use-alt-transfer-source 参见第6.2.16.7节中对use-alt-transfer-source的描述。

notify-source 参见第6.2.16.7节中对notify-source的描述。

notify-source-v6 参见第6.2.16.7节中对notify-source-v6的描述。

min-refresh-time, max-refresh-time, min-retry-time, max-retry-time 参见第6.2.16.15节中的描述。

ixfr-from-differences 参见第6.2.16.1节中对ixfr-from-differences的描述。（注意ixfr-from-differences master和slave选择不能用于区这一级。）

key-directory 参见第6.2.16节中对key-directory的描述。

auto-dnssec 被配置为动态DNS的区也使用这个选项来允许多个级别的自动DNSSEC密钥管理。有下列三种可能的设置：

auto-dnssec allow;允许密钥被更新，区被全部重新签名，无论何时用户发出命令**rndc sign zonename**。

auto-dnssec maintain;包含以上命令，还能够工具密钥的计时元数据自动按时间表调整区的DNSSEC密钥（参见**dnssec-keygen(8)**和**dnssec-settime(8)**）。命令**rndc sign zonename**让named从密钥仓库装载密钥并使用所有激活的密钥签名区。**rndc loadkeys zonename**让named从密钥仓库装载密钥并调度将来发生的的密钥维护事件，但是它不立即签名整个区。注意：一旦一个区的密钥被首次装载后，将定期搜索仓库以查找变化，无论是否使用了**rndc loadkeys**。重新检查的间隔被**dnssec-loadkeys-interval**所定义。）

缺省设置是**auto-dnssec off**。

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。

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 参见第6.2.16.1节中对**multi-master**的描述。

masterfile-format 参见第6.2.16.15节中对**masterfile-format**的描述。

max-zone-ttl 参见第6.2.16节中对**max-zone-ttl**的描述。

dnssec-secure-to-insecure 参见第6.2.16.1节中对**dnssec-secure-to-insecure**的描述。

6.2.28.4 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，分别是配置**allow-update**和**update-policy**选项。

allow-update子句与先前的BIND是同样的工作方式。它授权客户端更新一个区中任何名字的任何记录的权限。

update-policy子句允许对所授权的更新作更细粒度的控制。指定一个规则集，其中的每个规则是授权或禁止一个或多个名字被一个或多个身份所更新。如果动态更新请求消息是签名的（即包含TSIG或者SIG(0)记录），可以决定签名者的身份。

在**update-policy**区选项中所指定的规则，仅仅对主区有意义。当出现**update-policy**语句时，再出现**allow-update**语句就会报配置错误。**update-policy**语句仅仅检查一个消息的签名者；而不管源地地址。

有一个预定义的**update-policy**规则，可以使用命令**update-policy local**打开。对一个区打开这个规则会导致**named**生成一个TSIG会话密钥并将其放到一个文件中，并且允许用这个密钥更新这个区。

（缺省时，文件是/var/run/named/session.key，密钥名是“local-ddns”，密钥算法是HMAC-SHA256，但这些值是分别配置在**session-keyfile**，**session-keyname**和**session-keyalg**选项中）。

一个运行在本地系统上并带有合适授权的客户端可以读那个文件，并使用密钥来签名更新请求。区的签名策略将会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”，这个策略等效为：

```
update-policy { grant local-ddns zonesub any; };
```

nsupdate -l命令向本机发出更新请求，并使用会话密钥对其签名。

其它的规则看起来像这个：

```
( grant | deny ) identity nametype [ name ] [ types ]
```

每条规则授予或禁止权限。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。一条规则匹配是指签名者与**identity**字段匹配，名字按照**nametype**字段与**name**字段匹配，并且类型与**types**字段所指定的类型匹配。

对tcp-self或6to4-self不要求有签名者，然而，标准的反向映射/前缀转换必须匹配标识字段。

标识符字段指定一个名字或一个通配名字。通常情况下，这是用来签名更新请求的TSIG或SIG(0)密钥的名字。当使用TKEY交换建立了一个共享密钥之后，共享密钥的标识符就与用于认证TKEY交换的密钥的标识符相同。TKEY也是GSS-TSIG所使用的协商方法，后者建立一个标识，即客户端的Kerberos标识，如"user@host.domain"。当identity字段指定了一个通配名字，它服从DNS通配符扩展，这样规则会应用到多个标识符。identity字段必须包含一个完整域名。

对名字类型krb5-self, ms-self, krb5-subdomain, 和ms-subdomain, identity字段指定机器所属的Windows或Kerberos域。


nametype字段有13个值：name, subdomain, wildcard, self, selfsub, selfwild, krb5-self, ms-self, krb5-subdomain, ms-subdomain, tcp-self, 6to4-self, zonesub和external。

name	精确匹配语义。当被更新的名字与name字段内容一致时，这个规则匹配。
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。这就排除了一次输入区名的需要，并且能够在多个区中使用标准的update-policy语句而不需要修改。
wildcard	在使用这条规则时，name字段会被忽略。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	当被更新的名字匹配identity字段的内容时，这条规则匹配。name字段被忽略，但是应该与identity字段一致。在允许使用每个名字一个密钥时，self名字类型是最有用的，这时密钥与被更新的名字有同样的名字。这种情况下，identity应被指定为*（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	这条规则接受Windows机器标识（machine\$@REALM）作为域（REALM）中的机器，并将其转换为machine.realm，允许这台机器更新machine.realm。所匹配的域（REALM）在identity字段中指定。
ms-subdomain	这条规则接受Windows机器标识（machine\$@REALM）作为域（REALM）中的机器，并将其转换为machine.realm，允许这台机器更新machine.realm的子域。所匹配的域（REALM）在identity字段中指定。
krb5-self	这条规则接受Kerberos机器标识（host/machine@REALM）作为域（REALM）中的机器，并将其转换为machine.realm，允许这台机器更新machine.realm。所匹配的域（REALM）在identity字段中指定。
krb5-subdomain	这条规则接受Kerberos机器标识（host/machine@REALM）作为域（REALM）中的机器，并将其转换为machine.realm，允许这台机器更新machine.realm的子域。所匹配的域（REALM）在identity字段中指定。
tcp-self	允许通过TCP发送更新，并且对这些更新，从发起更新的IP地址到IN-ADDR.ARPA和IP6.ARPA名字空间进行标准的映射，并与被更新的名字进行匹配。

注意



理论上有可能欺骗这些TCP会话。

6to4-self	<p>允许通过任何从6to4网络或从对应的IPv4地址的TCP连接更新6to4前缀。这是计划允许NS或DNAME资源记录集被添加到反向树中。</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p style="text-align: center;">注意</p> <div style="display: flex; align-items: center;">  <p>理论上有可能欺骗这些TCP会话。</p> </div> </div>
external	<p>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。</p> <p>和服务进程通信的方法在<i>identity</i>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。）</p> <p>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：</p> <p style="margin-left: 20px;">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。</p>

在所有的情况中，*name*字段必须指定为一个完整域名。

如果没有显式指定*types*，这条规则就匹配除RRSIG，NS，SOA，NSEC和NSEC3之外的所有类型。*types*可以由名字指定，包括“ANY”（ANY匹配除NSEC和NSEC3之外的所有类型，后者不能被更新）。注意当遇到一个删除与一个名字相关的所有记录的企图时，需要针对每个现存记录类型对规则进行检查。

6.2.28.5 多视图

当使用了多个视图，一个区可能是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;
    };
};
```

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。）

一个**in-view**区不能用作一个响应策略区。

6.3 区文件

6.3.1 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。

6.3.1.1 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，也不会名字服务器，解析器或DNS的其它部份中保存。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。参见第6.2.16.13节和第6.2.16.14节。

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（有时包括类）相关的。

以下是有效资源记录的类型：

A	一个主机地址。在IN类中，这是一个32位的IP地址。在RFC 1035中描述。
AAAA	IPv6地址。在RFC 1886中描述。
A6	IPv6地址。它可以是一部份地址（一个后缀）加上一个可以找到其余部份地址（前缀）的间接指向的名字。试验性的。在RFC 2874中描述。
AFSDB	AFS数据库服务器的位置。试验性的。在RFC 1183中描述。
APL	地址后缀列表。试验性的。在RFC 3123中描述。
CERT	存放一个数字签名。在RFC 2538中描述。
CNAME	指定一个别名的正规名。在RFC 1035中描述。
DHCID	用于识别哪一个DHCP客户端被关联到这个名字。在RFC 4701中描述。
DNAME	使用另一个名字替代所指定的名字后进行查找，有效地将域名空间的一个完整子树作为别名，与之相对的使CNAME RR将单个记录作为别名。在RFC 2672中描述。
DNSKEY	存储与一个签名DNS区相关的公钥。在RFC 4034中描述。
DS	存储与一个签名DNS区相关的公钥的hash。在RFC 4034中描述。

GPOS	指定全球位置。为LOC所替代。
HINFO	指定主机所用的CPU和OS。在RFC 1035中描述。
IPSECKEY	提供一个存储DNS中的IPsec keying材料的方法。在RFC 4025中描述。
ISDN	ISDN地址的表示。试验性的。在RFC 1183中描述。
KEY	存放一个与DNS名字相关的公钥。用于最初的DNSSEC； 现被DNSSECbis中的DNSKEY所替代，但仍然用于SIG(0)。 在RFC 2535 和2931中描述。
KX	指定一个DNS名字的密钥交换者。在RFC 2230中描述。
LOC	存放GPS信息。在RFC 1876中描述。试验性的。
MX	为一个域指定邮件交换者。由邮件交换者的主机名后跟一个16 位的优先数（数值越小越优先）。在RFC 974、RFC 1035中描述。
NAPTR	名字权威指针。在RFC 2915中描述。
NSAP	一个网络服务访问点。在RFC 1706中描述。
NS	域的权威名字服务器。在RFC 1035中描述。
NSEC	用于DNSSECbis中，指明在某个名字内部并带有属主名的资源记录不在一个区内，也指明某个存在的名字所提供的资源记录类型。在RFC 4034中描述。
NSEC3	用于DNSSECbis，以安全地指示在一个区中某一名字间隔内不存在某个指定名字的资源记录，以及指示一个存在的名字具有那种资源记录类型。NSEC3与NSEC的区别是它阻止对区的枚举，但是比NSEC更加消耗服务器和客户端的计算资源。在RFC 5155中描述。
NSEC3PARAM	用于DNSSECbis，告诉权威服务器哪个NSEC3链是可用的。在RFC 5155中描述。
NXT	用于DNSSEC中，安全地指明在某个名字之间的给定属主名的资源记录不存在于一个区中，并指明对一个存在的名字具有哪种类型的资源记录。在原始的DNSSEC中使用； 在DNSSECbis中被NSEC取代。在RFC 2535中描述。
PTR	指向域名空间另一部份的指针。在RFC 1035中描述。
PX	在RFC 822和X.400地址之间提供映射。在RFC 2163中描述。
RP	关于域的负责人的信息。试验性的。在RFC 1183中描述。
RRSIG	包含DNSSECbis签名数据。在RFC 4034中描述。
RT	绑定在没有自己的直接的广域网地址的主机上的路由出口。试验性的。在RFC 1183中描述。
SIG	包含DNSSEC签名数据。在原始的DNSSEC中使用；已被DNSSECbis 中的RRSIG所替代，但是仍然用作SIG(0)。 在RFC 2535和2931中描述。
SOA	标识一个授权区的头部。在RFC 1035中描述。
SPF	包含一个给定电子邮件域名的发送者策略框架信息。在RFC 4408中描述。
SRV	关于众所周知的网络服务的信息（替代WKS）。在RFC 2782中描述。
SSHFP	提供发布一个安全shell密钥指纹的安全方式。在RFC 4255中描述。
TXT	文本记录。在RFC 1035中描述。
WKS	关于一个域所支持的众所周知的网络服务的信息，如SMTP。已成历史。
X25	X.25网络地址的表示。试验性的。在RFC 1183中描述。

以下是当前DNS中有有效的资源记录的类：

IN	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字段的含义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由数据源的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”。

6.3.1.2 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, 类型和类的缩略语不能连着写, 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。

类似的我们可以看到:

XX.LCS.MIT.EDU.	IN A	10.0.0.44
	CH A	MIT.EDU. 2420

这个例子显示了XX.LCS.MIT.EDU的两个地址, 分别在不同的类中。

6.3.2 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数据对，即一个与一个相关数据匹配的域名，和被存储的一些附加类型信息，后者是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记录就是错误的，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	

邮件投递将先尝试mail.example.com和mail2.example.com（以任何顺序），然后如果都没有成功，将会尝试投递到mail.backup.org。

6.3.3 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。它控制从你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，虽然单位都可以被显式指定，例如，1h30m。

6.3.4 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.

注意



例子中的**\$ORIGIN**行用于提供相当于例子的上下文，仅仅用在实际中不需要出现的地方。它们用于这里，仅仅是指明例子是相对于所列出的起点。

6.3.5 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**。

6.3.5.1 The @ (at-sign)

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)。

6.3.5.2 \$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，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.
```

6.3.5.3 \$INCLUDE指令

语法: **\$INCLUDE** *filename* [*origin*] [*comment*]

读入并处理文件*filename*，就象它在这一点包含此文件进来。如果设定了**origin**，文件就使用**\$ORIGIN**所设定的值被处理，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。

注意



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。

6.3.5.4 \$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。

6.3.6 BIND主文件扩展: \$GENERATE指令

语法: **\$GENERATE** *range lhs* [*tll*] [*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: 无类IN-ADDR.ARPA授权。

```
$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.
```

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**都必须是在于0 和(2³¹)-1之间的正整数。 **start**必须小于等于**stop**。为正值。

lhs	描述所建立的资源记录的属主名。任何在 lhs 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）引入，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 lhs 不是绝对名字，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，作为指示在输出中的一个字面的\$。
ttl	指定所生成的记录的生存期。如果在这里未指定，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	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	所有有效的类型。
rhs	rhs ，可选，被引号包含的字符串。

\$GENERATE指令是一个BIND 的扩展，并不是标准区文件格式的一部份。

BIND 8不支持可选的TTL和CLASS字段。

6.3.7 附加文件格式

除了标准的文件格式，BIND 9支持读或者列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完成区传送之后转储区或者应用上次的更新时自动生成的（如果使用masterfile-format选项指定了这种格式）。

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格式是及其体系相关的。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。

6.4 BIND9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BIND 8中可用的并在BIND 9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（进入请求）	对每个OPCODE的进入的DNS请求数目。
-------------------------	-----------------------

Incoming Queries (进入查询)	对每个资源记录类型的进入的查询数目。
Outgoing Queries (出去查询)	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)	存放在缓存数据库中的每种资源记录类型的资源记录集和不存在的名字的数目。如果为某个资源记录类型打印了惊叹号(!)，它表示特定类型的资源记录是不存在的(也即“NXRRSET”)。如果出现一个散列符号(#)，资源记录集被标记为垃圾回收。每个视图各自维护。
Socket I/O Statistics (套接字I/O统计)	网络相关事件的统计计数器。

当**zone-statistics**被设为**yes**时，对服务器是其权威的每个区，都收集并显示名字服务器统计的一个子集。这些统计计数器与其区和视图名一起显示。在某些情况，缺省视图的视图名被省略。

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，转印到由**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。(参见第6.2.19节)。

6.4.0.1 统计文件

文本格式的统计转印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。

统计转印结束是一个带有与开始行同样数字的行；例如：

```
— Statistics Dump — (973798949)
```

6.4.1 统计计数器

下表概括了BIND所提供的统计计数器。对表的每一行，最左一列是计数器的缩写符号名。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。最右列给出了计数器的描述，这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）。附加的说明也在这列提供。当这两列之间的中间列存在时，它给出BIND 8统计中对应的计数器名，如果合适的时候。

6.4.1.1 名字服务器统计计数器

<i>Symbol</i>	<i>BIND8 Symbol</i>	<i>Description</i>
Requestv4	RQ	收到的IPv4请求。注意：这也会记录不是查询的请求。
Requestv6	RQ	收到的IPv6请求。注意：这也会记录不是查询的请求。
ReqEdns0		收到的带EDNS(0)的请求。
ReqBadEDNSVer		收到的带有不支持的EDNS版本的请求。

ReqTSIG		收到的带有TSIG的请求。
ReqSIG0		收到的带有SIG(0)的请求。
ReqBadSIG		收到的带有不正确(TSIG或SIG(0))签名的请求。
ReqTCP	RTCP	收到的TCP请求。
AuthQryRej	RUQ	拒绝掉的权威（非递归）请求。
RecQryRej	RURQ	拒绝掉的递归查询。
XfrRej	RUXFR	拒绝掉的区传送请求。
UpdateRej	RUUpd	拒绝掉的动态更新请求。
Response	SAns	已发出的响应。
RespTruncated		已发出的截断响应。
RespEDNS0		已发出的带EDNS(0)的响应。
RespTSIG		已发出的带TSIG的响应。
RespSIG0		已发出的带SIG(0)的响应。
QrySuccess		导致一个成功回答的查询。这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		导致一个权威回答的查询。
QryNoauthAns	SNaAns	导致一个非权威回答的查询。
QryReferral		导致一个引用回答的查询。这相当于先前BIND 9版本中的 referral 计数器。
QryNxrrset		导致在NOERROR响应中没有数据的查询。这相当于先前BIND 9版本中的 nxrrset 计数器。
QrySERVFAIL	SFail	导致SERVFAIL的查询。
QryFORMERR	SFErr	导致FORMERR的查询。
QryNXDOMAIN	SNXD	导致NXDOMAIN的查询。这相当于先前BIND 9版本中的 nxdomain 计数器。
QryRecursion	RFwdQ	导致服务器执行递归解析以获取最终回答的查询。这相当于先前BIND 9版本中的 recursion 计数器。
QryDuplicate	RDupQ	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。这相当于先前BIND 9版本中的 duplicate 计数器。
QryDropped		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。这是由于 clients-per-query 和 max-clients-per-query 选项（参见关于 clients-per-query 的描述）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		其它查询失败。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。
XfrReqDone		已完成的所请求的区传送。
UpdateReqFwd		已转发的更新请求。
UpdateRespFwd		已转发的更新响应。
UpdateFwdFail		已失败的动态更新转发。
UpdateDone		已完成的动态更新。
UpdateFail		已失败的动态更新。
UpdateBadPrereq		由于先决条件失败而被拒绝的动态更新。
RateDropped		被比率限制丢弃的响应。
RateSlipped		被比率限制截断的响应。
RPZRewrites		响应策略区重写。

6.4.1.2 区维护统计计数器

<i>Symbol</i>	<i>Description</i>
NotifyOutv4	已发送的IPv4通知。
NotifyOutv6	已发送的IPv6通知。
NotifyInv4	收到的IPv4通知。
NotifyInv6	收到的IPv6通知。
NotifyRej	拒绝掉的进入通知。
SOAOutv4	发送出的IPv4 SOA查询。
SOAOutv6	发送出的IPv6 SOA查询。
AXFRReqv4	已请求的IPv4 AXFR。
AXFRReqv6	已请求的IPv6 AXFR。
IXFRReqv4	已请求的IPv4 IXFR。
IXFRReqv6	已请求的IPv6 IXFR。
XfrSuccess	已成功的区传送请求。
XfrFail	已失败的区传送请求。

6.4.1.3 解析器统计计数器

<i>Symbol</i>	<i>BIND8 Symbol</i>	<i>Description</i>
Queryv4	SFwdQ	已发送的IPv4查询。
Queryv6	SFwdQ	已发送的IPv6查询。
Responsev4	RR	已收到的IPv4响应。
Responsev6	RR	已收到的IPv6响应。
NXDOMAIN	RNXD	已收到的NXDOMAIN。
SERVFAIL	RFail	已收到的SERVFAIL。
FORMERR	RFErr	已收到的FORMERR。
OtherError	RErr	已收到的其它错误。
EDNS0Fail		EDNS(0)查询失败。
Mismatch	RDupR	已收到的不匹配响应。DNS ID，响应的源地址和（或）响应的源端口与预计的不匹配。（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		已收到的截断响应。
Lame	RLame	已收到的不完整授权。
Retry	SDupQ	执行过的查询重试。
QueryAbort		由于配额控制而被终止的查询。
QuerySockFail		打开查询套接字时的失败。这类失败的一个通常原因是由于在文件描述符上的限制而导致打开一个新套接字失败。
QueryTimeout		查询超时。
GlueFetchv4	SSysQ	发起的取IPv4 NS的地址操作。
GlueFetchv6	SSysQ	发起的取IPv6 NS的地址操作。
GlueFetchv4Fail		失败的取IPv4 NS的地址操作。
GlueFetchv6Fail		失败的取IPv6 NS的地址操作。
ValAttempt		试探过的DNSSEC验证。
ValOk		成功的DNSSEC验证。
ValNegOk		在否定信息上成功的DNSSEC验证。
ValFail		失败的DNSSEC验证。

QryRTTnn	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。
-----------------	--

6.4.1.4 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。

<i>Symbol</i>	<i>Description</i>
<TYPE> Open	套接字成功打开。这个计数器不适用于 FDwatch 类型。
<TYPE> OpenFail	套接字打开失败。这个计数器不适用于 FDwatch 类型。
<TYPE> Close	套接字关闭。
<TYPE> BindFail	绑定套接字失败。
<TYPE> ConnFail	连接套接字失败。
<TYPE> Conn	成功建立连接。
<TYPE> AcceptFail	接受进入连接请求失败。这个计数器不适用于 UDP 和 FDwatch 类型。
<TYPE> Accept	成功接受进入连接请求。这个计数器不适用于 UDP 和 FDwatch 类型。
<TYPE> SendErr	套接字发送操作中的错误。这个计数器对应于 BIND 8 的 SErr 计数器。
<TYPE> RecvErr	套接字接收操作中的错误。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。

6.4.1.5 对BIND 8计数器的兼容性

如以上的表格所示，大多数在**BIND 8**中可用的统计计数器在**BIND 9**中也被支持。这里有一些关于这些表格中没有的其它计数器的注意事项。

RFwdR,SFwdR 这些计数器不被支持，因为**BIND 9**没有采用**BIND 8**中所采用的*forwarding*概念。

RAXFR 这个计数器在进入查询部份中是可访问的。

RIQ 这个计数器在进入请求部份中是可访问的。

ROpts 这个计数器已不再支持，因为**BIND 9**不关心第一个位置的IP选项。

Chapter 7

BIND 9安全考虑

7.1 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**，**allow-recursion-on**，**blackhole**，**allow-transfer** 等语句中。

使用ACL允许你对谁能够访问你的名字服务器有一个良好的控制，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。

这是一个如何正确应用ACL的例子：

```
// Set up an ACL named "bogusnets" that will block
// 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 {
    ...
    ...
    allow-query { our-nets; };
    allow-recursion { our-nets; };
    ...
    blackhole { bogusnets; };
    ...
};

zone "example.com" {
    type master;
    file "m/example.com";
    allow-query { any; };
};
```

这将允许外部对服务器进行递归查询，除非在之前递归被禁止。

7.2 Chroot和Setuid

在UNIX服务器上，可以通过为**named** 设定“-t”选项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
```

7.2.1 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，你需要设置一个环境，使其包含BIND运行所需的所有东西。从BIND的视角，**/var/named**是文件系统的根。你需要通过调整象**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，你不需要静态地编译**named**，也不需要新的根下面安装共享库。然而，依赖于你的操作系统，你可能需要设置诸如这样的一些东西：**/dev/zero**，**/dev/random**，**/dev/log**以及**/etc/localtime**。

7.2.2 使用setuid函数

在运行**named**服务之前，在你想让BIND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）。

注意



注意如果**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限制端口上。

7.3 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琐碎的攻击，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，包含象公共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。

Chapter 8

排除故障

8.1 通常问题

8.1.1 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，后者用于判断哪里出现了错误以及如何来解决问题。

8.2 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。

8.3 从哪里获得帮助？

互联网系统集团（Internet Systems Consortium, ISC）对BIND和DHCP服务器提供广泛的支持和协议服务。可以提供四级费用的支持，每级包括对所有ISC的程序的支持，产品和培训上的大幅折扣，在故障解决上的认识优先和非盈利特性请求。另外，ISC提供标准的支持协议包，它包括的服务从故障解决通告到远程支持。它也包括对BIND和DHCP的培训。

要讨论对支持的安排，请联系info@isc.org <<mailto:info@isc.org>>或访问ISC的主页<<http://www.isc.org/services/support/>> 以了解更多信息。

Appendix A

发行注记

A.1 BIND版本9.10.2的发行注记

A.1.1 介绍

本文档总结了在相应的主版本分支上自从上一个BIND产品版本以来的变化。

A.1.2 Download

最新版的BIND 9软件总是可以在<<http://www.isc.org/downloads/>>找到。在那里，你会发现每个版本的附加信息，源码和微软Windows操作系统上的预编译版本。

A.1.3 安全修补

- 在配置了使用受管理的信任锚（即，通过**managed-keys**显式配置密钥，或通过**dnssec-validation auto**；或者**dnssec-lookaside auto**；隐式配置）来执行DNSSEC验证的服务器上，撤销一个信任锚并发送一个新的未受信任的替代者可能导致**named**崩溃并伴有一个断言失败。This could occur in the event of a botched key rollover, or potentially as a result of a deliberate attack if the attacker was in position to monitor the victim's DNS traffic.

这个缺陷由Jan-Piet Mens发现，并在CVE-2015-1349中公开。[RT #38344]

- 一个授权处理中的缺陷可能被利用并将**named**置于一个无限循环中，在其中每个对一个名字服务器的查找都触发附加的对更多名字服务器的查找。这已被处理，通过限制**named**所允许的递归层级数目（缺省是7），和在中断一个递归请求之前它会发出的请求个数（缺省是50）。

递归深度限制由max-recursion-depth选项配置，而请求限制由max-recursion-queries选项配置。

这个缺陷由ANSSI的Florian Maury发现，并在CVE-2014-8500中公开。[RT #37580]

- 在BIND的GeoIP代码中，有两个分离的问题被鉴定可能导致一个断言失败。其中一个由同时使用IPv4和IPv6地址族所触发，另一个由在named.conf中引用了一个GeoIP数据库却未安装这个数据库所触发。这两个都被CVE-2014-8680所覆盖。[RT #37672] [RT #37679]

A less serious security flaw was also found in GeoIP: changes to the **geoip-directory** option in named.conf were ignored when running **rndc reconfig**. In theory, this could allow **named** to allow access to unintended clients.

A.1.4 新特性

- 无

A.1.5 有变化的特性

- ACLs containing **geoip asnum** elements were not correctly matched unless the full organization name was specified in the ACL (as in **geoip asnum "AS1234 Example, Inc."**). They can now match against the AS number alone (as in **geoip asnum "AS1234"**).
- 在使用原生PKCS#11加密时（即**configure --enable-native-pkcs11**），可以使用至多256个字符的HSM PIN。
- NXDOMAIN responses to queries of type DS are now cached separately from those for other types. This helps when using "grafted" zones of type forward, for which the parent zone does not contain a delegation, such as local top-level domains. Previously a query of type DS for such a zone could cause the zone apex to be cached as NXDOMAIN, blocking all subsequent queries. (Note: This change is only helpful when DNSSEC validation is not enabled. "Grafted" zones without a delegation in the parent are not a recommended configuration.)
- NOTIFY messages that are sent because a zone has been updated are now given priority above NOTIFY messages that were scheduled when the server started up. This should mitigate delays in zone propagation when servers are restarted frequently.
- 运行**rndc addzone**时（例如，当一个区文件不能被装载时）的错误报告被澄清，以使诊断问题更容易。
- 增加对OPENPGPKEY类型的支持。
- 当遇到一个权威名字服务器，其名字为一个别名指向另一个名字时，解析器将这个当成一个错误并调到下一个服务器。之前这个静默地发生；现在将会在新建的"cname"日志类别中记录错误。
- If named is not configured to validate the answer then allow fallback to plain DNS on timeout even when we know the server supports EDNS. This will allow the server to potentially resolve signed queries when TCP is being blocked.

A.1.6 Bug Fixes

- **dig**, **host** and **nslookup** aborted when encountering a name which, after appending search list elements, exceeded 255 bytes. Such names are now skipped, but processing of other names will continue. [RT #36892]
- The error message generated when **named-checkzone** or **named-checkconf -z** encounters a \$TTL directive without a value has been clarified. [RT #37138]
- Semicolon characters (;) included in TXT records were incorrectly escaped with a backslash when the record was displayed as text. This is actually only necessary when there are no quotation marks. [RT #37159]
- When files opened for writing by **named**, such as zone journal files, were referenced more than once in **named.conf**, it could lead to file corruption as multiple threads wrote to the same file. This is now detected when loading **named.conf** and reported as an error. [RT #37172]
- 由于私钥文件中的一个差异，**dnssec-keygen -S**无法对某些算法类型（包括ECDSA和GOST）生成后续的密钥。这个已被纠正。[RT #37183]
- UPDATE messages that arrived too soon after an **rndc thaw** could be lost. [RT #37233]
- Forwarding of UPDATE messages did not work when they were signed with SIG(0); they resulted in a BADSIG response code. [RT #37216]
- When checking for updates to trust anchors listed in **managed-keys**, **named** now revalidates keys based on the current set of active trust anchors, without relying on any cached record of previous validation. [RT #37506]
- Large-system tuning (**configure --with-tuning=large**) caused problems on some platforms by setting a socket receive buffer size that was too large. This is now detected and corrected at run time. [RT #37187]

- When NXDOMAIN redirection is in use, queries for a name that is present in the redirection zone but a type that is not present will now return NOERROR instead of NXDOMAIN.
- When a zone contained a delegation to an IPv6 name server but not an IPv4 name server, it was possible for a memory reference to be left un-freed. This caused an assertion failure on server shutdown, but was otherwise harmless. [RT #37796]
- Due to an inadvertent removal of code in the previous release, when **named** encountered an authoritative name server which dropped all EDNS queries, it did not always try plain DNS. This has been corrected. [RT #37965]
- A regression caused nsupdate to use the default recursive servers rather than the SOA MNAME server when sending the UPDATE.
- Adjusted max-recursion-queries to accommodate the smaller initial packet sizes used in BIND 9.10 and higher when contacting authoritative servers for the first time.
- Built-in "empty" zones did not correctly inherit the "allow-transfer" ACL from the options or view. [RT #38310]
- 两个可能导致**named**进程增长到非常大的内存泄漏被修复。[RT #38454]
- 修复一些RFC 5011信任锚管理的错误，包括一个内存泄漏和一个可能的状态信息丢失。[RT #38458]

A.1.7 End of Life

BIND 9.10的生命周期还未确定，但不会早于BIND 9.12.0发布之后半年。<<https://www.isc.org/downloads/software-support-policy/>>

A.1.8 Thank You

谢谢每一个你对我们的帮助，使得这个版本成为可能。如果你想捐助ISC以帮助我们不断提供优质的开源软件，请访问我们的赞助网页<<http://www.isc.org/donate/>>.

Appendix B

DNS和BIND的简要历史

B.1 Section

尽管域名系统的“官方的”开始是随着RFC 920的公布而发生在1984年，但这个新系统的核心是由RFC 882和883在1983年描述的。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 Cherenon, 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之后由互联网系统集团及其前身互联网软件集团进行开发和维护，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，在几个公司的赞助下，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。

Appendix C

通用DNS参考信息

C.1 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，“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。

全局路由前缀有上层提供者或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，就象在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IPv4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，通常可以翻盖缺省设置。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。

C.2 参考书目（和建议读物）

C.2.1 注释请求（RFC）

包括DNS的互联网协议族的规范文档是作为注释请求（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可以通过FTP由以下地址获得：

`ftp://www.isi.edu/in-notes/RFCxxxx.txt` <`ftp://www.isi.edu/in-notes/`>

（在这里xxxx是RFC的编号）。RFC也可以通过主页获得：

<`http://www.ietf.org/rfc/`>.

References

标准

[RFC1034] *Domain Names — Concepts and Facilities*, P.V. Mockapetris, November 1987.

[RFC1035] *Domain Names — Implementation and Specification*, P. V. Mockapetris, November 1987.

[RFC974] *Mail Routing and the Domain System*, C. Partridge, January 1986.

建议标准

- [RFC1995] *Incremental Zone Transfer in DNS*, M. Ohta, August 1996.
- [RFC1996] *A Mechanism for Prompt Notification of Zone Changes*, P. Vixie, August 1996.
- [RFC2136] *Dynamic Updates in the Domain Name System*, P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, April 1997.
- [RFC2181] *Clarifications to the DNS Specification*, R., R. Bush Elz, July 1997.
- [RFC2308] *Negative Caching of DNS Queries*, M. Andrews, March 1998.
- [RFC2671] *Extension Mechanisms for DNS (EDNS0)*, P. Vixie, August 1997.
- [RFC2672] *Non-Terminal DNS Name Redirection*, M. Crawford, August 1999.
- [RFC2845] *Secret Key Transaction Authentication for DNS (TSIG)*, P. Vixie, O. Gudmundsson, D. Eastlake, 3rd, and B. Wellington, May 2000.
- [RFC2930] *Secret Key Establishment for DNS (TKEY RR)*, D. Eastlake, 3rd, September 2000.
- [RFC2931] *DNS Request and Transaction Signatures (SIG(0)s)*, D. Eastlake, 3rd, September 2000.
- [RFC3007] *Secure Domain Name System (DNS) Dynamic Update*, B. Wellington, November 2000.
- [RFC3645] *Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)*, S. Kwan, P. Garg, J. Gilroy, L. Esibov, J. Westhead, and R. Hall, October 2003.

DNS安全建议标准

- [RFC3225] *Indicating Resolver Support of DNSSEC*, D. Conrad, December 2001.
- [RFC3833] *Threat Analysis of the Domain Name System (DNS)*, D. Atkins and R. Austein, August 2004.
- [RFC4033] *DNS Security Introduction and Requirements*, R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, March 2005.
- [RFC4034] *Resource Records for the DNS Security Extensions*, R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, March 2005.
- [RFC4035] *Protocol Modifications for the DNS Security Extensions*, R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, March 2005.

其它关于DNS实现的重要RFC

- [RFC1535] *A Security Problem and Proposed Correction With Widely Deployed DNS Software.*, E. Gavron, October 1993.
- [RFC1536] *Common DNS Implementation Errors and Suggested Fixes*, A. Kumar, J. Postel, C. Neuman, P. Danzig, and S. Miller, October 1993.
- [RFC1982] *Serial Number Arithmetic*, R. Elz and R. Bush, August 1996.
- [RFC4074] *Common Misbehaviour Against DNS Queries for IPv6 Addresses*, Y. Morishita and T. Jinmei, May 2005.

资源记录类型

- [RFC1183] *New DNS RR Definitions*, C.F. Everhart, L. A. Mamakos, R. Ullmann, and P. Mockapetris, October 1990.
- [RFC1706] *DNS NSAP Resource Records*, B. Manning and R. Colella, October 1994.
- [RFC1876] *A Means for Expressing Location Information in the Domain Name System*, C. Davis, P. Vixie, T., and I. Dickinson, January 1996.
- [RFC2052] *A DNS RR for Specifying the Location of Services.*, A. Gulbrandsen and P. Vixie, October 1996.
- [RFC2163] *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping*, A. Al-locchio, January 1998.
- [RFC2168] *Resolution of Uniform Resource Identifiers using the Domain Name System*, R. Daniel and M. Mealling, June 1997.
- [RFC2230] *Key Exchange Delegation Record for the DNS*, R. Atkinson, October 1997.
- [RFC2536] *DSA KEYs and SIGs in the Domain Name System (DNS)*, D. Eastlake, 3rd, March 1999.
- [RFC2537] *RSA/MD5 KEYs and SIGs in the Domain Name System (DNS)*, D. Eastlake, 3rd, March 1999.
- [RFC2538] *Storing Certificates in the Domain Name System (DNS)*, D. Eastlake, 3rd and O. Gudmundsson, March 1999.
- [RFC2539] *Storage of Diffie-Hellman Keys in the Domain Name System (DNS)*, D. Eastlake, 3rd, March 1999.
- [RFC2540] *Detached Domain Name System (DNS) Information*, D. Eastlake, 3rd, March 1999.
- [RFC2782] *A DNS RR for specifying the location of services (DNS SRV)*, A. Gulbrandsen, P. Vixie, L. Esibov, February 2000.
- [RFC2915] *The Naming Authority Pointer (NAPTR) DNS Resource Record*, M. Mealling, R. Daniel, September 2000.
- [RFC3110] *RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)*, D. Eastlake, 3rd, May 2001.
- [RFC3123] *A DNS RR Type for Lists of Address Prefixes (APL RR)*, P. Koch, June 2001.
- [RFC3596] *DNS Extensions to support IP version 6*, S. Thomson, C. Huitema, V. Ksinant, and M. Souissi, October 2003.
- [RFC3597] *Handling of Unknown DNS Resource Record (RR) Types*, A. Gustafsson, September 2003.

DNS和互联网

- [RFC1101] *DNS Encoding of Network Names and Other Types*, P. V. Mockapetris, April 1989.
- [RFC1123] *Requirements for Internet Hosts - Application and Support*, Braden, October 1989.
- [RFC1591] *Domain Name System Structure and Delegation*, J. Postel, March 1994.
- [RFC2317] *Classless IN-ADDR.ARPA Delegation*, H. Eidnes, G. de Groot, and P. Vixie, March 1998.
- [RFC2826] *IAB Technical Comment on the Unique DNS Root*, Internet Architecture Board, May 2000.
- [RFC2929] *Domain Name System (DNS) IANA Considerations*, D. Eastlake, 3rd, E. Brunner-Williams, and B. Manning, September 2000.

DNS运行

- [RFC1033] *Domain administrators operations guide.*, M. Lottor, November 1987.
- [RFC1537] *Common DNS Data File Configuration Errors*, P. Beertema, October 1993.
- [RFC1912] *Common DNS Operational and Configuration Errors*, D. Barr, February 1996.
- [RFC2010] *Operational Criteria for Root Name Servers.*, B. Manning and P. Vixie, October 1996.
- [RFC2219] *Use of DNS Aliases for Network Services.*, M. Hamilton and R. Wright, October 1997.

国际化域名

- [RFC2825] *A Tangled Web: Issues of I18N, Domain Names, and the Other Internet protocols*, IAB and R. Daigle, May 2000.
- [RFC3490] *Internationalizing Domain Names in Applications (IDNA)*, P. Faltstrom, P. Hoffman, and A. Costello, March 2003.
- [RFC3491] *Nameprep: A Stringprep Profile for Internationalized Domain Names*, P. Hoffman and M. Blanchet, March 2003.
- [RFC3492] *Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)*, A. Costello, March 2003.

其它DNS相关的RFC

- [RFC1464] *Using the Domain Name System To Store Arbitrary String Attributes*, R. Rosenbaum, May 1993.
- [RFC1713] *Tools for DNS Debugging*, A. Romao, November 1994.
- [RFC1794] *DNS Support for Load Balancing*, T. Brisco, April 1995.
- [RFC2240] *A Legal Basis for Domain Name Allocation*, O. Vaughan, November 1997.
- [RFC2345] *Domain Names and Company Name Retrieval*, J. Klensin, T. Wolf, and G. Oglesby, May 1998.
- [RFC2352] *A Convention For Using Legal Names as Domain Names*, O. Vaughan, May 1998.
- [RFC3071] *Reflections on the DNS, RFC 1591, and Categories of Domains*, J. Klensin, February 2001.
- [RFC3258] *Distributing Authoritative Name Servers via Shared Unicast Addresses*, T. Hardie, April 2002.
- [RFC3901] *DNS IPv6 Transport Operational Guidelines*, A. Durand and J. Ihren, September 2004.

废弃和未实现的试验性RFC

- [RFC1712] *DNS Encoding of Geographical Location*, C. Farrell, M. Schulze, S. Pleitner, and D. Baldoni, November 1994.
- [RFC2673] *Binary Labels in the Domain Name System*, M. Crawford, August 1999.
- [RFC2874] *DNS Extensions to Support IPv6 Address Aggregation and Renumbering*, M. Crawford and C. Huitema, July 2000.

废弃的DNS安全RFC

- [RFC2065] *Domain Name System Security Extensions*, D. Eastlake, 3rd and C. Kaufman, January 1997.
- [RFC2137] *Secure Domain Name System Dynamic Update*, D. Eastlake, 3rd, April 1997.
- [RFC2535] *Domain Name System Security Extensions*, D. Eastlake, 3rd, March 1999.
- [RFC3008] *Domain Name System Security (DNSSEC) Signing Authority*, B. Wellington, November 2000.
- [RFC3090] *DNS Security Extension Clarification on Zone Status*, E. Lewis, March 2001.
- [RFC3445] *Limiting the Scope of the KEY Resource Record (RR)*, D. Massey and S. Rose, December 2002.
- [RFC3655] *Redefinition of DNS Authenticated Data (AD) bit*, B. Wellington and O. Gudmundsson, November 2003.
- [RFC3658] *Delegation Signer (DS) Resource Record (RR)*, O. Gudmundsson, December 2003.
- [RFC3755] *Legacy Resolver Compatibility for Delegation Signer (DS)*, S. Weiler, May 2004.
- [RFC3757] *Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag*, O. Kolkman, J. Schlyter, and E. Lewis, April 2004.
- [RFC3845] *DNS Security (DNSSEC) NextSECure (NSEC) RDATA Format*, J. Schlyter, August 2004.

C.2.2 互联网草案

互联网草案（ID）是互联网工程任务组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。

C.2.3 其它关于BIND的文档

References

- [1] *DNS and BIND*, Paul Albitz and Cricket Liu, Copyright © 1998 Sebastopol, CA: O'Reilly and Associates.

Appendix D

BIND 9 DNS库支持

D.1 BIND 9 DNS Library Support

本版本的BIND 9 “导出” 其内部库，使后者可以更容易地被第三方应用所利用（我们在本文档中将其称为“导出” 库）。除了BIND 9当前用到的所有主要的DNS相关的API，导出库还提供下列特性：

- 新创建的“DNS客户端” 模块。这是一个更高层的API，它为名字解析，与特定服务器之间的单次DNS事物，和动态更新提供了一个接口。关于名字解析，它支持诸如DNSSEC验证和缓存这样的高级特性。这个模块支持同步和异步模式。
- 新的“IRS”（Information Retrieval System，信息检索系统）库。它为分析传统的`resolv.conf`文件和这个包中其余应用所用到的，更高级的，DNS专用配置文件提供一个接口（参见下面对`dns.conf`文件的描述）。
- 作为IRS库的一部份，提供了新实现的标准地址-名字映射函数，`getaddrinfo()`和`getnameinfo()`。它们使用支持DNSSEC验证解析器后端，并能够使用其他BIND 9库的高级特性，如缓存。`getaddrinfo()`函数并发解析A和AAAA资源记录（当未指定地址族时）。
- 一个试验性框架，用于支持与BIND 9内部事件任务系统不一样的事件库。

D.1.1 先决条件

构建导出库要求用GNU `make`（BIND 9的其它部分仍然可以使用其它类型的`make`来构建）。在本文档的剩余部份，“`make`”表示GNU `make`。注意在某些平台上，你可能需要发起一个不同于“`make`”的命令名（例如“`gmake`”）来表示其是GNU `make`。

D.1.2 编译

```
$ ./configure --enable-exportlib [other flags]
$ make
```

这将会在通常的BIND 9程序之外，在`lib/export`目录下建立一套独立的库。例如，`lib/export/dns/libdns.a`是BIND 9 DNS库的导出版本的库文件。使用库的例子应用程序也将在`lib/export/samples`目录下构建（见下）。

D.1.3 安装

```
$ cd lib/export
```

```
$ make install
```

这 将 把 库 的 目 标 文 件 安 装 到 `--with-export-libdir` 配 置 选 项 所 指 定 的 目 录 下 （ 缺 省： `EPREFIX/lib/bind9` ）， 把 头 文 件 安 装 到 `--with-export-includedir` 配 置 选 项 所 指 定 的 目 录 下 （ 缺 省： `EPREFIX/include/bind9` ）。 通常 需 要 根 用 户 特 权。 在 顶 级 目 录 下 用 “**make install**” 也 会 做 同 样 的 事 情。

要 了 解 如 何 在 安 装 之 后 构 建 你 自 己 的 应 用， 参 见 `lib/export/samples/Makefile-postinstall.in`。

D.1.4 已知的缺陷/限制

- 当前，导出库不支持win32。（普通的BIND 9应用可以照旧构建）。
- （当前）导出库不支持“固定的”资源记录集顺序。如果你想在某些应用，如**named**，使用“固定的”资源记录集顺序而同时仍然向在没有固定的顺序支持的情况下构建导出库，就分别构建他们：

```
$ ./configure --enable-fixed-rrset [other flags, but not --enable-exportlib]
$ make
$ ./configure --enable-exportlib [other flags, but not --enable-fixed-rrset]
$ cd lib/export
$ make
```

- 当前，客户端模块和IRS库不支持使用DLV的DNSSEC验证（下层模块可以处理后者，但是没有可调的借口来打开这个特性）。
- 导出库的验证存根解析器不支持RFC 5011。实际上，是否应该支持并不明确：信任锚应该是一项系统范围的配置，它应该由一名系统管理员来管理，同时，存根解析器是由普通用户运行的普通应用所使用。
- 不是所有通常的 `/etc/resolv.conf` 选项都在IRS库中有支持。在当前版本中，能够使用的选项仅有“**debug**”和“**ndots**”。

D.1.5 dns.conf文件

IRS库支持一个与DNS库相关的“高级”配置文件，其配置参数会超越 `resolv.conf` 的功能。特别地，它还打算提供DNSSEC相关的配置参数。缺省时，这个配置文件的路径为 `/etc/dns.conf`。这个模块极具试验性，其配置语法和库接口在将来的版本中可能会有变化。当前，只支持**trusted-keys**语句，其语法与 `named.conf` 中的同名语句一样。（更详细的内容，参见第6.2.21节。）

D.1.6 例子应用

这里提供了一些使用这个API的例子应用程序供参考。下面是对这些应用的一个简要描述。

D.1.6.1 sample: 一个简单的存根解析器应用

它向一个指定的递归服务器发送一个指定名字（作为选项，也可指定资源记录类型）的请求，并将结果打印为一个资源记录的列表。如果通过命令行选项的集合给出了一个信任锚，它也可以用作一个验证存根解析器。

Usage: **sample** [options] server_address hostname

选项和参数：

-t RRtype 指定请求的资源记录类型。缺省是A。

[-a algorithm] [-e] -k keyname -K keystack 在命令行指定一个DNS密钥，用于验证响应。例如，指定下面example.com的DNSKEY：

```
example.com. 3600 IN DNSKEY 257 3 5 xxx
```

以下指定选项：

```
-e -k example.com -K "xxx"
```

-e表示这个密钥是一个区的“密钥签名密钥”（也被称为“安全入口点”）。当省略**-a**时，缺省使用rsasha1。

-s domain:alt_server_address 为特定“域”指定一个单独的递归服务器地址。例如：**-s example.com:2001:db8::1234**

server_address 请求发向的递归服务器的一个IP（v4/v6）地址。

hostname 请求的域名

D.1.6.2 sample-async: 一个例子存根解析器，异步工作

与“sample”相似，但是以一个独立文件方式接受（请求）域名清单并异步解析这些名字。

Usage: sample-async [-s server_address] [-t RR_type] input_file

选项和参数：

-s server_address 请求发向的递归服务器的一个IPv4地址。（在这个实现中不支持IPv6地址）

-t RR_type 指定请求的资源记录类型。缺省是A。

input_file 要解析域名的清单。每行由一个域名组成。例如：

```
www.example.com
mx.example.net
ns.xxx.example
```

D.1.6.3 sample-request: 一个简单的DNS事物客户端

它向一个特定的服务器发送一个请求，并输出经过最小处理的响应。它不充当一个“存根解析器”：一旦它从服务器获得任何响应，就停止处理进程，无论响应是否一个引用或者别名（CNAME或DNAME），本该需要以更多的请求来获得最终的答案。换句话说，这个应用扮演了一个非常简化的**dig**。

Usage: sample-request [-t RRtype] server_address hostname

选项和参数：

-t RRtype 指定请求的资源记录类型。缺省是A。

server_address 请求发向的递归服务器的一个IP（v4/v6）地址。

hostname 请求的域名

D.1.6.4 sample-gai: getaddrinfo()和getnameinfo()测试代码

这是一个用于检查getaddrinfo()和getnameinfo()行为的测试程序。它使用一个主机名作为测试，以给定的主机名调用getaddrinfo()，并用getaddrinfo()所返回的结果IP地址调用getnameinfo()。如果dns.conf文件存在并且定义了一个信任锚，下层解析器充当一个验证解析器，并且在DNSSEC验证失败时，getaddrinfo()/getnameinfo()会失败并返回一个EALINSECUREDATA错误。

Usage: sample-gai hostname

D.1.6.5 sample-update: 一个简单的动态更新客户端程序

它接受一个更新命令作为一个命令行参数，发送一个更新请求消息给权威服务器，并输出从服务器回来的响应。换句话说，这就是一个简化版的nsupdate。

Usage: sample-update [options] (add|delete) "update data"

选项和参数：

-a auth_server 作为包含所更新名字的区的权威服务器的一个IP地址。这通常应该是接受动态更新的主权威服务器。它也可以是一个被配置为向主服务器转发更新请求的辅服务器。

-k keyfile 一个TSIG密钥文件，用于保证更新事务的安全。密钥文件格式与nsupdate实用程序所用的一致。

-p prerequisite 一个更新的先决条件（只能指定一个先决条件）。这个先决条件的格式与nsupdate实用程序所能接受的一致。

-r recursive_server 这个应用将会用到的一个递归服务器的一个IP地址。必须使用一个递归服务器来获取权威服务器的地址，再向后者发送更新请求。

-z zonename The domain name of the zone that contains

(add|delete) 指定更新操作的类型。必须是“add”或者“delete”之一。

“update data” 指定要改更新的数据。典型的数据例子看起来像“name TTL RRtype RDATA”。

注意



实际上，必须指定-a或者-r。其它是可选的；下层的库例程试图为更新标识合适的服务器和区名。

例如：假设dynamic.example.com区的主权威服务器有一个IPv6地址2001:db8::1234,

```
$ sample-update -a sample-update -k Kxxx.+nnn+mmmm.key add "foo.dynamic.example.com 30
```

使用给定的密钥为foo.dynamic.example.com增加一条A记录。

```
$ sample-update -a sample-update -k Kxxx.+nnn+mmmm.key delete "foo.dynamic.example.com
```

使用给定的密钥删除foo.dynamic.example.com的所有A记录。

```
$ sample-update -a sample-update -k Kxxx.+nnn+mmmm.key delete "foo.dynamic.example.com
```

使用给定的密钥删除foo.dynamic.example.com的所有记录。

D.1.6.6 nsprobe: domain/name server checker in terms of RFC 4074

它检查一个域名集合，按照RFC 4074的描述查看域名的名字服务器的行为是否正确。它被包含在一系列例子程序的集合中，这些程序展示了如何将导出库用于DNS相关的应用中。

Usage: nsprobe [-d] [-v [-v...]] [-c cache_address] [input_file]

Options

-d 在“debug”模式运行。使用这个选项，nsprobe将会转储其所收到的每个资源记录。

-v 增加其它普通日志消息的可见性。这个可以被多次指定。

-c cache_address 指定一个递归（缓存）名字服务器的一个IP地址。nsprobe使用这个服务器来获取每个域名的NS资源记录集和名字服务器的A和/或AAAA资源记录集。缺省值是127.0.0.1。

input_file 一个包含了需要被探测的域（区）名清单的文件的文件名。输入文件的每行指定了一个域名，如“example.com”。通常这个域名必须是某些DNS区的顶点名（而不是普通的“主机名”如“www.example.com”）。nsprobe首先识别给定域名的NS资源记录集，并向这些服务器发送A和AAAA请求，请求区下某些“广泛使用的”名字；特别地，在区名前增加“www”和“ftp”。

D.1.7 库参考

截至目前撰写的文稿，这个库还没有正式的“手册”，除了本文档，头文件（其中一些提供了非常详细的解释），和例子应用程序。

Appendix E

手册页

E.1 dig

名字

dig — DNS查找工具

概要

```
dig [@server] [-b address] [-c class] [-f filename] [-k filename] [-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...]
```

描述

dig (domain information groper, 域信息搜索者) 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, 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。

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” 。

简单用法

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。

选项

`-b`选项设置到地址`address`的请求的源地址。这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。

缺省的请求类 (IN, 表示internet) 被`-c`选项所覆盖。`class`是任何一个有效的类, 如HS, 表示Hesiod记录, 或者CH, 表示CHAOSNET记录。

`-f`选项让**dig**以批处理模式工作, 从文件`filename`中读入要查找请求的列表, 并进行处理。文件包含了一组请求, 每行一个。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。

如果要在一个非标准的端口进行查询, 使用`-p`选项。`port#`是**dig**请求发向的端口号, 而不是标准DNS端口号53。这个选项可以用于测试一个名字服务器, 将其配置成监听在一个非标准端口号。

`-4`选项强制**dig**仅使用IPv4传输请求。`-6`选项强制**dig**仅使用IPv6传输请求。

`-t`选项设置请求类型为`type`。它可以是BIND 9所支持的任何有效请求类型。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`。

`-q`选项设置请求名为`name`。这个帮助区别`name`和其它参数。

`-v`使**dig**打印出版本号并退出。

反向查找— 将地址映射到名字— 很简单, 只需设置`-x`选项。`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。在使用这个选项时, 不需要提供`name`, `class`和`type`参数。**dig**自动执行一个类似`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地址使用半字节格式在IP6.ARPA域名下面查找。要使用旧的RFC1886中的方法而使用IP6.INT域, 需要指定`-i`选项。位串标记 (RFC2874) 现在是试验性的, 最好不要使用。

为对**dig**发出的DNS请求和其使用事务签名 (transaction signatures, TSIG) 的响应, 使用`-k`选项指定一个TSIG密钥。你也可以在命令行使用`-y`选项指定TSIG密钥本身; `hmac`是TSIG的类型, 缺省是HMAC-MD5, `name`是TSIG密钥的名字, `key`是实际的密钥。密钥是一个base-64编码的字符串, 典型情况是由`dnssec-keygen(8)`所生成。在多用户系统上使用`-y`选项应当谨慎, 因为密钥可以在`ps(1)`的输出中或者在`shell`的历史文件中被看到。在**dig**中使用TSIG认证时, 所请求的名字服务器需要知道所用的密钥和算法。在BIND中, 这是通过在`named.conf`中使用合适的`key`和`server`语句来完成的。

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。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]besteffort` 试图显示坏包消息的内容。缺省是不显示坏包回答。

+`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。这请求服务器不对响应执行DNSSEC验证。

+`[no]cl` 打印记录时显示[不显示]类。

+`[no]cmd` 翻转打印在输出中标识**dig**版本和所应用的请求选项的初始注释的状态。这个注释缺省是要打印的。

+`[no]comments` 翻转在输出中显示注释行的状态。缺省是打印注释。

+`[no]crypto` 翻转对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。

+`[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]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]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]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]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]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`中必须出现的点的个数为`D`，`name`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。

`resolv.conf`中的`'ndots'`(缺省是1)，它可以被`+ndots`覆盖，决定了名字是否被作为相对名字对待，以及因此是否最终执行一个搜索。

+`[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]qr` 打印[不打印]所发出的请求，缺省是不打印请求。

+`[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]recurse` 翻转请求中的RD（期望递归）位设置。这个位缺省是置位的，意味着**dig**普通情况是发送递归的请求。在使用了`+nssearch`或者`+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` 翻转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`searchlist`或者`domain`命令所定义的搜索列表。缺省是不使用搜索列表。

+`[no]short` 提供一个简洁的回答。缺省是以冗长形式打印回答。

+`[no]showsearch` 执行[不执行]立即显示结果的搜索。

+`[no]sigchase` 追随DNSSEC签名链。要求dig编译时带-DDIG.SIGCHASE。

+`[no]sit [=####]` 发送一个源特征符号 (Source Identity Token) EDNS选项, 带有可选值。从一个先前的响应重放一个SIT将允许服务器识别一个先前的客户端。缺省值是+nosit。当前使用试验值65001作为选项码。

+`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被离散到距其最近的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` 这个请求选项翻转打印统计的选项: 生成请求的时间, 响应的大小等等。缺省行为是打印统计。

+`[no]subnet=addr/prefix` 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。

+`[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, 除非要求一个ixfr=N的请求, 这种情况下缺省是TCP。AXFR请求总是使用TCP。

+`time=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` 在追随DNSSEC签名链时执行至上而下的验证。要求dig编译时带-DDIG.SIGCHASE。

+`[no]trace` 翻转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。

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=####` 指定一个文件, 其包含用于+sigchase的信任密钥。每个DNSKEY必须在其本身行。

如果没有指定, dig会查找/etc/trusted-key.key, 然后在当前目录中查找trusted-key.key。

要求dig编译时带-DDIG.SIGCHASE。

+`[no]ttlid` 在打印记录时显示[不显示]TTL。

+`[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”。

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。

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，就定义IDN_DISABLE环境变量。在**dig**运行时，如果变量已设置，IDN支持就是关闭的。

文件

```
/etc/resolv.conf
```

```
${HOME}/.digrc
```

参见

`host(1)`, `named(8)`, `dnssec-keygen(8)`, *RFC1035*.

BUGS

有可能是太多的请求选项。

E.2 host

名字

host — DNS查找工具

概要

```
host [-aCdlnrsTwv] [-c class] [-N ndots] [-R number] [-t type] [-W wait]
    [-m flag] [-4] [-6] [-v] [-V] name [server]
```

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。

-a（全部）选项与设置-v选项并让**host**发一个ANY类型的请求等效。

当使用了-c选项，**host**会试图从这个区的所有列出的权威名字服务器中显示区*name*的SOA记录。名字服务器的列表是由所找到的这个区的NS记录所定义的。

-c选项指示生成一个类*class*的请求。这个可以用于查找Hesiod或者Chaosnet类的资源记录。缺省类是IN（Internet）。

在使用-d或-v选项时，**host**产生冗余输出。这两个选项是等效的。它们提供了向后兼容性。在先前的版本，-d选项打开调试跟踪而-v打开冗余输出。

-l选项选择列表模式。这使**host**对区*name*执行一个区传送。传送区并打印出NS，PTR和地址记录（A/AAAA）。如果与-a组合，就打印全部记录。

-i选项指定对IPv6地址的反向查找应该使用IP6.INT域，这个域是在RFC1886中定义的。缺省是使用IP6.ARPA。

-n选项设置出现在被当做决定名字*name*中的点的数目。缺失值是在/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。

一个查找的UDP重试次数可以使用-R选项改变。*number*指示**host**将会在没有收到回答时重发多少次请求。缺省的重试次数是1。如果*number*是负数或为0，重试次数缺省为1。

可以通过-r选项生成非递归请求。设置这个选项清除**host**生成的请求中的RD — 期望递归— 位。这意味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，这些回答通常是对其它名字服务器指向。

缺省情况，**host**使用UDP生成请求。-T选项使其在请求名字服务器时使用一个TCP连接。在需要时，会自动为请求选择TCP，例如区传送（AXFR）请求。

-4选项强制**host**只使用IPv4请求传输。-6选项强制**host**只使用IPv6请求传输。

-t选项用于选择请求类型。*type*可以是任何可识别的请求类型：CNAME, NS, SOA, SIG, KEY, 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，但是如果给出-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（例如，-t IXFR=12345678）来指定开始序列号。

等待响应的时间可以通过-w和-W选项来控制。-w选项使**host**等待*wait*秒。如果*wait*小于1，等待间隔就设置为1秒。如果使用-W选项，**host**将有效地永远等待一个响应。等待一个响应的时间可以设置成由硬件的最大整数所给出的秒数。

-s选项告诉**host**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。

-m可以用于设置内存使用调试标志*record*，*usage*和*trace*。

-v选项使**host**打印版本号并退出。

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。

文件

/etc/resolv.conf

参见

dig(1), named(8).

E.3 delv

名字

delv — DNS查找和验证工具

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...]
```

描述

delv (Domain Entity Lookup & Validation, 域名实体查找和验证)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, DS和DLV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, 响应使用内置根区 (“.”) 以及ISC DNSSEC后备验证区 (“dlv.isc.org”) 的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。

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。::1 for IPv6).

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。

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，并含有根区（"."）和ISC DNSSEC后备验证区（"`dlv.isc.org`"）的信任锚点。

与根或DLV信任锚点名字不匹配的密钥会被忽略；这些密钥名字可以使用`+dlv=NAME`或者`+root=NAME`选项覆盖。

注意：在读取信任锚点文件时，**delv**同等对待`managed-keys`语句和`trusted-keys`语句。即，对于一个被管理的密钥，它就被信任的初始密钥；RFC 5011密钥管理不被支持。**delv**将不咨询由**named**维护的被管理密钥数据库。这意谓如果`/etc/bind.keys`有一个密钥被撤销并轮转，就必须更新`/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。As with `-q`, this is useful to distinguish query name type or class when they are ambiguous. 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。

请求选项

`delv` 提供一些请求选项，它们影响结果的显式方式，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]`ttdl 控制在打印一个记录时是否显示TTL。缺省是显示TTL。

+`[no]`rtrace 翻转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 翻转消息日志。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 翻转验证日志。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** 翻转在输出中显示注释。缺省是打印注释。
- + [no] rrcomments** 翻转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** 翻转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。output.
- + [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和+nodlv禁止。
- + [no] root [=ROOT]** 指示是否执行传统的（非后备）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] dlv [=DLV]** 指示是否执行DNSSEC后备验证，如果执行，就指定DLV信任锚点的名字。缺省是使用信任锚点"dlv.isc.org"执行后备验证，对这个锚点存在一个内置密钥。如果指定一个不同的名字，就必须使用-a 指定一个包含DLV密钥的文件。

文件

/etc/bind.keys
/etc/resolv.conf

参见

dig(1), named(8), RFC4034, RFC4035, RFC4431, RFC5074, RFC5155.

E.4 dnssec-checkds

名字

dnssec-checkds — 一个DNSSEC授权一致性检查工具

概要

```
dnssec-checkds [-l domain] [-f file] [-d dig path] [-D dsfromkey path] zone
dnssec-dsfromkey [-l domain] [-f file] [-d dig path] [-D dsfromkey path]
zone
```

描述

dnssec-checkds 为指定区中的密钥验证授权签名者（DS）或者DNSSEC旁观验证（DLV）资源记录的正确性。

选项

-f file 如果指定了一个file，就在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。

-l domain 在指定的旁观域中核对一个DLV记录，而不是与本区的父区中一个DS记录进行核对。例如，要在ISC的DLV区中核对“example.com”的DLV记录，使用：**dnssec-checkds -l dlv.isc.org example.com**

-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。

参见

dnssec-dsfromkey(8), dnssec-keygen(8), dnssec-signzone(8),

作者

Internet Systems Consortium

E.5 dnssec-coverage

名字

dnssec-coverage — 检查一个区DNSKEY将来的覆盖范围

概要

```
dnssec-coverage [-K directory] [-l length] [-f file] [-d DNSKEY TTL] [-m
max TTL] [-r interval] [-c compilezone path] [-k] [-z] [zone]
```

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覆盖的间隔。

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。）

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，区就在那个文件读取；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。

这个选项是必须的，除非使用-f指定了一个区文件。（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。）

-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。

这个选项是必须的，除非使用-f指定了一个区文件，或者使用-l给**dnssec-keygen**设置了一个缺省的密钥TTL。（如果使用了上述两者之一，仍然可以使用这个选项；它将覆盖在区或密钥文件中发现的值。）

-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。

参见

dnssec-checkds(8), dnssec-dsfromkey(8), dnssec-keygen(8), dnssec-signzone(8)

作者

Internet Systems Consortium

E.6 dnssec-dsfromkey

名字

dnssec-dsfromkey — DNSSEC DS资源记录生成工具

概要

```
dnssec-dsfromkey [-v level] [-1] [-2] [-a alg] [-l domain] [-T TTL] keyfile
dnssec-dsfromkey -s [-v level] [-1] [-2] [-a alg] [-K directory] [-l
    domain] [-s] [-c class] [-T TTL] [-f file] [-A] [-v level] dnsname
dnssec-dsfromkey [-h] [-V]
```

描述

dnssec-dsfromkey 为给定密钥输出在RFC 3658和RFC 4509中所定义的授权签名者（DS）资源记录（RR）。

选项

-1 使用SHA-1作为摘要算法（缺省是使用SHA-1和SHA-256）。

-2 使用SHA-256作为摘要算法。

-a algorithm 选择摘要算法。algorithm的值必须是SHA-1（SHA1），SHA-256（SHA256），GOST或SHA-384（SHA384）之一。这些值是大小写不敏感的。

-T TTL 指定DS记录的TTL。

-K directory 在directory中查找密钥文件（或者在keyset模式时，查找keyset-文件）。

-f file 区文件模式：作为密钥文件名的替代，此参数是一个区主文件的域名，它可以从file中读入。如果区名与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。这使使用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
```

-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。仅用于区文件模式。

-l domain 生成一个DLV集合而不是一个DS集合。所指定的domain被添加到集合中每条记录的名字。DNSSEC旁观验证（DLV）资源记录在RFC 4431中描述。

-s 密钥集合模式：替代密钥文件名字，参数是一个密钥集合文件的DNS域名。

-c class 指定DNS类（缺省是IN），仅用于密钥集合（keyset）或区文件模式。

-v level 设置调试级别。

-h 打印用法信息。

-V 打印版本信息。

例子

要从Kexample.com.+003+26160密钥文件（keyfile）名构建SHA-256 DS资源记录，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 3A1EADA7A74B8D0BA86726B0C227AA85AB8BBD2B2004F41A868A54F0C5EA0B94
```

文件

密钥文件可以根据密钥标识Knnnnn.+aaa+iiiiii 或者由dnssec-keygen(8) 所生成的完整文件名Knnnnn.+aaa+iiiiii.key来设计。

密钥集合文件名是从directory，字符串keyset-和dnsname中构建的。

注意

即使文件存在，一个密钥文件错误也会给出一个“文件不存在”消息。

参见

dnssec-keygen(8), dnssec-signzone(8), BIND 9管理员参考手册, RFC 3658, RFC 4431. RFC 4509.

作者

Internet Systems Consortium

E.7 dnssec-importkey

名字

dnssec-importkey — 从外部系统导入DNSKEY记录从而可对其进行管理。

概要

```
dnssec-importkey [-K directory] [-L ttl] [-P date/offset] [-D date/offset]  
                 [-h] [-v level] [-V] keyfile
```

```
dnssec-importkey -f filename [-K directory] [-L ttl] [-P date/offset] [-D  
                        date/offset] [-h] [-v level] [-V] [dnsname]
```

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，这意味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。

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。

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。

-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。）

文件

可以由密钥标识Knnnn.+aaa+iiiii或者dnssec-keygen(8)生成的完整文件名Knnnn.+aaa+iiiii.key来决定。

参见

dnssec-keygen(8), dnssec-signzone(8), BIND 9 Administrator Reference Manual, RFC 5011.

作者

Internet Systems Consortium

E.8 dnssec-keyfromlabel

名字

dnssec-keyfromlabel — DNSSEC密钥生成工具

概要

```
dnssec-keyfromlabel -l label [-3] [-a algorithm] [-A date/offset] [-c
class] [-D date/offset] [-E engine] [-f flag] [-G] [-I date/offset] [-i
interval] [-k] [-K directory] [-L ttl] [-n nametype] [-P date/offset]
[-p protocol] [-R date/offset] [-S key] [-t type] [-v level] [-V] [-y]
name
```

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。

选项

-a algorithm 选择加密算法。algorithm的值必须为RSAMD5, RSASHA1, DSA, NSEC3RSASHA1, NSEC3DSA, RSASHA1。这些值是大小写不敏感的。

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。)

注1: 对DNSSEC来说, RSASHA1是必须实现的算法, 而DSA是推荐的。

注2: DH自动设置-k标志。

-3 使用NSEC3兼容算法生成一个DNSSEC密钥。如果使用了这个选项并且没有在命令行显式设定算法, 缺省将使用NSEC3RSASHA1。

-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。它可能由一个可选的OpenSSL引擎名开始, 后跟一个冒号, 如同“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, 其它日期也可能在那儿设置 (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，这就会被用作密钥的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。）

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”。

-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 设置密钥被撤销的日期。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。）

-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。

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。

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)*。

作者

Internet Systems Consortium

E.9 dnssec-keygen

名字

`dnssec-keygen` — DNSSEC 密钥生成工具

概要

```
dnssec-keygen [-a algorithm] [-b keysize] [-n nametype] [-3] [-A
    date/offset] [-C] [-c class] [-D date/offset] [-E engine] [-f flag]
    [-G] [-g generator] [-h] [-I date/offset] [-i interval] [-K directory]
    [-L ttl] [-k] [-P date/offset] [-p protocol] [-q] [-R date/offset] [-r
    randomdev] [-S key] [-s strength] [-t type] [-v level] [-V] [-z name
```

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。

选项

-a algorithm 选择加密算法。对DNSSEC密钥，algorithm的值必须为RSAMD5, RSASHA1, DSA, NSEC3RSASHA1, NSEC3DSA, RSASHA256, RSASHA512, ECCGOST, ECDSAP256SHA256或ECDSAP384SHA384之一。对TSIG/TKEY，其值必须为DH (Diffie Hellman)，HMAC-MD5, HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, HMAC-SHA384或HMAC-SHA512之一。这些值是大小写不敏感的。

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。）

注1：对DNSSEC来说，RSASHA1是必须实现的算法，而DSA是推荐的。对TSIG来说，HMAC-MD5是必须的。

注2：DH, HMAC-MD5以及从HMAC-SHA1到HMAC-SHA512自动设置-T KEY选项。

-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512和2048位之间。Diffie Hellman密钥必须在128和4096位之间。DSA密钥必须在512和1024位之间并且是64的整数倍。HMAC密钥必须在1和512位之间。椭圆曲线算法不需要这个参数。

如果使用缺省算法，不需要指定密钥的大小。对区签名密钥（ZSK），缺省密钥大小是1024位，而对密钥签名密钥（KSK，由-f KSK所生成），缺省是2048位。然而，如果使用-a显式指定了一个算法，就没有缺省的密钥大小了，就必须使用-b。

-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。

-3 使用NSEC3兼容算法生成一个DNSSEC密钥。如果使用了这个选项并且没有在命令行显式设定算法，缺省将使用NSEC3RSASHA1。注意RSASHA256, RSASHA512, ECCGOST, ECDSAP256SHA256和ECDSAP384SHA384算法是NSEC3兼容的。

-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-keygen**将在存放于私钥的元数据中包含密钥的创建日期，其它日期也可能在那儿设置（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。

-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 废除的-T KEY。
- L *t1* 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 就不设置它有同样的效果。
- p *protocol* 为生成密钥设置协议值。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。
- 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。
- r *randomdev* 指定随机性的源。如果操作系统不提供/dev/random或等效的设备，缺省的随机性的源为键盘输入。randomdev指定一个字符设备名或包含随机数据的文件名，用来替代缺省值。特定值keyboard指示使用键盘输入。
- 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。
使用任何TSIG算法（HMAC-*或DH）强制这个选项为KEY。
- 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。

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”。

-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 设置密钥被撤销的日期。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。）

-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。

生成的密钥

当**dnssec-keygen**完全成功时，它打印一个Knnnn.+aaa+iiii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。
- iiii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+iiii.key包含公钥，而Knnnn.+aaa+iiii.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。

.key和.private文件都是由对称加密算法，如HMAC-MD5，生成，即使公钥和私钥是等价的。

例子

要为域**example.com**生成一个768位的DSA密钥，需要执行下列命令：

```
dnssec-keygen -a DSA -b 768 -n ZONE example.com
```

命令将会打印下列格式的字符串：

```
Kexample.com.+003+26160
```

在这个例子中，**dnssec-keygen**建立文件**Kexample.com.+003+26160.key**和**Kexample.com.+003+26160.private**。

参见

dnssec-signzone(8), BIND 9管理员参考手册, RFC 2539, RFC 2845, RFC 4034.

作者

Internet Systems Consortium

E.10 dnssec-revoke

名字

dnssec-revoke — 设置一个DNSSEC密钥中的REVOKED位

概要

```
dnssec-revoke [-hr] [-v level] [-V] [-K directory] [-E engine] [-f] [-R]
               keyfile
```

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。

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集之后删去原来的密钥集合文集。files.

-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。

参见

dnssec-keygen(8), BIND 9管理员参考手册, RFC 5011.

作者

Internet Systems Consortium

E.11 dnssec-settime

名字

dnssec-settime — 为一个DNSSEC密钥设置密钥定时元数据

概要

```
dnssec-settime [-f] [-K directory] [-L ttl] [-P date/offset] [-A
date/offset] [-R date/offset] [-I date/offset] [-D date/offset] [-h]
[-V] [-v level] [-E engine] keyfile
```

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 (Knnnnn.+aaa+iiiii.key和Knnnnn.+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)。

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, 这就会被用作密钥的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。

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。

-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 设置密钥被退休的日期。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。）

-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。

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/A/R/I/D/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，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，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。使用-p all打印所有的元数据。

参见

dnssec-keygen(8), dnssec-signzone(8), BIND 9管理员参考手册, RFC 5011.

作者

Internet Systems Consortium

E.12 dnssec-signzone

名字

dnssec-signzone — DNSSEC区签名工具

概要

```
dnssec-signzone [-a] [-c class] [-d directory] [-D] [-E engine] [-e
end-time] [-f output-file] [-g] [-h] [-K directory] [-k key] [-L
serial] [-l domain] [-M domain] [-i interval] [-I input-format] [-j
jitter] [-N soa-serial-format] [-o origin] [-O output-format] [-P] [-p]
[-R] [-r randomdev] [-S] [-s start-time] [-T ttl] [-t] [-u] [-v level]
[-V] [-X extended end-time] [-x] [-z] [-3 salt] [-H iterations] [-A]
zonefile [key...]
```

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。

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。

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。

-l domain 在生成密钥（DNSKEY）和DS集合之外还生成一个DLV集合。域被添加到记录名之后。

-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称。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”和“unixtime”。

“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。

-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 在签名区时使用伪随机数据。这样比使用真随机数据更快，但安全性较低。这个选项在签名区较大或者在熵源有限的时候可能会有用。

-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第4.2.1.1部份（“Pre-Publish Key Rollover”）中描述的过程进行轮转。

-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第4.2.1.2部份（“Double Signature Zone Signing Key Rollover”）中描述的过程进行轮转。

-r randomdev 指定随机性的源。如果操作系统不提供/dev/random或等效的设备，缺省的随机性的源为键盘输入。randomdev指定一个字符设备名或包含随机数据的文件名，用来替代缺省值。特定值keyboard指示使用键盘输入。

-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。

-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，使用这么多次循环。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。

例子

下列命令使用由**dnssec-keygen**所生成的DSA密钥（Kexample.com.+003+17247）对**example.com**区签名。因为现在没有使用**-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.+003+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
%
```

参见

dnssec-keygen(8), BIND 9管理员参考手册, RFC 4033, RFC 4641.

作者

Internet Systems Consortium

E.13 dnssec-verify

名字

dnssec-verify — DNSSEC区验证工具

概要

```
dnssec-verify [-c class] [-E engine] [-I input-format] [-o origin] [-v
level] [-V] [-x] [-z] zonefile
```

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。

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。

-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。

参见

dnssec-signzone(8), *BIND 9 Administrator Reference Manual*, RFC 4033.

作者

Internet Systems Consortium

E.14 named-checkconf

名字

named-checkconf — named配置文件语法检查工具

概要

```
named-checkconf [-h] [-v] [-j] [-t directory] filename [-p] [-x] [-z]
```

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。

选项

-h 打印用法摘要并退出。

-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。

-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。

-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。

-j 在装载一个区文件时，如果存在日志文件，就读入。

filename 所要检查的配置文件的名称。如果未指定，缺省为/etc/named.conf。

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。

参见

named(8), named-checkzone(8), BIND 9管理员参考手册。

作者

Internet Systems Consortium

E.15 named-checkzone

名字

named-checkzone, named-compilezone — 区文件正确性检查和转换工具

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] -o filename 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] [-w directory] [-D] [-W
                  mode] zonename filename
```

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。

选项

-d 打开调试。

-h 打印用法摘要并退出。

-q 安静模式- 仅仅有退出码。

-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**）。日志文件名是由区文件名后加上字符串 `.jnl`。

-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。

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。

参见

named(8), named-checkconf(8), RFC 1035, BIND 9管理员参考手册。

作者

Internet Systems Consortium

E.16 named

名字

named — 互联网域名服务器

概要

```
named [-4] [-6] [-c config-file] [-d debug-level] [-D string] [-E
engine-name] [-f] [-g] [-m flag] [-n #cpus] [-p port] [-s] [-S
#max-socks] [-t directory] [-U #listeners] [-u user] [-v] [-V] [-x
cache-file]
```

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, 1034, 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。

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实例的字符串。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。

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（如，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。

-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。

注意



这个选项主要是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。

-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4096，在使用“configure --with-tuning=large”构建的系统上缺省值为21000。

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。还要注意实际的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，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1，2-4个CPU为2，大于4个CPU为检测到的CPU个数除以2。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。

-u user 在完成特权操作后，设置用户ID(setuid)为用户，这样就可以创建套接字，使其监听在特权端口上。

注意

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，这意味着当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 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。

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。

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**。

文件

/etc/named.conf 缺省配置文件。

/var/run/named/named.pid 缺省进程ID文件。

参见

RFC 1033, RFC 1034, RFC 1035, **named-checkconf(8)**, **named-checkzone(8)**, **rndc(8)**, **lwresd(8)**, **named.conf(5)**, BIND 9管理员参考手册。

作者

Internet Systems Consortium

E.17 named-journalprint

名字

named-journalprint — 以人可读的格式打印区日志文件

概要

named-journalprint journal

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，日志文件的名称由相应区文件的名称加上扩展名.jnl组成。zone file.

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。

参见

named(8), nsupdate(8), BIND 9管理员参考手册.

作者

Internet Systems Consortium

E.18 named-rrchecker

名字

named-rrchecker — 一个针对单个DNS资源记录的语法检查器

概要

```
named-rrchecker [-h] [-o origin] [-p] [-u] [-C] [-T] [-P]
```

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。

参见

RFC 1034, RFC 1035, named(8)

E.19 nsupdate

名字

nsupdate — 动态DNS更新工具

概要

```
nsupdate [-d] [-D] [-g | -o | -l | -y [hmac:]keyname:secret | -k keyfile]
          [-t timeout] [-u udptimeout] [-r udpretries] [-R randomdev] [-v] [-T]
          [-P] [-V] [filename]
```

描述

nsupdate 是用于提交在RFC 2136中所定义动态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。

-d选项使**nsupdate**运行在调试模式。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选项让**nsupdate**报告比-d更多的附加调试信息。

-L选项带有一个整数参数，为0或更大，用于设置日志的调试级别。如果为0，就关掉日志。

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。当前，TSIG中唯一支持的加密算法是在RFC 2104中所定义的HMAC-MD5。一旦为TSIG定义了其它算法，应用程序需要确保在相互认证时，它们选择合适的算法，如同选择合适的密钥一样。例如，合适的key和server语句会被添加到/etc/named.conf，这样名字服务器可以将合适的密钥和算法关联到将会使用TSIG认证的客户端应用所在的IP地址上。SIG(0)使用概要加密算法。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。**nsupdate**不会读/etc/named.conf。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。

nsupdate 使用-y或-k选项提供生成一个认证动态DNS更新请求的TSIG记录所需的共享密钥，缺省是HMAC-MD5。这些选项是互斥的。

当使用了-y选项时，从[hmac:]keyname:secret 生成一个签名。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。注意：不鼓励使用-y选项，因为共享密钥是以文本形式作为命令行参数提供的。在ps(1)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。

使用-k选项，**nsupdate**从文件keyfile读入共享密钥。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。

nsupdate可以用于一个只local-host的模式，通过使用-l标志。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，这个密钥可以由named 自动生成。这个密钥文件的位置可以使用-k 选项覆盖。

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。-v选项让**nsupdate**使用一个TCP连接。当有一批更新请求时，这个可能是更优的。

-p设置用于连接一个名字服务器的缺省端口。缺省为53。

-t选项设置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。

-u选项设置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。

-r选项设置UDP重试次数。缺省是3。如果为0，仅仅会生成一次更新请求。

-R randomdev选项指定一个随机性的源。如果操作系统不提供/dev/random或等效的设备，缺省的随机性的源为键盘输入。**randomdev**指定一个字符设备名或包含随机数据的文件名，用来替代缺省值。特定值**keyboard**指示使用键盘输入。这个选项可以指定多次。

其它类型可以使用“TYPEXXXXX”键入，这里“XXXXX”是不以零开始的类型的十进制值。如果有资源数据（**rdata**），它将被分析成使用UNKNOWN资源数据格式，（<backslash> <hash> <space> <length> <space> <hexstring>）。

-T和**-P**选项打印输出非元类型的列表，对这些类型来说，其类型专用的表示格式是已知的。**-T**打印输出由IANA分配类型的列表。**-P**打印输出指定给**named**的私有类型的列表。这些选项可以组合。**nsupdate**将在打印列表后退出。

-V选项使**nsupdate**打印版本号并退出。

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于区中，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给名字服务器。当没有提供**local**语句时，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**。**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。

[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] del[ete] 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。

以一个分号开始的行是注释并被忽略。

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。）

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-keygen(8)所创建的HMAC-MD5密钥的base-64编码。

K{name}+.157.+{random}.private 由dnssec-keygen(8)所创建的HMAC-MD5密钥的base-64编码。

参见

RFC 2136, RFC 3007, RFC 2104, RFC 2845, RFC 1034, RFC 2535, RFC 2931, named(8), ddns-confgen(8), dnssec-keygen(8).

BUGS

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。

E.20 rndc

名字

rndc — 名字服务器控制工具

概要

```
rndc [-b source-address] [-c config-file] [-k key-file] [-s server] [-p
port] [-q] [-V] [-y key.id] command
```

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。他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。

选项

-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。

-V 打开冗余日志。

-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。

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：

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。

refresh zone [class [view]] 对指定的区进行区维护。

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。

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**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”。）

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**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”。）

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。

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。

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。

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。

notify zone [class [view]] 重新发出区的NOTIFY消息。

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。

stats 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。

stats 写服务器的统计信息到统计文件。

querylog 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**；。

dumpdb [-all|-cache|-zone] [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。

secroots [view ...] 转储服务器的安全根到指定视图的**secroots**文件中。如果没有指定视图，就转储所有视图的安全根。

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的进程号。这可以让一个外部进程来检查named是否完全被停止。

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的进程号。这可以让一个外部进程来检查named是否完全被中断。

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。

notrace 将服务器的调试级别设置为0。

flush 刷新服务器的缓存。

flushname name 刷新来自服务器的DNS缓存以及，如果合适，以及来自服务器的名字服务器地址库或异常服务器缓存的给定名字。

flush tree name [view] 刷新来自服务器的DNS缓存，地址库或异常服务器缓存中的给定名字，及其所有子域。

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/JIN区。

recursing 转储named当前为其提供递归服务的请求列表。

validation (on | off | check) [view ...] 打开，关闭DNSSEC验证或检查DNSSEC验证的状态。注意dnssec-enable也需要设置成yes或auto才能生效。缺省打开的。

tsig-list 列出当前被配置由named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。

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。）

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中的区配置文本。

配置被保存在名为hash.nzf的文件中，其中hash是由视图的名字加密hash后所生成的。当named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。）

delzone [-clean] zone [class [view]] 在服务器运行时删除一个区。只有最初由rndc addzone所增加的区才能用这种方式删除。

如果指定了`-clean`，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。）

signing [(`-list` | `-clear keyid/algorithm` | `-clear all` | `-nsec3param (parameters | none)`)

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 hash的算法中。`salt`是一个表示成十六进制数的一串数据，一个连字符（`-`）表示不使用salt，或者关键字**auto**，它使**named**生成一个随机64位salt。

例如，要创建一个NSEC3链，使用SHA-1 hash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。

限制

rndc还不支持BIND 8的**ndc**工具中的所有命令。

当前没有在不使用配置文件的方式下提供共享密钥**key_id**的方式。

几个错误消息可以被清除。

参见

`rndc.conf(5)`, `rndc-confgen(8)`, `named(8)`, `named.conf(5)`, `ndc(8)`, *BIND 9*管理员参考手册。

作者

Internet Systems Consortium

E.21 *rndc.conf*

名字

`rndc.conf` — *rndc*的配置文件

概要

`rndc.conf`

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。

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      "6FMfj430sz4lyb240Ie2iGEz9lf11lJO+lz";
};

key testkey {
    algorithm hmac-sha256;
    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HMAC-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
```

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。

参见

rndc(8)，**rndc-confgen**(8)，**mmencode**(1)，*BIND 9*管理员参考手册。

作者

Internet Systems Consortium

E.22 rndc-confgen

名字

rndc-confgen — **rndc**密钥生成工具

概要

```
rndc-confgen [-a] [-A algorithm] [-b keysize] [-c keyfile] [-h] [-k
    keyname] [-p port] [-r randomfile] [-s address] [-t chrootdir] [-u
    user]
```

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。

选项

-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-md5。

-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。

-r randomfile 指定用于生成授权的随机数据的源。如果操作系统不提供/dev/random或等效的设备，缺省的随机性的源为键盘输入。**randomdev**指定一个字符设备名或包含随机数据的文件名，用来替代缺省值。特定值**keyboard**指示使用键盘输入。

-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。

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
```

参见

`rndc(8)`, `rndc.conf(5)`, `named(8)`, *BIND 9* 管理员参考手册。

作者

Internet Systems Consortium

E.23 ddns-confgen

名字

`ddns-confgen` — `ddns` 密钥生成工具

概要

```
tsig-keygen [-a algorithm] [-h] [-r randomfile] [name]
```

```
ddns-confgen [-a algorithm] [-h] [-k keyname] [-q] [-r randomfile] [-s name  
| -z zone]
```

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。

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（只对 `ddns-confgen`。）安静模式：只打印密钥，没有解释的文本或用法举例；这对 `tsig-keygen` 基本相同。

-r randomfile 指定生成授权的随机数据的源。如果操作系统不提供 `/dev/random` 或等效的设备，缺省的随机性的源为键盘输入。`randomdev` 指定一个字符设备名或包含随机数据的文件名，用来替代缺省值。特定值 `keyboard` 指示使用键盘输入。

-s name（只对 `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 `name` 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（只对 `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。

参见

`nsupdate(1)`, `named.conf(5)`, `named(8)`, *BIND 9* 管理员参考手册。

作者

Internet Systems Consortium

E.24 arpaname

名字

`arpaname` — 将IP地址翻译成对应的ARPA名字

概要

```
arpaname ipaddress...
```

描述

arpaname 将IP地址（IPv4和IPv6）翻译成对应的IN-ADDR.ARPA或IP6.ARPA名字。

参见

BIND 9 管理员参考手册。

作者

Internet Systems Consortium

E.25 genrandom

名字

genrandom — 生成一个包含随机数据的文件

概要

```
genrandom [-n number] size filename
```

描述

genrandom 生成一个或多个包含指定数量的伪随机数据的文件，在没有随机设备的系统上，这些数据可以用作其它命令的熵源。

参数

-n number 代替生成一个文件，生成number（从2到9）个文件，在文件名后面附加number。

size 要生成的文件大小，以千字节为单位。

filename 要写入随机数据的文件名。

参见

rand(3), arc4random(3)

作者

Internet Systems Consortium

E.26 isc-hmac-fixup

名字

isc-hmac-fixup — 修补旧版本的BIND所生成的HMAC密钥

概要

```
isc-hmac-fixup algorithm secret
```

描述

BIND 9的各个版本直到并包含BIND 9.6都包含一个缺陷，它导致HMAC-SHA* TSIG 密钥比hash算法（如，SHA1密钥超过160位，SHA256密钥超过256位等等）的摘要长度更长，因此不能正确使用，其生成一个消息认证码与其它DNS实现不兼容。

这个缺陷在BIND 9.7中被修补了。然而，这个修补可能导致在使用长的密钥时，旧的和新的BIND版本之间不兼容。**isc-hmac-fixup**修改了那些密钥以恢复兼容性。

要修改一个密钥，运行**isc-hmac-fixup**并在命令行指定密钥的算法和口令。如果口令比算法的摘要长度更长（对SHA1到SHA256为64字节，或者对SHA384和SHA512为128字节），将会生成一个新的口令，它由旧的口令的一个hash摘要所组成。（如果口令不要求转换，就不加修改地输出它。）

安全考虑

被**isc-hmac-fixup**所转换的口令被缩短了，但是这就是HMAC协议的工作方式，它没有影响安全。RFC 2104中提到，“比[摘要长度]更长的密钥时可接受的，但是额外的长度不应较大地增加函数的强度。”

参见

BIND 9管理员参考手册, RFC 2104.

作者

Internet Systems Consortium

E.27 nsec3hash

名字

nsec3hash — 生成NSEC3 hash值

概要

```
nsec3hash salt algorithm iterations domain
```

描述

nsec3hash基于一个NSEC3参数集生成一个NSEC3 hash值。这可以用于检查一个签名区中NSEC3记录的有效性。

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hash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。

iterations hash应该额外执行的次数。

domain 被hash的域名。

参见

BIND 9 管理员参考手册, RFC 5155.

作者

Internet Systems Consortium