```
**********************************************************
   29275 Thu Oct 11 15:45:32 2018
new/usr/src/lib/libresolv2/common/nameser/ns_print.c
9886 libresolv2: fix fallthrough in ns_sprintrrf()
**********************************************************
______unchanged_portion_omitted_

  89 /*%
  90  *      Convert the fields of an RR into presentation format.
  91  *
  92  * return:
  93  *\li   Number of characters written to buf, or -1 (check errno).
  94  */
  95 int
  96 ns_sprintrrf(const u_char *msg, size_t msglen,
  97              const char *name, ns_class class, ns_type type,
  98              u_long ttl, const u_char *rdata, size_t rdlen,
  99              const char *name_ctx, const char *origin,
 100              char *buf, size_t buflen)
 101 {
 102         const char *obuf = buf;
 103         const u_char *edata = rdata + rdlen;
 104         int spaced = 0;

 106         const char *comment;
 107         char tmp[100];
 108         int len, x;

 110         /*
 111          * Owner.
 112          */
 113         if (name_ctx != NULL && ns_samename(name_ctx, name) == 1) {
 114                 T(addstr("\t\t\t", 3, &buf, &buflen));
 115         } else {
 116                 len = prune_origin(name, origin);
 117                 if (*name == '\0') {
 118                         goto root;
 119                 } else if (len == 0) {
 120                         T(addstr("@\t\t\t", 4, &buf, &buflen));
 121                 } else {
 122                         T(addstr(name, len, &buf, &buflen));
 123                         /* Origin not used or not root, and no trailing dot? */
 124                         if (((origin == NULL || origin[0] == '\0') ||
 125                             (origin[0] != '.' && origin[1] != '\0' &&
 126                             name[len] == '\0')) && name[len - 1] != '.') {
 127  root:
 128                                 T(addstr(".", 1, &buf, &buflen));
 129                                 len++;
 130                         }
 131                         T(spaced = addtab(len, 24, spaced, &buf, &buflen));
 132                 }
 133         }

 135         /*
 136          * TTL, Class, Type.
 137          */
 138         T(x = ns_format_ttl(ttl, buf, buflen));
 139         addlen(x, &buf, &buflen);
 140         len = SPRINTF((tmp, " %s %s", p_class(class), p_type(type)));
 141         T(addstr(tmp, len, &buf, &buflen));
 142         T(spaced = addtab(x + len, 16, spaced, &buf, &buflen));

 144         /*
 145          * RData.
 146          */
 147         switch (type) {
```



```
 148         case ns_t_a:
 149                 if (rdlen != (size_t)NS_INADDRSZ)
 150                         goto formerr;
 151                 (void) inet_ntop(AF_INET, rdata, buf, buflen);
 152                 addlen(strlen(buf), &buf, &buflen);
 153                 break;

 155         case ns_t_cname:
 156         case ns_t_mb:
 157         case ns_t_mg:
 158         case ns_t_mr:
 159         case ns_t_ns:
 160         case ns_t_ptr:
 161         case ns_t_dname:
 162                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
 163                 break;

 165         case ns_t_hinfo:
 166         case ns_t_isdn:
 167                 /* First word. */
 168                 T(len = charstr(rdata, edata, &buf, &buflen));
 169                 if (len == 0)
 170                         goto formerr;
 171                 rdata += len;
 172                 T(addstr(" ", 1, &buf, &buflen));

 174
 175                 /* Second word, optional in ISDN records. */
 176                 if (type == ns_t_isdn && rdata == edata)
 177                         break;
 178
 179                 T(len = charstr(rdata, edata, &buf, &buflen));
 180                 if (len == 0)
 181                         goto formerr;
 182                 rdata += len;
 183                 break;

 185         case ns_t_soa: {
 186                 u_long t;

 188                 /* Server name. */
 189                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
 190                 T(addstr(" ", 1, &buf, &buflen));

 192                 /* Administrator name. */
 193                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
 194                 T(addstr(" (\n", 3, &buf, &buflen));
 195                 spaced = 0;

 197                 if ((edata - rdata) != 5*NS_INT32SZ)
 198                         goto formerr;

 200                 /* Serial number. */
 201                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
 202                 T(addstr("\t\t\t\t\t", 5, &buf, &buflen));
 203                 len = SPRINTF((tmp, "%lu", t));
 204                 T(addstr(tmp, len, &buf, &buflen));
 205                 T(spaced = addtab(len, 16, spaced, &buf, &buflen));
 206                 T(addstr("; serial\n", 9, &buf, &buflen));
 207                 spaced = 0;

 209                 /* Refresh interval. */
 210                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
 211                 T(addstr("\t\t\t\t\t", 5, &buf, &buflen));
 212                 T(len = ns_format_ttl(t, buf, buflen));
 213                 addlen(len, &buf, &buflen);
```

```
214                 T(spaced = addtab(len, 16, spaced, &buf, &buflen));
215                 T(addstr("; refresh\n", 10, &buf, &buflen));
216                 spaced = 0;

218                 /* Retry interval. */
219                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
220                 T(addstr("\t\t\t\t\t", 5, &buf, &buflen));
221                 T(len = ns_format_ttl(t, buf, buflen));
222                 addlen(len, &buf, &buflen);
223                 T(spaced = addtab(len, 16, spaced, &buf, &buflen));
224                 T(addstr("; retry\n", 8, &buf, &buflen));
225                 spaced = 0;

227                 /* Expiry. */
228                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
229                 T(addstr("\t\t\t\t\t", 5, &buf, &buflen));
230                 T(len = ns_format_ttl(t, buf, buflen));
231                 addlen(len, &buf, &buflen);
232                 T(spaced = addtab(len, 16, spaced, &buf, &buflen));
233                 T(addstr("; expiry\n", 9, &buf, &buflen));
234                 spaced = 0;

236                 /* Minimum TTL. */
237                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
238                 T(addstr("\t\t\t\t\t", 5, &buf, &buflen));
239                 T(len = ns_format_ttl(t, buf, buflen));
240                 addlen(len, &buf, &buflen);
241                 T(addstr(" )", 2, &buf, &buflen));
242                 T(spaced = addtab(len, 16, spaced, &buf, &buflen));
243                 T(addstr("; minimum\n", 10, &buf, &buflen));

245                 break;
246             }

248         case ns_t_mx:
249         case ns_t_afsdb:
250         case ns_t_rt:
251         case ns_t_kx: {
252                 u_int t;

254                 if (rdlen < (size_t)NS_INT16SZ)
255                         goto formerr;

257                 /* Priority. */
258                 t = ns_get16(rdata);
259                 rdata += NS_INT16SZ;
260                 len = SPRINTF((tmp, "%u ", t));
261                 T(addstr(tmp, len, &buf, &buflen));

263                 /* Target. */
264                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));

266                 break;
267             }

269         case ns_t_px: {
270                 u_int t;

272                 if (rdlen < (size_t)NS_INT16SZ)
273                         goto formerr;

275                 /* Priority. */
276                 t = ns_get16(rdata);
277                 rdata += NS_INT16SZ;
278                 len = SPRINTF((tmp, "%u ", t));
279                 T(addstr(tmp, len, &buf, &buflen));
```



```
281                 /* Name1. */
282                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
283                 T(addstr(" ", 1, &buf, &buflen));

285                 /* Name2. */
286                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));

288                 break;
289             }

291         case ns_t_x25:
292                 T(len = charstr(rdata, edata, &buf, &buflen));
293                 if (len == 0)
294                         goto formerr;
295                 rdata += len;
296                 break;

298         case ns_t_txt:
299         case ns_t_spf:
300                 while (rdata < edata) {
301                         T(len = charstr(rdata, edata, &buf, &buflen));
302                         if (len == 0)
303                                 goto formerr;
304                         rdata += len;
305                         if (rdata < edata)
306                                 T(addstr(" ", 1, &buf, &buflen));
307                 }
308                 break;

310         case ns_t_nsap: {
311                 char t[2+255*3];

313                 (void) inet_nsap_ntoa(rdlen, rdata, t);
314                 T(addstr(t, strlen(t), &buf, &buflen));
315                 break;
316             }

318         case ns_t_aaaa:
319                 if (rdlen != (size_t)NS_IN6ADDRSZ)
320                         goto formerr;
321                 (void) inet_ntop(AF_INET6, rdata, buf, buflen);
322                 addlen(strlen(buf), &buf, &buflen);
323                 break;

325         case ns_t_loc: {
326                 char t[255];

328                 /* XXX protocol format checking? */
329                 (void) loc_ntoa(rdata, t);
330                 T(addstr(t, strlen(t), &buf, &buflen));
331                 break;
332             }

334         case ns_t_naptr: {
335                 u_int order, preference;
336                 char t[50];

338                 if (rdlen < 2U*NS_INT16SZ)
339                         goto formerr;

341                 /* Order, Precedence. */
342                 order = ns_get16(rdata);        rdata += NS_INT16SZ;
343                 preference = ns_get16(rdata);   rdata += NS_INT16SZ;
344                 len = SPRINTF((t, "%u %u ", order, preference));
345                 T(addstr(t, len, &buf, &buflen));
```

```
347                 /* Flags. */
348                 T(len = charstr(rdata, edata, &buf, &buflen));
349                 if (len == 0)
350                         goto formerr;
351                 rdata += len;
352                 T(addstr(" ", 1, &buf, &buflen));
353
354                 /* Service. */
355                 T(len = charstr(rdata, edata, &buf, &buflen));
356                 if (len == 0)
357                         goto formerr;
358                 rdata += len;
359                 T(addstr(" ", 1, &buf, &buflen));
360
361                 /* Regexp. */
362                 T(len = charstr(rdata, edata, &buf, &buflen));
363                 if (len < 0)
364                         return (-1);
365                 if (len == 0)
366                         goto formerr;
367                 rdata += len;
368                 T(addstr(" ", 1, &buf, &buflen));
369
370                 /* Server. */
371                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
372                 break;
373             }
374
375         case ns_t_srv: {
376                 u_int priority, weight, port;
377                 char t[50];
378
379                 if (rdlen < 3U*NS_INT16SZ)
380                         goto formerr;
381
382                 /* Priority, Weight, Port. */
383                 priority = ns_get16(rdata);  rdata += NS_INT16SZ;
384                 weight   = ns_get16(rdata);  rdata += NS_INT16SZ;
385                 port     = ns_get16(rdata);  rdata += NS_INT16SZ;
386                 len = SPRINTF((t, "%u %u %u ", priority, weight, port));
387                 T(addstr(t, len, &buf, &buflen));
388
389                 /* Server. */
390                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
391                 break;
392             }
393
394         case ns_t_minfo:
395         case ns_t_rp:
396                 /* Name1. */
397                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
398                 T(addstr(" ", 1, &buf, &buflen));
399
400                 /* Name2. */
401                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
402
403                 break;
404
405         case ns_t_wks: {
406                 int n, lcnt;
407
408                 if (rdlen < 1U + NS_INT32SZ)
409                         goto formerr;
410
411                 /* Address. */
```



```
412                 (void) inet_ntop(AF_INET, rdata, buf, buflen);
413                 addlen(strlen(buf), &buf, &buflen);
414                 rdata += NS_INADDRSZ;
415
416                 /* Protocol. */
417                 len = SPRINTF((tmp, " %u ( ", *rdata));
418                 T(addstr(tmp, len, &buf, &buflen));
419                 rdata += NS_INT8SZ;
420
421                 /* Bit map. */
422                 n = 0;
423                 lcnt = 0;
424                 while (rdata < edata) {
425                         u_int c = *rdata++;
426                         do {
427                                 if (c & 0200) {
428                                         if (lcnt == 0) {
429                                                 T(addstr("\n\t\t\t\t", 5,
430                                                          &buf, &buflen));
431                                                 lcnt = 10;
432                                                 spaced = 0;
433                                         }
434                                         len = SPRINTF((tmp, "%d ", n));
435                                         T(addstr(tmp, len, &buf, &buflen));
436                                         lcnt--;
437                                 }
438                                 c <<= 1;
439                         } while (++n & 07);
440                 }
441                 T(addstr(")", 1, &buf, &buflen));
442
443                 break;
444             }
445
446         case ns_t_key:
447         case ns_t_dnskey: {
448                 char base64_key[NS_MD5RSA_MAX_BASE64];
449                 u_int keyflags, protocol, algorithm, key_id;
450                 const char *leader;
451                 int n;
452
453                 if (rdlen < 0U + NS_INT16SZ + NS_INT8SZ + NS_INT8SZ)
454                         goto formerr;
455
456                 /* Key flags, Protocol, Algorithm. */
457                 key_id = dst_s_dns_key_id(rdata, edata-rdata);
458                 keyflags = ns_get16(rdata);  rdata += NS_INT16SZ;
459                 protocol = *rdata++;
460                 algorithm = *rdata++;
461                 len = SPRINTF((tmp, "0x%04x %u %u",
462                                keyflags, protocol, algorithm));
463                 T(addstr(tmp, len, &buf, &buflen));
464
465                 /* Public key data. */
466                 len = b64_ntop(rdata, edata - rdata,
467                                base64_key, sizeof base64_key);
468                 if (len < 0)
469                         goto formerr;
470                 if (len > 15) {
471                         T(addstr(" (", 2, &buf, &buflen));
472                         leader = "\n\t\t";
473                         spaced = 0;
474                 } else
475                         leader = " ";
476                 for (n = 0; n < len; n += 48) {
477                         T(addstr(leader, strlen(leader), &buf, &buflen));
```

```
478                         T(addstr(base64_key + n, MIN(len - n, 48),
479                                 &buf, &buflen));
480                 }
481                 if (len > 15)
482                         T(addstr(" )", 2, &buf, &buflen));
483                 n = SPRINTF((tmp, " ; key_tag= %u", key_id));
484                 T(addstr(tmp, n, &buf, &buflen));
485
486                 break;
487             }
488
489         case ns_t_sig:
490         case ns_t_rrsig: {
491                 char base64_key[NS_MD5RSA_MAX_BASE64];
492                 u_int type, algorithm, labels, footprint;
493                 const char *leader;
494                 u_long t;
495                 int n;
496
497                 if (rdlen < 22U)
498                         goto formerr;
499
500                 /* Type covered, Algorithm, Label count, Original TTL. */
501                 type = ns_get16(rdata);  rdata += NS_INT16SZ;
502                 algorithm = *rdata++;
503                 labels = *rdata++;
504                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
505                 len = SPRINTF((tmp, "%s %d %d %lu ",
506                                p_type(type), algorithm, labels, t));
507                 T(addstr(tmp, len, &buf, &buflen));
508                 if (labels > (u_int)dn_count_labels(name))
509                         goto formerr;
510
511                 /* Signature expiry. */
512                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
513                 len = SPRINTF((tmp, "%s ", p_secstodate(t)));
514                 T(addstr(tmp, len, &buf, &buflen));
515
516                 /* Time signed. */
517                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
518                 len = SPRINTF((tmp, "%s ", p_secstodate(t)));
519                 T(addstr(tmp, len, &buf, &buflen));
520
521                 /* Signature Footprint. */
522                 footprint = ns_get16(rdata);  rdata += NS_INT16SZ;
523                 len = SPRINTF((tmp, "%u ", footprint));
524                 T(addstr(tmp, len, &buf, &buflen));
525
526                 /* Signer's name. */
527                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
528
529                 /* Signature. */
530                 len = b64_ntop(rdata, edata - rdata,
531                                base64_key, sizeof base64_key);
532                 if (len > 15) {
533                         T(addstr(" (", 2, &buf, &buflen));
534                         leader = "\n\t\t";
535                         spaced = 0;
536                 } else
537                         leader = " ";
538                 if (len < 0)
539                         goto formerr;
540                 for (n = 0; n < len; n += 48) {
541                         T(addstr(leader, strlen(leader), &buf, &buflen));
542                         T(addstr(base64_key + n, MIN(len - n, 48),
543                                 &buf, &buflen));
```



```
544                 }
545                 if (len > 15)
546                         T(addstr(" )", 2, &buf, &buflen));
547                 break;
548             }
549
550         case ns_t_nxt: {
551                 int n, c;
552
553                 /* Next domain name. */
554                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
555
556                 /* Type bit map. */
557                 n = edata - rdata;
558                 for (c = 0; c < n*8; c++)
559                         if (NS_NXT_BIT_ISSET(c, rdata)) {
560                                 len = SPRINTF((tmp, " %s", p_type(c)));
561                                 T(addstr(tmp, len, &buf, &buflen));
562                         }
563                 break;
564             }
565
566         case ns_t_cert: {
567                 u_int c_type, key_tag, alg;
568                 int n;
569                 unsigned int siz;
570                 char base64_cert[8192], tmp[40];
571                 const char *leader;
572
573                 c_type  = ns_get16(rdata); rdata += NS_INT16SZ;
574                 key_tag = ns_get16(rdata); rdata += NS_INT16SZ;
575                 alg = (u_int) *rdata++;
576
577                 len = SPRINTF((tmp, "%d %d %d ", c_type, key_tag, alg));
578                 T(addstr(tmp, len, &buf, &buflen));
579                 siz = (edata-rdata)*4/3 + 4; /* "+4" accounts for trailing \0 */
580                 if (siz > sizeof(base64_cert) * 3/4) {
581                         const char *str = "record too long to print";
582                         T(addstr(str, strlen(str), &buf, &buflen));
583                 }
584                 else {
585                         len = b64_ntop(rdata, edata-rdata, base64_cert, siz);
586
587                         if (len < 0)
588                                 goto formerr;
589                         else if (len > 15) {
590                                 T(addstr(" (", 2, &buf, &buflen));
591                                 leader = "\n\t\t";
592                                 spaced = 0;
593                         }
594                         else
595                                 leader = " ";
596
597                         for (n = 0; n < len; n += 48) {
598                                 T(addstr(leader, strlen(leader),
599                                         &buf, &buflen));
600                                 T(addstr(base64_cert + n, MIN(len - n, 48),
601                                         &buf, &buflen));
602                         }
603                         if (len > 15)
604                                 T(addstr(" )", 2, &buf, &buflen));
605                 }
606                 break;
607             }
608
609         case ns_t_tkey: {
```

```
610                 /* KJD - need to complete this */
611                 u_long t;
612                 int mode, err, keysize;

614                 /* Algorithm name. */
615                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
616                 T(addstr(" ", 1, &buf, &buflen));

618                 /* Inception. */
619                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
620                 len = SPRINTF((tmp, "%s ", p_secstodate(t)));
621                 T(addstr(tmp, len, &buf, &buflen));

623                 /* Experation. */
624                 t = ns_get32(rdata);  rdata += NS_INT32SZ;
625                 len = SPRINTF((tmp, "%s ", p_secstodate(t)));
626                 T(addstr(tmp, len, &buf, &buflen));

628                 /* Mode , Error, Key Size. */
629                 /* Priority, Weight, Port. */
630                 mode = ns_get16(rdata);  rdata += NS_INT16SZ;
631                 err  = ns_get16(rdata);  rdata += NS_INT16SZ;
632                 keysize  = ns_get16(rdata);  rdata += NS_INT16SZ;
633                 len = SPRINTF((tmp, "%u %u %u ", mode, err, keysize));
634                 T(addstr(tmp, len, &buf, &buflen));

636                 /* XXX need to dump key, print otherdata length & other data */
637                 break;
638             }

640         case ns_t_tsig: {
641                 /* BEW - need to complete this */
642                 int n;

644                 T(len = addname(msg, msglen, &rdata, origin, &buf, &buflen));
645                 T(addstr(" ", 1, &buf, &buflen));
646                 rdata += 8; /*%< time */
647                 n = ns_get16(rdata); rdata += INT16SZ;
648                 rdata += n; /*%< sig */
649                 n = ns_get16(rdata); rdata += INT16SZ; /*%< original id */
650                 sprintf(buf, "%d", ns_get16(rdata));
651                 rdata += INT16SZ;
652                 addlen(strlen(buf), &buf, &buflen);
653                 break;
654             }

656         case ns_t_a6: {
657                 struct in6_addr a;
658                 int pbyte, pbit;

660                 /* prefix length */
661                 if (rdlen == 0U) goto formerr;
662                 len = SPRINTF((tmp, "%d ", *rdata));
663                 T(addstr(tmp, len, &buf, &buflen));
664                 pbit = *rdata;
665                 if (pbit > 128) goto formerr;
666                 pbyte = (pbit & ~7) / 8;
667                 rdata++;

669                 /* address suffix: provided only when prefix len != 128 */
670                 if (pbit < 128) {
671                         if (rdata + pbyte >= edata) goto formerr;
672                         memset(&a, 0, sizeof(a));
673                         memcpy(&a.s6_addr[pbyte], rdata, sizeof(a) - pbyte);
674                         (void) inet_ntop(AF_INET6, &a, buf, buflen);
675                         addlen(strlen(buf), &buf, &buflen);
```



```
676                         rdata += sizeof(a) - pbyte;
677                 }

679                 /* prefix name: provided only when prefix len > 0 */
680                 if (pbit == 0)
681                         break;
682                 if (rdata >= edata) goto formerr;
683                 T(addstr(" ", 1, &buf, &buflen));
684                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
685
686                 break;
687             }

689         case ns_t_opt: {
690                 len = SPRINTF((tmp, "%u bytes", class));
691                 T(addstr(tmp, len, &buf, &buflen));
692                 break;
693             }

695         case ns_t_ds:
696         case ns_t_dlv:
697         case ns_t_sshfp: {
698                 u_int t;

700                 if (type == ns_t_ds || type == ns_t_dlv) {
701                         if (rdlen < 4U) goto formerr;
702                         t = ns_get16(rdata);
703                         rdata += NS_INT16SZ;
704                         len = SPRINTF((tmp, "%u ", t));
705                         T(addstr(tmp, len, &buf, &buflen));
706                 } else
707                         if (rdlen < 2U) goto formerr;

709                 len = SPRINTF((tmp, "%u ", *rdata));
710                 T(addstr(tmp, len, &buf, &buflen));
711                 rdata++;

713                 len = SPRINTF((tmp, "%u ", *rdata));
714                 T(addstr(tmp, len, &buf, &buflen));
715                 rdata++;

717                 while (rdata < edata) {
718                         len = SPRINTF((tmp, "%02X", *rdata));
719                         T(addstr(tmp, len, &buf, &buflen));
720                         rdata++;
721                 }
722                 break;
723             }

725         case ns_t_nsec3:
726         case ns_t_nsec3param: {
727                 u_int t, w, l, j, k, c;
728
729                 len = SPRINTF((tmp, "%u ", *rdata));
730                 T(addstr(tmp, len, &buf, &buflen));
731                 rdata++;

733                 len = SPRINTF((tmp, "%u ", *rdata));
734                 T(addstr(tmp, len, &buf, &buflen));
735                 rdata++;

737                 t = ns_get16(rdata);
738                 rdata += NS_INT16SZ;
739                 len = SPRINTF((tmp, "%u ", t));
740                 T(addstr(tmp, len, &buf, &buflen));
```

```
742                 t = *rdata++;
743                 if (t == 0) {
744                         T(addstr("-", 1, &buf, &buflen));
745                 } else {
746                         while (t-- > 0) {
747                                 len = SPRINTF((tmp, "%02X", *rdata));
748                                 T(addstr(tmp, len, &buf, &buflen));
749                                 rdata++;
750                         }
751                 }
752                 if (type == ns_t_nsec3param)
753                         break;
754                 T(addstr(" ", 1, &buf, &buflen));

756                 t = *rdata++;
757                 while (t > 0) {
758                         switch (t) {
759                         case 1:
760                                 tmp[0] = base32hex[((rdata[0]>>3)&0x1f)];
761                                 tmp[1] = base32hex[((rdata[0]<<2)&0x1c)];
762                                 tmp[2] = tmp[3] = tmp[4] = '=';
763                                 tmp[5] = tmp[6] = tmp[7] = '=';
764                                 break;
765                         case 2:
766                                 tmp[0] = base32hex[((rdata[0]>>3)&0x1f)];
767                                 tmp[1] = base32hex[((rdata[0]<<2)&0x1c)|
768                                                    ((rdata[1]>>6)&0x03)];
769                                 tmp[2] = base32hex[((rdata[1]>>1)&0x1f)];
770                                 tmp[3] = base32hex[((rdata[1]<<4)&0x10)];
771                                 tmp[4] = tmp[5] = tmp[6] = tmp[7] = '=';
772                                 break;
773                         case 3:
774                                 tmp[0] = base32hex[((rdata[0]>>3)&0x1f)];
775                                 tmp[1] = base32hex[((rdata[0]<<2)&0x1c)|
776                                                    ((rdata[1]>>6)&0x03)];
777                                 tmp[2] = base32hex[((rdata[1]>>1)&0x1f)];
778                                 tmp[3] = base32hex[((rdata[1]<<4)&0x10)|
779                                                    ((rdata[2]>>4)&0x0f)];
780                                 tmp[4] = base32hex[((rdata[2]<<1)&0x1e)];
781                                 tmp[5] = tmp[6] = tmp[7] = '=';
782                                 break;
783                         case 4:
784                                 tmp[0] = base32hex[((rdata[0]>>3)&0x1f)];
785                                 tmp[1] = base32hex[((rdata[0]<<2)&0x1c)|
786                                                    ((rdata[1]>>6)&0x03)];
787                                 tmp[2] = base32hex[((rdata[1]>>1)&0x1f)];
788                                 tmp[3] = base32hex[((rdata[1]<<4)&0x10)|
789                                                    ((rdata[2]>>4)&0x0f)];
790                                 tmp[4] = base32hex[((rdata[2]<<1)&0x1e)|
791                                                    ((rdata[3]>>7)&0x01)];
792                                 tmp[5] = base32hex[((rdata[3]>>2)&0x1f)];
793                                 tmp[6] = base32hex[(rdata[3]<<3)&0x18];
794                                 tmp[7] = '=';
795                                 break;
796                         default:
797                                 tmp[0] = base32hex[((rdata[0]>>3)&0x1f)];
798                                 tmp[1] = base32hex[((rdata[0]<<2)&0x1c)|
799                                                    ((rdata[1]>>6)&0x03)];
800                                 tmp[2] = base32hex[((rdata[1]>>1)&0x1f)];
801                                 tmp[3] = base32hex[((rdata[1]<<4)&0x10)|
802                                                    ((rdata[2]>>4)&0x0f)];
803                                 tmp[4] = base32hex[((rdata[2]<<1)&0x1e)|
804                                                    ((rdata[3]>>7)&0x01)];
805                                 tmp[5] = base32hex[((rdata[3]>>2)&0x1f)];
806                                 tmp[6] = base32hex[((rdata[3]<<3)&0x18)|
807                                                    ((rdata[4]>>5)&0x07)];
```



```
808                                 tmp[7] = base32hex[(rdata[4]&0x1f)];
809                                 break;
810                         }
811                         T(addstr(tmp, 8, &buf, &buflen));
812                         if (t >= 5) {
813                                 rdata += 5;
814                                 t -= 5;
815                         } else {
816                                 rdata += t;
817                                 t -= t;
818                         }
819                 }

821                 while (rdata < edata) {
822                         w = *rdata++;
823                         l = *rdata++;
824                         for (j = 0; j < l; j++) {
825                                 if (rdata[j] == 0)
826                                         continue;
827                                 for (k = 0; k < 8; k++) {
828                                         if ((rdata[j] & (0x80 >> k)) == 0)
829                                                 continue;
830                                         c = w * 256 + j * 8 + k;
831                                         len = SPRINTF((tmp, " %s", p_type(c)));
832                                         T(addstr(tmp, len, &buf, &buflen));
833                                 }
834                         }
835                         rdata += l;
836                 }
837                 break;
838             }

840         case ns_t_nsec: {
841                 u_int w, l, j, k, c;

843                 T(addname(msg, msglen, &rdata, origin, &buf, &buflen));

845                 while (rdata < edata) {
846                         w = *rdata++;
847                         l = *rdata++;
848                         for (j = 0; j < l; j++) {
849                                 if (rdata[j] == 0)
850                                         continue;
851                                 for (k = 0; k < 8; k++) {
852                                         if ((rdata[j] & (0x80 >> k)) == 0)
853                                                 continue;
854                                         c = w * 256 + j * 8 + k;
855                                         len = SPRINTF((tmp, " %s", p_type(c)));
856                                         T(addstr(tmp, len, &buf, &buflen));
857                                 }
858                         }
859                         rdata += l;
860                 }
861                 break;
862             }

864         case ns_t_dhcid: {
865                 int n;
866                 unsigned int siz;
867                 char base64_dhcid[8192];
868                 const char *leader;

870                 siz = (edata-rdata)*4/3 + 4; /* "+4" accounts for trailing \0 */
871                 if (siz > sizeof(base64_dhcid) * 3/4) {
872                         const char *str = "record too long to print";
873                         T(addstr(str, strlen(str), &buf, &buflen));
```

```
874                 } else {
875                         len = b64_ntop(rdata, edata-rdata, base64_dhcid, siz);
876
877                         if (len < 0)
878                                 goto formerr;
879
880                         else if (len > 15) {
881                                 T(addstr(" (", 2, &buf, &buflen));
882                                 leader = "\n\t\t";
883                                 spaced = 0;
884                         }
885                         else
886                                 leader = " ";
887
888                         for (n = 0; n < len; n += 48) {
889                                 T(addstr(leader, strlen(leader),
890                                         &buf, &buflen));
891                                 T(addstr(base64_dhcid + n, MIN(len - n, 48),
892                                         &buf, &buflen));
893                         }
894                         if (len > 15)
895                                 T(addstr(" )", 2, &buf, &buflen));
896                 }
897                 break;
898         }
898         /* FALLTHROUGH */
899
900         case ns_t_ipseckey: {
901                 int n;
902                 unsigned int siz;
903                 char base64_key[8192];
904                 const char *leader;
905
906                 if (rdlen < 2)
907                         goto formerr;
908
909                 switch (rdata[1]) {
910                 case 0:
911                 case 3:
912                         if (rdlen < 3)
913                                 goto formerr;
914                         break;
915                 case 1:
916                         if (rdlen < 7)
917                                 goto formerr;
918                         break;
919                 case 2:
920                         if (rdlen < 19)
921                                 goto formerr;
922                         break;
923                 default:
924                         comment = "unknown IPSECKEY gateway type";
925                         goto hexify;
926                 }
927
928                 len = SPRINTF((tmp, "%u ", *rdata));
929                 T(addstr(tmp, len, &buf, &buflen));
930                 rdata++;
931
932                 len = SPRINTF((tmp, "%u ", *rdata));
933                 T(addstr(tmp, len, &buf, &buflen));
934                 rdata++;
935
936                 len = SPRINTF((tmp, "%u ", *rdata));
937                 T(addstr(tmp, len, &buf, &buflen));
938                 rdata++;
```



```
940                 switch (rdata[-2]) {
941                 case 0:
942                         T(addstr(".", 1, &buf, &buflen));
943                         break;
944                 case 1:
945                         (void) inet_ntop(AF_INET, rdata, buf, buflen);
946                         addlen(strlen(buf), &buf, &buflen);
947                         rdata += 4;
948                         break;
949                 case 2:
950                         (void) inet_ntop(AF_INET6, rdata, buf, buflen);
951                         addlen(strlen(buf), &buf, &buflen);
952                         rdata += 16;
953                         break;
954                 case 3:
955                         T(addname(msg, msglen, &rdata, origin, &buf, &buflen));
956                         break;
957                 }
958
959                 if (rdata >= edata)
960                         break;
961
962                 siz = (edata-rdata)*4/3 + 4; /* "+4" accounts for trailing \0 */
963                 if (siz > sizeof(base64_key) * 3/4) {
964                         const char *str = "record too long to print";
965                         T(addstr(str, strlen(str), &buf, &buflen));
966                 } else {
967                         len = b64_ntop(rdata, edata-rdata, base64_key, siz);
968
969                         if (len < 0)
970                                 goto formerr;
971
972                         else if (len > 15) {
973                                 T(addstr(" (", 2, &buf, &buflen));
974                                 leader = "\n\t\t";
975                                 spaced = 0;
976                         }
977                         else
978                                 leader = " ";
979
980                         for (n = 0; n < len; n += 48) {
981                                 T(addstr(leader, strlen(leader),
982                                         &buf, &buflen));
983                                 T(addstr(base64_key + n, MIN(len - n, 48),
984                                         &buf, &buflen));
985                         }
986                         if (len > 15)
987                                 T(addstr(" )", 2, &buf, &buflen));
988                 }
989                 break;
990         }
990         /* FALLTHROUGH */
991
992         case ns_t_hip: {
993                 unsigned int i, hip_len, algorithm, key_len;
994                 char base64_key[NS_MD5RSA_MAX_BASE64];
995                 unsigned int siz;
996                 const char *leader = "\n\t\t\t\t\t";
997
998                 hip_len = *rdata++;
999                 algorithm = *rdata++;
1000                key_len = ns_get16(rdata);
1001                rdata += NS_INT16SZ;
1002
1003                siz = key_len*4/3 + 4; /* "+4" accounts for trailing \0 */
```

```
1004                if (siz > sizeof(base64_key) * 3/4) {
1005                        const char *str = "record too long to print";
1006                        T(addstr(str, strlen(str), &buf, &buflen));
1007                } else {
1008                        len = sprintf(tmp, "( %u ", algorithm);
1009                        T(addstr(tmp, len, &buf, &buflen));

1011                        for (i = 0; i < hip_len; i++) {
1012                                len = sprintf(tmp, "%02X", *rdata);
1013                                T(addstr(tmp, len, &buf, &buflen));
1014                                rdata++;
1015                        }
1016                        T(addstr(leader, strlen(leader), &buf, &buflen));

1018                        len = b64_ntop(rdata, key_len, base64_key, siz);
1019                        if (len < 0)
1020                                goto formerr;

1022                        T(addstr(base64_key, len, &buf, &buflen));
1023
1024                        rdata += key_len;
1025                        while (rdata < edata) {
1026                                T(addstr(leader, strlen(leader), &buf, &buflen))
1027                                T(addname(msg, msglen, &rdata, origin,
1028                                          &buf, &buflen));
1029                        }
1030                        T(addstr(" )", 2, &buf, &buflen));
1031                }
1032                break;
1033        }

1035        default:
1036                comment = "unknown RR type";
1037                goto hexify;
1038        }
1039        return (buf - obuf);
1040 formerr:
1041        comment = "RR format error";
1042 hexify: {
1043        int n, m;
1044        char *p;

1046        len = SPRINTF((tmp, "\\# %u%s\t; %s", (unsigned)(edata - rdata),
1047                       rdlen != 0U ? " (" : "", comment));
1048        T(addstr(tmp, len, &buf, &buflen));
1049        while (rdata < edata) {
1050                p = tmp;
1051                p += SPRINTF((p, "\n\t"));
1052                spaced = 0;
1053                n = MIN(16, edata - rdata);
1054                for (m = 0; m < n; m++)
1055                        p += SPRINTF((p, "%02x ", rdata[m]));
1056                T(addstr(tmp, p - tmp, &buf, &buflen));
1057                if (n < 16) {
1058                        T(addstr(")", 1, &buf, &buflen));
1059                        T(addtab(p - tmp + 1, 48, spaced, &buf, &buflen));
1060                }
1061                p = tmp;
1062                p += SPRINTF((p, "; "));
1063                for (m = 0; m < n; m++)
1064                        *p++ = (isascii(rdata[m]) && isprint(rdata[m]))
1065                                ? rdata[m]
1066                                : '.';
1067                T(addstr(tmp, p - tmp, &buf, &buflen));
1068                rdata += n;
1069        }
```



```
1070        return (buf - obuf);
1071    }
1072 }
```

_____***unchanged_portion_omitted_***