پروتکل مدیریت ساده شبکه (SNMP)

۱. وضعیت این یادداشت

این RFC در واقع نسخه مجدد RFC 1098 به همراه بخش تغییر یافته "وضعیت این یادداشت"، به علاوه برخی اصلاحات نوشتاری است. این یادداشت پروتکل ساده­ای را تعریف می­کند که توسط آن، اطلاعات مدیریتی مربوط به یک عنصر شبکه، می­تواند توسط کاربران، به صورت منطقی و یا از راه دور مورد بازرسی یا تغییر قرار بگیرد. در واقع این یادداشت و یادداشت­های همراه آن، که ساختار مدیریت اطلاعات را در کنار پایگاه مدیریت اطلاعات توصیف می­کنند، این مستندات، ساختاری ساده، کارآمد و سیستمی را برای مدیریت شبکه های مبتنی بر TCP/IP و به طور خاص اینترنت، فراهم می­کند.

انجمن فعالیت های اینترنتی توصیه می­کند که تمامی پیاده سازی های IP و TCP به صورت شبکه­ای قابل مدیریت باشند. این موضوع پیاده سازی MIB Internet (RFC-1156) و حداقل یکی از دو پروتکل مدیریتی پیشنهادی یعنی SNMP (RFC-1157) یا CMOT (RFC-1095) را به دنبال دارد. بایستی توجه داشته باشید که در این زمان SNMP یک استاندارد کامل اینترنتی است و CMOT تنها پیش نویسی از یک استاندارد است. همچنین RFC های مورد نیاز Host (میزبان) و Gateway (درگاه) را برای اطلاعات بیشتری درباره کاربرد این استاندارد ببینید.

۲. مقدمه

مطابق با گزارش داده شده در RFC 1052، توصیه های IAB برای توسعه استاندارد­های مدیریت شبکه اینترنت [1]، یک استراتژی دو شاخه­ای برای مدیریت شبکه میان شبکه­های مبتنی بر TCP/IP اتخاذ شده است. در کوتاه مدت پروتکل مدیریت ساده شبکه[[1]](#footnote-1) (SNMP) برای مدیریت نود­ها در اینترنت مورد استفاده قرار گرفت و در بلند مدت استفاده از ساختار مدیریت شبکه OSI مورد بررسی قرار گرفت. سپس دو مستند برای تعریف مدیریت اطلاعات تولید شد: RFC 1065 که ساختار مدیریت اطلاعات[[2]](#footnote-2) (SMI) [2] و RFC 1066 که پایگاه مدیریت اطلاعات[[3]](#footnote-3) (MIB) را تعریف کرد [3]. هر دو این مستندات برای سازگاری با SNMP و ساختار مدیریت شبکه OSI طراحی شدند.

این استراتژی در کوتاه مدت تقریبا موفق بود: فناوری مدیریت شبکه بر مبنای اینترنت از طرف انجمن های پژوهشی و تجاری در چندین ماه مورد بررسی قرار گرفت. در نتیجه این کار، بخش هایی از جامعه اینترنت به صورت شبکه ای قابل مدیریت شدند.

مطابق با گزارش آورده شده در RFC 1109، گزارش دومین کارگروه بازبینی مدیریت شبکه اد هاک [4]، نیازمندی های SNMP و ساختارهای مدیریت شبکه OSI از آنچه که پیش بینی می­گردید متفاوت­تر بودند. به همین صورت نیازمندی برای سازگاری میان SMI/MIB و هر دو ساختار مورد تعلیق قرار گرفت. این اقدام به ساختار مدیریت کاربردی شبکه یعنی SNMP اجازه داد تا به نیازهای عملیاتی جدید در جامعه اینترنتی با تولید مستندات تعریف کننده آیتم های جدید MIB، پاسخ دهد.

IAB پروتکل های SNMP، SMI و Internet MIB اولیه را برای "پروتکل های استاندارد" کامل مطابق با وضعیت "توصیه شده" تعریف کرده است. با این کار IAB توصیه می کند که تمامی پیاده سازی های IP و TCP به صورت شبکه ای، قابل مدیریت باشند و انتظار می­رود که آن پیاده سازی­هایی که به صورت شبکه­ای قابل مدیریت هستند مورد پذیرش قرار گیرند که SMI، MIB و SNMP را پیاده سازی می­کنند.

به همین صورت ساختار فعلی مدیریت شبکه برای اینترنت­های مبتنی بر TCP/IP از موارد زیر تشکیل می­شود:

ساختار و شناسایی مدیریت شبکه برای اینترنت­های مبتنی بر TCP/IP ، که نحوه مدیریت اشیا (آبجکت ها) در MIB را، به عنوان یک گام رو به جلو در RFC 1155، توصیف می­کند [5]؛ پایگاه مدیریت اطلاعات برای مدیریت شبکه اینترنت­های مبتنی بر TCP/IP که اشیای مدیریت شده موجود در MIB را به عنوان یک گام رو به جلو در RFC 1156 توصیف می­کند [6] و پروتکل مدیریت ساده شبکه که پروتکل مورد استفاده برای مدیریت این اشیا را به عنوان گام آغازین، در این یادداشت تعریف می­کند.

مطابق با گزارش آورده شده در RFC 1052، توصیه های IAB برای توسعه استاندارد های مدیریت شبکه اینترنت [1]، انجمن فعالیت های اینترنتی کارگروه مهندسی اینترنت[[4]](#footnote-4) (IETF) را برای ایجاد دو کارگروه در حوزه مدیریت شبکه هدایت کرده است. یک گروه مسئولیت مشخصه بیشتر و تعریف عناصر برای در نظر گرفتن در پایگاه مدیریت اطلاعات (MIB) را بر عهده داشت. مسئولیت گروه دیگر تعریف اصلاحات برای پروتکل مدیریت ساده شبکه (SNMP) برای رفع نیازهای کوتاه مدت سازنده شبکه و هماهنگ سازی خروجی کارگروه MIB است.

کارگروه MIB دو یادداشت تولید کرد که یکی ساختار مدیریت اطلاعات (SMI) [2] برای استفاده توسط اشیاء مدیریت شده در MIB را تعریف می­کند. یادداشت دوم [3] فهرست اشیای مدیریت شده را تعریف می­کند.

خروجی کارگروه توسعه های SNMP، این یادداشت است، که تغییرات لازم برای دستیابی به سازگاری با خروجی کارگروه MIB را با تعریف اولیه SNMP ترکیب می­کند [7]. این تغییرات می بایست به منظور تطابق با بخشنامه IAB حداقل باشد، به طوری که کارگروه ها می بایست به شدت برای حفظ سادگی SNMP حساسیت به خرج دهند. اگرچه تمهیدات قابل توجهی در تغییرات به SNMP وارد شده، که در این یادداشت منعکس شده است، اما پروتکل حاصل با نسخه قبلی اش یعنی پروتکل نظارت ساده بر درگاه[[5]](#footnote-5) (SGMP) سازگاری چندانی ندارد [8]. با اینکه سینتکس این پروتکل تغییر کرده است، اما اصالت اولیه، تصمیمات طراحی و معماری حفظ شده است. به منظور اجتناب از تداخل، پورت های UDP جدید برای کارکرد پروتکل توصیف شده در این یادداشت اختصاص یافته است.

۳. معماری SNMP

به طور ضمنی مدل ساختاری SNMP مجموعه­ای از ایستگاه­های مدیریت و عناصر شبکه است. این ایستگاه­ها، برنامه­های مدیریتی را اجرا می­کنند، که عناصر شبکه­ای را مورد نظارت و کنترل قرار ­دهند. عناصر شبکه­ای دستگاه هایی مانند میزبان ها، گیت وی ها، سرورهای پایانه­ای و موارد مشابه هستند، که دارای عامل­های[[6]](#footnote-6) مسئول اجرای عملیات مدیریتی، هستند که توسط ایستگاه­های مدیریت شبکه درخواست می­شود. پروتکل مدیریت ساده شبکه (SNMP) به منظور برقراری ارتباط مدیریت اطلاعات میان ایستگاه­های مدیریت شبکه و عامل­ها، در عناصر شبکه مورد استفاده قرار می­گیرد.

۳.۱. اهداف ساختار

پروتکل SNMP به وضوح، تعداد و پیچیدگی عملیات مدیریتی را که توسط عامل مدیریتی ادراک می­شود، به حداقل می­رساند. این هدف حداقل از چهار جنبه مورد توجه است:

۱. هزینه توسعه نرم افزار عامل مدیریتی که برای پشتیبانی از پروتکل ضروری است, کاهش می­یابد.

۲. درجه عملکرد مدیریتی که از راه دور مدیریت می­شود افزایش می­یابد، بنابراین اجازه استفاده کامل را از منابع اینترنت در وظایف مدیریتی می­دهد.

۳. درجه عملکرد مدیریت که از راه دور مدیریت می­شود افزایش می یابد، بنابراین کمترین محدودیت های ممکن را بر روی شکل و دشواری ابزار های مدیریت اعمال می­کند.

۴. مجموعه­های ساده شده­ای از عملکردهای مدیریت به آسانی درک می­شوند و توسط توسعه دهندگان ابزارهای مدیریت شبکه مورد استفاده قرار می­گیرند.

هدف دوم پروتکل این است که پدیده عملکردی برای نظارت و کنترل به اندازه کافی قابل تعمیم باشد تا بتواند جنبه­ های اضافی عملکرد و مدیریت شبکه را برآورده سازد.

هدف سوم این است که ساختار تا جای ممکن، مستقل از ساختار و مکانیزم میزبان­ها یا گیت وی های خاصی باشد.

۳.۲. عناصر ساختار

ساختار (معماری) SNMP راهکاری را برای مساله مدیریت شبکه بر حسب موارد ذیل پیشنهاد می­کند:

۱. حوزه مدیریت اطلاعات در ارتباط با پروتکل

۲. ارائه مدیریت اطلاعات در ارتباط با پروتکل

۳. شکل و ابزار تبادل در میان موجودیت های مدیریت

۴. اقدامات بر روی مدیریت اطلاعات تحت پشتیبانی پروتکل

۵. تعریف رابطه مدیریتی میان موجودیت های مدیریت

۶. شکل و ابزار ارجاع ها به مدیریت اطلاعات

۳.۲.۱. حوزه مدیریت اطلاعات

حوزه مدیریت اطلاعات در ارتباط با عملیات SNMP دقیقا به صورتی است که توسط نمونه­های تمام اشیا غیر متراکم نمایش داده می­شود، که یا در MIB اینترنت استاندارد، و یا در محل دیگری، مطابق با قرارداد های ابتدایی در SMI اینترنت-استاندارد، تعریف شده باشند [5].

پشتیبانی از انواع اشیا یکپارچه در MIB، نه برای سازگاری با SMI نیاز است و نه توسط SNMP شناخته می­شوند.

۳.۲.۲. ارائه مدیریت اطلاعات

مدیریت اطلاعات در ارتباط با عملیات SNMP مطابق با زیر مجموعه ای از زبان ANS.1 [9] ارائه می­شود که برای تعریف انواع غیر یکپارچه در SMI مشخص شده است.

SGMP قرارداد استفاده از یک زیر مجموعه تعریف شده از زبان ANS.1 را پذیرفت [9]. SNMP نیز این سنت را با استفاده از مجموعه پیچیده تری از ANS.1، برای توصیف اشیای مدیریت شده و توصیف واحد های داده پروتکل مورد استفاده برای مدیریت آن اشیا، ادامه و توسعه می­دهد. علاوه بر این، تمایل برای ساده سازی انتقال به پروتکل های مدیریت شبکه مبتنی بر OSI منجر به تعریف زبان (استاندارد اینترنت) ساختار مدیریت اطلاعات (SMI) [5] در ANS.1 و پایگاه مدیریت اطلاعات (MIB) شد [6]. استفاده از زبان ANS.1، به دلیل استفاده موفقیت آمیز از ANS.1 در تلاش­های اولیه، به خصوص در SGMP، مورد حمایت قرار گرفت. محدودیت­های موجود در استفاده از ANS.1 که بخشی از SMI هستند، به ساده سازی کمک می­کنند که توسط SGMP حمایت و اعتبار سنجی می­شود.

همچنین برای تداوم سادگی، SNMP از زیرمجموعه­ای از قوانین کدگذاری پایه­ای ANS.1 استفاده می­کند [10]. تمامی کدگذاری­ها از شکل طول معین استفاده می­کنند. علاوه بر این هر زمانی که امکان پذیر باشد کدگذاری های غیر سازنده به جای کدگذاری­های سازنده استفاده می­شوند. این محدودیت به تمامی جنبه­های کدگذاری ANS.1، هم برای واحد های داده پروتکل سطح بالا و هم اشیای داده اعمال می­شود.

۳.۲.۳. اقدامات مورد پشتیبانی در مدیریت اطلاعات

SNMP تمامی کارکرد های عامل مدیریت را به عنوان تغییرات یا بررسی متغیر ها مدل سازی می­کند. بنابراین موجودیت یک پروتکل بر روی میزبان که منطقا در راه دور (احتمالا خود عنصر شبکه) قرار دارد، با عامل مدیریتی ساکن بر روی عنصر شبکه به منظور دستیابی(get) یا تغییر(set) متغیرها، تعامل می­کند. این استراتژی حداقل دو نتیجه مثبت دارد:

۱. دارای اثر محدود سازی تعداد کارکرد های اساسی مدیریت که توسط عامل مدیریتی ادراک می شوند به دو کارکرد است: یک، اقدام برای اختصاص یک مقدار به یک پیکربندی خاص یا پارامتر دیگر؛ و دو، دستیابی به چنین مقداری.

۲. اثر دوم این تصمیم، اجتناب از القا به درون پشتیبانی تعریف پروتکل برای فرامین اجرایی مدیریتی: تعداد چنین فرمان هایی به صورت عملی در حال افزایش است و مفهوم چنین فرامینی به طور کلی پیچیده می­باشد.

استراتژی ضمنی در SNMP، به صورتی است که نظارت وضعیت شبکه، در هر سطح قابل توجهی از جزئیات، اساسا از طریق رای گیری برای اطلاعات مناسب، در بخشی از مراکز نظارت انجام می­شود. تعداد محدودی از پیام های خودجوش (trap)، زمان بندی و تمرکز رای­گیری را هدایت می­کنند. محدود کردن تعداد پیام­های خودجوش، با هدف ساده سازی و حداقل کردن مقدار ترافیک تولیدی توسط عملکرد مدیریت شبکه سازگاری دارد.

حذف دستورات امری از مجموعه کارکرد های مدیریتی که پشتیبانی می­شوند، عملکردهای مطلوب عامل مدیریتی را مسدود نمی­کند. در حال حاضر بیشتر فرامین، درخواست هایی برای تنظیم برخی پارامترها یا دستیابی به یک مقدار است، و عملکرد برخی فرامین در حال حاضر به سادگی در مد غیرهمزمان، توسط این مدل مدیریتی اجرا می­شود. در این روش، یک فرمان می­تواند به عنوان تنظیم مقدار یک پارامتر تحقق یابد که عملکرد مطلوب را تنظیم می­کند. به عنوان مثال در عوض پیاده سازی یک "فرمان ریبوت"، این اقدام می­تواند با تنظیم یک پارامتر به عنوان نشان دهنده تعداد ثانیه­های باقیمانده تا ریبوت سیستم، تعبیر شود.

۳.۲.۴. شکل و ابزار تبادل های پروتکل

ارتباط مدیریت اطلاعات در میان موجودیت های مدیریت در SNMP از طریق تبادل پیام­های پروتکل محقق می شود. شکل و ابزار این پیام ها در دیل در بخش 4 تعریف شده است.

سازگار با هدف حداقل سازی پیچیدگی مدیریت عامل، تبادل پیام های SNMP نیازمند تنها یک سرویس دیتاگرام غیر قابل اتکا است و هر پیام به طور کامل و مستقلا توسط یک دیتاگرام انتقال ارائه می­شود. هنگامی که این مستند تبادل پیام ها توسط پروتکل UDP مشخص می­شود[11]، مکانیزم ها SNMP به طور کلی برای استفاده با انواع مختلفی از خدمات انتقالی مناسب هستند.

۳.۲.۵. تعریف ارتباطات مدیریتی

ساختار SNMP اجازه ارتباطات مدیریتی متنوعی را در میان موجودیت هایی که در پروتکل همکاری می­کنند، می­دهد. موجودیت­های واقع در ایستگاه­ های مدیریتی و عناصر شبکه که با یکدیگر بوسیله SNMP ارتباط برقرار می­کنند، موجودیت­های کاربرد SNMP نامیده می­شوند. پردازه­های همتا که SNMP را پیاده سازی کرده، و بنابراین موجودیت های کاربری SNMP را پشتیبانی می کنند، موجودیت های پروتکل نامیده می­شوند.

زوج یک عامل SNMP، با مجموعه ای دلخواه از موجودیت های کاربردی SNMP، یک جامعه SNMP [[7]](#footnote-7) نامیده می­شود. هر جامعه SNMP توسط رشته ای از هشت بیتی ها نامگذاری می شود که به آن نام جامعه گفته می شود.

یک پیام SNMP که از یک موجودیت کاربرد SNMP ناشی می­شود و در واقع به جامعهSNMP ای تعلق دارد، که هم نام با مؤلفه نام جامعه پیام است، یک پیام معتبر SNMP نامیده می­شود. مجموعه ای از قوانین که توسط آن­ها، پیام SNMP به عنوان یک پیام معتبر SNMP برای یک جامعه SNMP خاص شناسایی می­شود، یک رویه احراز هویت نامیده می­شود. پیاده سازی کارکردی که پیام معتبر SNMP را مطابق با یک یا چند رویه احراز هویت شناسایی می­کند، یک سرویس احراز هویت نامیده می­شود.

به صورت واضح، مدیریت اثربخش ارتباطات مدیریتی در میان موجودیت­های کاربری SNMP نیازمند سرویس های احراز هویتی است که (با استفاده از رمزنگاری یا سایر تکنیک­ها) قادر به شناسایی پیام­های معتبر SNMP با درصد بالایی از اطمینان هستند. برخی پیاده سازی­های SNMP ممکن است تنها از یک سرویس احراز هویت پشتیبانی کنند که تمامی پیام­های SNMP را به عنوان پیام­های معتبر SNMP شناسایی می­کند.

برای هر عنصر شبکه، زیرمجموعه ای از اشیاء در MIB که به آن عنصر تعلق دارد، یک SNMP MIB view نامیده می­شود. توجه داشته باشید که نام های اشیا ارائه شده در SNMP MIB view، نیازی به تعلق به زیردرختی از فضای نام شی ندارد.

عنصری از مجموعه {تنها خواندنی، تنها نوشتنی}، یک حالت دسترسی SNMP نامیده می­شود.

زوجی از یک حالت دسترسی SNMP با یک SNMP MIB view یک SNMP community profile نامیده می­شود. یک SNMP community profile مجوزهای خاص دسترسی به متغیرها در یک MIB view مشخص را نمایش می­دهد. برای هر متغیر در MIB view در یک SNMP community profile داده شده، دسترسی به آن متغیر توسط پروفایل، مطابق با قرارداد های ذیل ارائه می­شود:

1. اگر گفته شود متغیر در MIB به همراه Access : از None تعریف شده است، برای هر اپراتوری در دسترس نخواهد بود.

2. اگر گفته شود متغیر در MIB به صورت Access : از read-write یا read-only تعریف شده است و حالت درسترسی به پروفایل داده شده read-write باشد، آن متغیر به عنوان یک operand برای عملیات های get، set و trap در دسترس خواهد بود.

3. در غیر این صورت متغیر به عنوان یک operand برای عملیات­های get و trap در دسترس خواهد بود.

4. در مواردی که متغیر های write-only یک operand استفاده شده برای عملیات های get یا trap است، مقدار داده شده برای متغیر مخصوص پیاده سازی است.

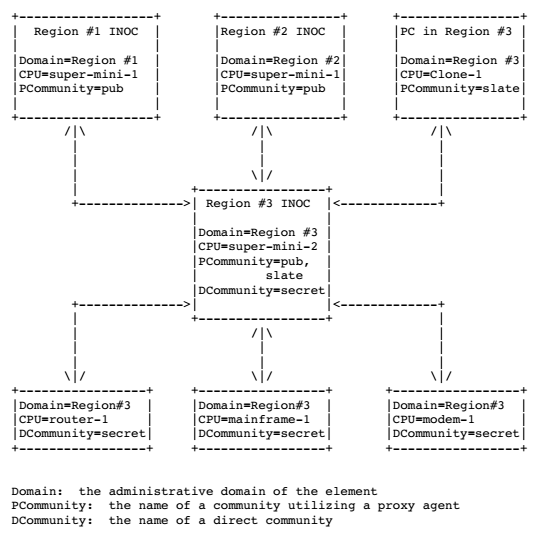
یک زوج از SNMP community به همراه SNMP community profile یک SNMP access policy نامیده می­شود. یک قانون دسترسی یک community profile مشخص را نشان می­دهد که توسط عامل SNMP از یک SNMP community خاص برای اعضای دیگر آن community فراهم شده است. تمامی روابط مدیریتی در میان موجودیت­های کاربری SNMP بر حسب قوانین دسترسی SNMP تعریف شده­اند.

برای هر قانون دسترسی SNMP اگر عنصر شبکه ای که بر مبنای آن عامل SNMP برای SNMP community خاص قرار می­گیرد به صورتی نیست که MIB view برای پروفایل مشخص به آن دسترسی یابد، سپس آن قانون یک SNMP proxy access policy نامیده می­شود. عامل SNMP مرتبط با یک قانون دسترسی پروکسی یک عامل پروکسی SNMP نامیده می­شود. در حالی که تعریف غیر دقیق قوانین دسترسی پروکسی می­تواند منجر به حلقه­های مدیریت شود، تعریف قوانین پروکسی حداقل از دو جنبه مفید است:

1. اجازه نظارت و کنترل عناصر شبکه را که در هر صورت غیر قابل آدرسی دهی هستند را با استفاده از پروتکل مدیریت و پروتکل انتقال می­دهد. این به معنای آن است که یک عامل پروکسی ممکن است یک کارکرد تبدیل پروتکل را ایجاد نماید که به یک ایستگاه مدیریت اجازه می­دهد تا تنها یک ساختاز سازگار با تمامی عناصر شبکه را که شامل دستگاه­هایی چون مودم­ها، مالتی پلکسرها و سایر دستگاه­ها که از ساختار مختلف مدیریتی پشتیبانی می­کنند را به کار گیرد.

2. این ها عناصر شبکه­ای را از قوانین کنترل دسترسی حفظ می­کند. به عنوان مثال یک عامل پروکسی ننمن است کنترل دسترسی پیچیده را پیاده سازی کند در حالی که زیرمجموعه های مختلف از متغیرها در داخل MIB به ایستگاه های مختلف مدیریتی بدون افزایش پیچیدگی عنصر شبکه دسترسی پیدا می­کنند.

طبق مثال، شکل 1 رابطه میان ایستگاه­های مدیریت، عامل­های پروکسی و عامل­های مدیریت را نمایش می­دهد. در این مثال عامل پروکسی به مرکز عملیات­های شبکه اینترنت[[8]](#footnote-8) (INOC) حوزه مدیریتی مربوط شده است که دارای رابطه مدیریتی استاندارد با مجموعه­ای از عامل­های مدیریت است.

شکل 1 – مثال مدیریت پیکربندی شبکه

3-2-6 شکل و مفهوم ارجاع ها به اشیای مدیریت شده

SMI نیازمند تعریف آدرس پروتکل مدیریتی است:

1. وضوح ارجاع های MIB

2. وضوح ارجاع های MIB در حضور چندین نسخه MIB

3. شناسایی نمونه های ویژه ای از انواع شی تعریف شده در MIB

3-2-6-1 وضوح ارجاع های MIB

از آنجا که حوزه هر عملیات SNMP به صورت مفهوم به اشیای مرتبط با یک عنصر شبکه محدود شده است و به این دلیل که تمامی ارجاع­های SNMP به اشیای MIB ( به صورت واضح یا ضمنی) نام های متغیر یکتایی هستند، این امکان وجود ندارد که هر گونه ارجاع به هر شی تعریف شده در MIB بتواند به چندین نمونه از آن نوع شناسایی شود.

2-2-6-2 وضوح ارجاع ها در میان نسخه های MIB

نمونه شی ارجاع شده توسط هر عملیات SNMP دقیقا به عنوان بخشی از درخواست عملیات ( یا در وضعیت یک عملیات get-next ) یا موفقیت لحظه ای آن در MIB به عنوان یک کل مشخص می شود. به طور خاص ارجاع به یک شی به عنوان بخشی از برخی نسخه های MIB اینترنت استاندارد به هر شی ای که بخشی از آن نسخه گفته شده MIB اینترنت استاندارد نیست تعبیر نمی شود، به استثنای وضعیتی که عملیات مورد درخواست get-next است و نام شی خاص در میان نام های تمام اشیای حاضر به عنوان بخشی از نسخه گفته شده MIB اینترنت استاندارد وجود داشته باشد.

3-2-6-3 شناسایی نمونه های شی

نام های برای تمامی انواع شی در MIB به صورت واضح یا بر حسب MIB اینترنت استاندارد یا سایر مستندات مربوط به قواعد نام گذاری SMI تعریف شده­اند. SMI نیازمند پروتکل­های مدیریتی است که مکانیزم­های شناسایی نمونه­های آن انواع شی را برای عنصر خاص شبکه تعریف می­کند.

هر نمونه از هر شی تعریف شده در MIB در عملیات­های SNMP توسط یک نام یکتا تعریف می­شود که " نام متغیر" نامیده می­شود. به طور کلی یک متغیر SNMP یک شناسه شی به شکل x.y است که x نام یک نوع شی غیر یکپارچه تعریف شده در MIB است و y یک بخش شناسه شی است که به یک روشی به نوع شی نامگذاری شده مرتبط است و نمونه مطلوب را تعریف می کند.

این روش نامگذاری اجازه استخراج کامل مفاهیم GetNextRequest-PDU را می دهد (بخش 4 را ببینید) چرا که نام هایی را برای متغیر های مربوطه اختصاص می دهد که اولویت بندی نام های متغیر شناخت شده در MIB واضح باشد.

نامگذاری نوع ویژه نمونه های شی در ذیل برای تعدادی از دسته های انواع شی تعریف شده است. نمونه های یک نوع شی به صورتی است که هیچ یک از قرارداد های نامگذاری ذیل که توسط شناسه شی به شکل x.0 نام گذاری می شوند، قابل کاربرد نیستند که در آن x نام گفته شده نوع شی در تعریف MIB است.

به عنوان نمونه فرض کنید فردی می خواهد که یک نمونه از متغیر sysDescr را شناسایی کند، کلاس شی برای sysDescr برابر است با :



بنابراین نوع شی، x، می­تواند 1.3.6.1.2.1.1.1 باشد که یک زیر تعریف کننده نمونه از 0 است. یعنی 1.3.6.1.2.1.1.1.0 یک و تنها نمونه sysDescr را شناسایی می کند.

3-2-6-3-1 نام های نوع شی ifTable

نام یک رابط زیرشبکه، s، مقدار شناسه شی به شکل i است که i مقداری دارد که آن نمونه نوع شی ifIndex با s مرتبط است.

برای هر نوع شی، t، که برای آن نام تعریف شده، n دارای یک پیشوند ifEntry است، یک نمونه i از t توسط یک شناسه شی به شکل n.s تعریف می شود که s نام رابط زیرشبکه است که i اطلاعات را نمایش می دهد.

به عنوان مثال فرض کنید شخصی می خواهد نمونه متغیر ifType مرتبط با رابط 2 را شناسایی کند. در نتیجه ifType.2 می تواند نمونه مطلوب را شناسایی کند.

2-2-6-3-2 نام های نوع شی atTable

نام آدرش شبکه AT-cached، x، یک شناسه شی به شکل 1.a.b.c.d است که a.b.c.d مقدار ( در نشانه گذاری نقطه ای) نوع شی atNetAddress مربوط به x است.

نام یک معادل تبدیل آدرس e یک مقدار شناسه شی به شکل s.w است به طوری که s مقدار آن نمونه نوع شی atIndex مرتبط با e است و به صورتی است که w نام آدرس شبکه AT-cached مرتبط با e است.

برای هر نوع شی، t، که نام تعریف شده، n، دارای یک پیشوند atEntry است، یک نمونه، i از t توسط شناسه شی به شکل n.y نامگذاری می شود که y نام معادل آدرس تبدیل است که i اطلاعات را نمایش می دهد.

به عنوان مثال فرض کنید که شخصی می خواهد آدرس فیزیکی یک ورودی در جدول تبدیل آدرس (ARP cache) مرتبط با یک IP آدرس 89.1.1.42 و رابط 3 را پیدا کند. در نتیجه atPhysAddress.3.1.89.1.1.42 می تواند نمونه مطلوب را شناسایی کند.

3-2-6-3-3 نام های نوع شی ipAddrTable

نام عنصر شبکه که از طریق IP قابل آدرس دهی باشد، x، شناسه شی به شکل a.b.c.d به صورتی است که a.b.c.d مقدار ( در نشانه گذاری نقطه ای) آن نمونه نوع شی ipAdEntAddr در ارتباط با x است.

برای هر نوع شی t که برای آن نام تعریف شده n دارای پیشوند ipAddrEntry است، نمونه، i از t توسط یک شناسه شی به شکل n.y نام گذاری می شود که در آن y نام عنصر شبکه ای قابل آدرسی دهی ت.سط IP است که i اطلاعات را نمایش می دهد.

به عنوان مثال فرض کنید شخصی می خواهد network mask وروید در جدول رابط IP مربوط به IP آدرس 89.1.1.42 را پیدا کند. در نتیجه ipAdEntNetMask.89.1.1.42 می تواند نمونه مطلوب را شناسایی کند.

3-2-6-3-4 نام های نوع شی ipRoutingTable

نام یک مسیر IP، x ، شناسه شی شکل a.b.c.d به طوری است که a.b.c.d مقدار ( به شکل نمایش نقطه ای) نمونه نوع شی ipRouteDest مرتبط با x است.

برای هر نوع شی t که برای آن نام تعریف شده، n دارای پیشوند ipRoutingEntry است، نمونه i از t توسط یک شناسه شی به شکل n.y است که در آن y نام مسیر IP است که i اطلاعات را نمایش می دهد.

به عنوان نمونه فرض کنید که شخصی می خواهد مسیر بعدی یک ورودی در جدول مسیر یابی مرتبط با مقصد 89.1.1.42 را پیدا کند. در نتیجه ipRouteNextHop.89.1.1.42 می تواند نمونه مطلوب را شناسایی کند.

3-2-6-3-5 نام های نوع شی tcpConnTable

نام ارتباط TCP، x، شناسه شی شکل a.b.c.d.e.f.g.h.i.j است به طوری که a.b.c.d مقدار ( در نشانه گذاری "نقطه ای") آن نمونه نوع شی tcpConnLocalAddress مرتبط با x است و به طوری که f.g.h.i مقدار ( در شکل آشنای "نقطه ی") آن نمونه نوع شی tcpConnRemoteAddress است مرتبط با x است و به طوری است که e مقدار آن نمونه نوع شی tcpConnLocalPort مرتبط با x است و طوری است که j مقدار آن نمونه نوع شی tcpConnRemotePortمرتبط با x است.

برای هر نوع شی، t، که نان تعریف شده، n، است دارای پیشوند tcpConnEntry است، یک نمونه، i از t توسط یک شناسه شی به شکل n.y نامگذاری می شود که در آن y نام ارتباط TCP است که i اطلاعات را نمایش می دهد.

به عنوان مثال فرض کنید فردی می خواهد وضعیت ارتباط TCP را بین آدرس محلی 89.1.1.42 بر روی پورت TCP 21 و آدرس راه دور 10.0.0.51 بر روی پورت2059 TCP را پیدا کند. در نتیجه tcpConnState.89.1.1.42.21.10.0.0.51.2059 می تواند نمونه مطلوب را شناسایی کند.

3-2-6-3-6 نام های نوع شی egpNeighTable

نام یک همسایه EGP، x، شناسه شی[[9]](#footnote-9) شکل a.b.c.d است به طوری که یک a.b.c.d مقدار ( در نشانه گذاری "نقطه ای") آن نمونه نوع شی egpNeighAddr مربوط به x است.

برای هر نوع شی، t، که برای آن نام تعریف شده، n، دارای یک پیشوند egpNeighEntry است، یک نمونه، i، از t توسط یک شناسه شی به شکل n.y است که در آن y نام همسایه EGP است که i اطلاعات را نمایش می­دهد.

به عنوان نمونه فرض کنید که شخصی می هواهد وضعیا همسایه را برای IP آدرس 89.1.1.42 پیدا کند. در نتیجه egpNeighState.89.1.1.42 می تواند نمونه مطلوب را شناسایی کند.

4- مشخصه پروتکل

پروتکل مدیریت شبکه یک پروتکل کاربردی است که توسط آن متغیرهای عامل MIB ممکن است مورد بازرسی یا تغییر قرار بگیرد.

ارتباط میان موجودیت­های پروتکل توسط تبادل پیام ها انجام می شود که کاملا و به طور مستقل از یک دیتاگرام UDP واحد با استفاده از قوانین کدگذاری ANS.1 نمایش داده می شود (مطابق با بخش 3-2-2). یک پیام از یک نسخه شناسه، یک نام SNMP Community و واحد داده پروتکل[[10]](#footnote-10) (PDU) تشکیل می شود. یک موجودیت پروتکل پیام ها را در پورت UDP 161 در میزبان دریافت می کند به طوری که به تمامی پیام ها به استثنای آن هایی که ترپ ها را گزارش می دهند، مربوط می شود (یعنی تمامی پیام ها به استثنای آن هایی که حاوی Trap-UDP هستند). پیام هایی که ترپ ها را پیشنهاد می دهند می بایست در پورت UDP 162 برای پردازش بعدی دریافت شوند. یک پیاده سازی این نیازمندی پروتکل پیام هایی که طول اشان به 484 اکتت می رسد مورد پذیرش قرار نمی گیرد. هرچند توصیه می شود که پیاده سازی ها از دیتاگرام های بزرگتر هر زمان که ممکن باشد، پشتیبانی می کند.

ضروری است که تمامی پیاده سازی های SNMP از پنج PDU پشتیبانی کند که عبارتند از:

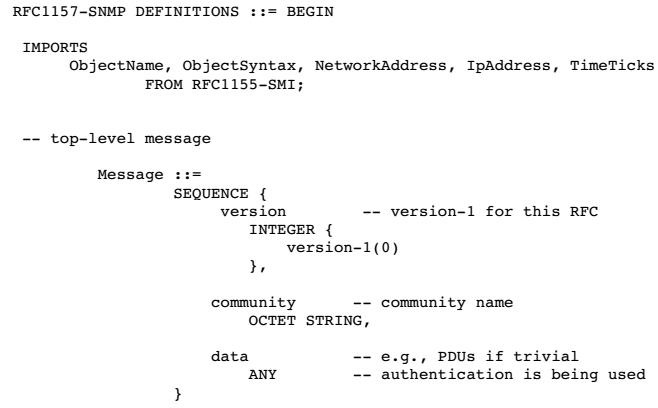
GetRequest-PDU

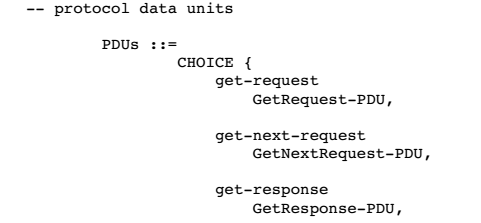
GetNextRequest-PDU

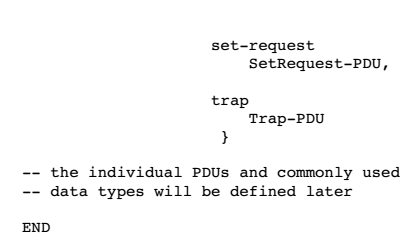
GetResponse-PDU

SetRequest-PDU

Trap-PDU







**۴. خصوصیات پروتکل**

پروتکل مدیریت شبکه یک پروتکل کاربردی(Application Protocol) می­باشدکه توسط آن متغیر­های موجود در MIB یک عامل مورد بررسی قرار گرفته و یا تغییر داده می­شوند.

ارتباط میان واحدهای پروتکلی از طریق مبادله پیام انجام می­شود، که هر پیام به طور کامل و مستقل در یک واحد دیتاگرام UDP با استفاده از قوانین کدگذاری ASN.1 نمایش داده می­شود.(همانطور که در بخش 3.2.2 بحث شد) . یک پیام شامل یک شناسه نسخه، یک نام جامعه SNMP و یک واحد داده پروتکلی (PDU)می­باشد. یک واحد پروتکلی پیام­ها را بر روی پورت UDP، 161 میزبان دریافت میکند،‌ این پورت برای دریافت همه­ی پیام­ها به جز گزارش­های Trap مورد استفاده قرار می­گیرد(همه­ی پیام ها به جز پیام­هایی که از نوع Trap-PDU هستند). پیام­هایی که شامل گزارش­های Trap هستند برای پردازش بیشتر باید بر روی پورت UDP، 162 دریافت شوند. در یک پیاده­سازی این پروتکل نیازی نیست پیام­هایی که دارای طول بیشتر از 484 بایت هستند،‌ پذیرش شوند. با این حال، توصیه می شود، پیاده سازی هایی که دیتاگرام­های بزرگتر را پشتیبانی می­کنند در هر زمان که امکان پذیر است.

در همه­ی پیاده­سازی­های SNMP الزامی است که از پنج نوع PDU‌پشتیبانی کنند:

GetRequest-PDU

GetNextRequest-PDU

GetResponse-PDU

SetRequest-PDU

and Trap-PDU.

RFC1157-SNMP DEFINITIONS ::= BEGIN

IMPORTS

ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks

FROM RFC1155-SMI;

-- top-level message

Message ::=

SEQUENCE {

version -- version-1 for this RFC

INTEGER {

version-1(0)

},

community -- community name

OCTET STRING,

data -- e.g., PDUs if trivial

ANY -- authentication is being used

}

--واحد­های داده پروتکلی

PDUs ::=

CHOICE {

get-request

GetRequest-PDU,

get-next-request

GetNextRequest-PDU,

get-response

GetResponse-PDU,

set-request

SetRequest-PDU,

trap

Trap-PDU

}

--PDU های مستقل و معمولا مورد استفاده قرار می­گیرند.

--انواع داده بعدا تعریف خواهند شد

END

4.1 **عناصر پردازه**

در این بخش عملیات یک واحد پروتکلی SNMP بررسی می­شود. اما توجه داشته باشید ، که این عملیات قصدی برای محدود کردن معماری داخلی هیچ پیاده­سازی منطبقی را ندارد.

در متنی که در ادامه می­آید، اصطلاح Transport Address مورد استفاده قرار گرفته است. در مورد UDP، یک Transport Address شامل یک آدرس IP به همراه یک پورت UDP می­باشد. ممکن است سایر خدمات انتقال نیز برای پشتیبانی از SNMP مورد استفاده قرار گیرد. در این موارد، تعریف آدرس انتقال (Transport Address) باید مطابق آن ساخته شود.

اقدامات سطح بالای یک واحد پروتکلی که یک پیام را تولید می­کند به شرح زیر است :

1. در ابتدا یک واحد داده پروتوکل) (PDU مناسب ایجاد می­کند. به عنوان مثال، GetRequest-PDU ، به عنوان یک شی ASN.1
2. سپس این شی ASN.1 به همراه یک نام جامعه و آدرس انتقال مبدا و آدرس انتقال مقصد،‌ به سرویسی که روش احراز هویت دلخواه را انجام می­دهد، هدایت خواهد شد. این سرویس احرازهویت یک شی ASN.1 دیگر را برمی­گرداند.
3. واحد پروتکلی یک شی پیام ASN.1 را با استفاده از نام جامعه و نتیجه شی ASN.1 ایجاد می­کند.
4. این شی ASN.1 جدید با استفاده از قوانین پایه کدگذاری ASN.1 به صورت سریالی شده در خواهد آمد و به وسیله یک سرویس انتقال به واحد پروتکلی متناظر فرستاده می­شود.

به طور مشابه، عملیات سطح بالای واحد پروتکلی که پیام را دریافت می­کند، به شرح زیر است :

1. برای ایجاد شی ASN.1 مربوط به با یک شی پیام ASN.1 ، یک تجزیه ابتدایی بر روی دیتاگرام ورودی انجام می­شود. . اگر که تجزیه با شکست مواجه شود،‌ دیتاگرام حذف خواهد شد و اقدام دیگری انجام نخواهد شد.
2. سپس شماره نسخه پیام SNMP را تایید می­کند، در صورت عدم تطابق، دیتاگرام حذف خواهد شد و اقدام دیگری انجام نخواهد شد.
3. سپس واحد پروتکلی نام جامعه و داده­های کاربر یافت شده در شی پیام ASN.1 را به همراه آدرس­های مبدا و مقصد دیتاگرام به سرویسی که روش احرازهویت دلخواه را انجام می­دهد، هدایت می­کند. این واحد یک شی ASN.1 دیگر و یا سیگنال شکست احراز هویت را به عنوان نتیجه برمی­گرداند. در مورد دوم، واحد پروتکلی این شکست را ثبت کرده و (احتمالا) یک پیام Trap تولید می­کند و این دیتاگرام را حذف کرده و اقدام دیگری انجام نخواهد شد.
4. واحد پروتکلی یک تجزیه ابتدایی بر روی شی ASN.1 بازگردانده شده توسط سرویس احراز هویت انجام می­دهد تا یک شی ASN.1 مربوط به شی PDU، ASN.1 ایجاد کند. اگر که تجزیه با شکست مواجه شود، دیتاگرام حذف خواهد شد و اقدام دیگری انجام نخواهد شد. در غیر این صورت، با استفاده از نام جامعه، پروفایل مناسب انتخاب می­شود و PDU بر اساس آن پردازش می­شود. اگر به عنوان نتیجه این پردازش، یک پیام برگردانده شده، پس آدرس منبعی که پیام پاسخ از آن فرستاده شده است باید برابر با آدرس مقصدی باشد که پیام درخواست اصلی از آن فرستاده شده است.

**4.1.1. ساختارهای متداول**

قبل از معرفی ۶ نوع PDU پروتکل، بهتر است برخی ساختارهای ASN.1 را که مکررا مورد استفاده قرار می­گیرند بررسی کنیم:

-- request/response information

RequestID ::=

INTEGER

ErrorStatus ::=

INTEGER {

noError(0),

tooBig(1),

noSuchName(2),

badValue(3),

readOnly(4)

genErr(5)

}

ErrorIndex ::=

INTEGER

-- variable bindings

VarBind ::=

SEQUENCE {

name

ObjectName,

value

ObjectSyntax

}

VarBindList ::=

SEQUENCE OF

VarBind

RequestIDها برای تمایز قائل شدن بین درخواست­های انجام نشده استفاده می­شود. با استفاده از RequestID، یک واحد کاربردی SNMP می­تواند، پاسخ­های دریافتی را با درخواست­ها ارتباط دهد. در مواردی که سرویس دیتاگرام غیر قابل اعتماد است، استفاده می­شود، RequestID همچنین یک روش ساده برای شناسایی پیام­های تکراری در شبکه ایجاد می­کند.

یک نمونه غیر صفر از ErrorStatus نشان می­دهد که در زمان پردازش یک درخواست خطایی رخ داده است،در این موارد، ممکن است ErrorIndex با مشخص کردن اینکه کدام متغیرها در لیست عامل ایجاد خطا بوده­اند، اطلاعات بیشتری در مورد خطای ایجاد شده، ارائه دهد.

اصطلاح متغیر (Variable) به یک نمونه از object manged اشاره دارد. یک متغیر متصل (Variable binding) یا VarBind به جفت­هایی از نام و مقدار متغیرها اشاره دارد. VarBindList یک لیست ساده از نام متغیرها به همراه مقدار متناظر آنهاست. برخی از انواع PDUها فقط شامل نام متغیر هستند و نه مقدار آن (مثل GetRequest-PDU ). در این مورد، واحد پروتکلی بخش مقدار یک متغیر را نادیده می­گیرد. با این حال، بخش مقدار هنوز هم باید دارای ساختار ASN.1 معتبر و کدگذاری باشد. توصیه می شود از مقدار NULL‌ برای بخش مقدار این گونه متغیرها استفاده شود.

**4.1.2. The GetRequest-PDU**

GetRequest-PDU به شکل زیر است:

GetRequest-PDU ::=

[0]

IMPLICIT SEQUENCE {

request-id

RequestID,

error-status -- always 0

ErrorStatus,

error-index -- always 0

ErrorIndex,

variable-bindings

VarBindList

}

GetRequest-PDU توسط واحد پروتکلی و فقط بنا به درخواست واحد کاربردی SNMP آن تولید می­شود.

پس از دریافت GetRequest-PDU، واحد پروتکلی دریافت کننده با توجه به قاعده مناسب از لیست زیر به آن پاسخ می­دهد:

1. اگر، برای هر متغیر که نام آن در فیلد variable-binding قرار دارد، نام متغیر با نام برخی از متغیرهای امکان­پذیر در MIB مرتبط دقیقا تطبیق نداشت، واحد دریافت کننده یک GetResponse-PDU به پدیدآورنده پیام با شکلی یکسان ارسال می­کند. به جز مقدار فیلد error-status که برابر noSuchName‌ است و مقدار فیلد error-index که ایندکس مولفه نام متغیر را در پیام دریافتی نشان می­دهد.
2. اگر، برای هر متغیر که نام آن در فیلد variable-binding قرار دارد، آن شی از نوع تجمیعی(aggregate) باشد (همان گونه که در SMI تعریف شده است)، واحد دریافت کننده یک GetResponse-PDU به پدیدآورنده پیام با شکلی یکسان ارسال می­کند. به جز مقدار فیلد error-status که برابر noSuchName‌ است و مقدار فیلد error-index که ایندکس مولفه نام متغیر را در پیام دریافتی نشان می­دهد.
3. اگر که اندازه پیام GetResponse-PDU تولید شده، همانگونه که در ادامه بیان شده،‌ بیش از محدودیت­های محلی شود، واحد دریافت کننده یک GetResponse-PDU به پدیدآورنده پیام با شکلی یکسان ارسال می­کند. به جز مقدار فیلد error-status که برابر tooBig‌ است و مقدار فیلد error-index که برابر با صفر است.
4. اگر، برای هر متغیر که نام آن در فیلد variable-binding قرار دارد، مقدار آن شی بنا به دلایلی که توسط هیچ یک از قاعده­های فوق پوشش داده نمی­شود، قابل بازیابی نباشد، واحد دریافت کننده یک GetResponse-PDU به پدیدآورنده پیام با شکلی یکسان ارسال می­کند. به جز مقدار فیلد error-status که برابر genErr است و مقدار فیلد error-index که ایندکس مولفه نام متغیر را در پیام دریافتی نشان می­دهد.

اگر هیچ یک از قاعده­های فوق اعمال نشود،‌ واحد پروتکلی دریافت­کننده یک پیام GetResponse-PDU‌ برای منشا پیام دریافت­شده ارسال میکند، به این صورت که، برای هر متغیر که نام آن در فیلد variable-binding پیام دریافتی قرار دارد، مولفه متناظر آن در GetResponse-PDU نام و مقدار آن متغیر را نشان خواهد داد. مقدار فیلد error-status در GetResponse-PDU برابر noError و مقدار فیلد error-index نیز برابر صفر خواهد یود. مقدار فیلد request-id در GetResponse-PDU همان مقداری خواهد بود که در پیام دریافت شده قرار داشته است.

**4.1.3. The GetNextRequest-PDU**

GetNextRequest-PDU دارای شکل یکسانی با GetRequest-PDU می­باشد، جز بخشی که نشان­دهنده نوع PDU می­باشد. در زبان ASN.1 :

GetNextRequest-PDU ::=

[1]

IMPLICIT SEQUENCE {

request-id

RequestID,

error-status -- always 0

ErrorStatus,

error-index -- always 0

ErrorIndex,

variable-bindings

VarBindList

}

GetNextRequest-PDU توسط واحد پروتکلی و فقط بنا به درخواست واحد کاربردی SNMP آن تولید می­شود.

پس از دریافت GetNextRequest-PDU، واحد پروتکلی دریافت کننده با توجه به قاعده مناسب از لیست زیر به آن پاسخ می­دهد:

1. اگر، برای نام هر متغیر در فیلد variable-binding ، آن نام بر اساس ترتیب الفبایی پیش از نام برخی از متغیرهای امکان­پذیر در MIB مرتبط وجود نداشت، واحد دریافت کننده یک GetResponse-PDU به پدیدآورنده پیام با شکلی یکسان ارسال می­کند. به جز مقدار فیلد error-status که برابر noSuchName‌ است و مقدار فیلد error-index که ایندکس مولفه نام متغیر را در پیام دریافتی نشان می­دهد.
2. اگر که اندازه پیام GetResponse-PDU تولید شده، همانگونه که در ادامه بیان شده،‌ بیش از محدودیت­های محلی شود، واحد دریافت کننده یک GetResponse-PDU به پدیدآورنده پیام با شکلی یکسان ارسال می­کند. به جز مقدار فیلد error-status که برابر tooBig‌ است و مقدار فیلد error-index که برابر با صفر است.
3. اگر، برای هر متغیر که نام آن در فیلد variable-binding قرار دارد، مقدار واژه جانشین برای نام متغیر، بنا به دلایلی که به وسیله قاعده­های فوق پوشش داده نمی­شود، قابل بازیابی نباشد، واحد دریافت کننده یک GetResponse-PDU به پدیدآورنده پیام با شکلی یکسان ارسال می­کند. به جز مقدار فیلد error-status که برابر genErr است و مقدار فیلد error-index که ایندکس مولفه نام متغیر را در پیام دریافتی نشان می­دهد.

اگر هیچ یک از قاعده­های فوق اعمال نشود،‌ واحد پروتکلی دریافت­کننده یک پیام GetResponse-PDU‌ برای پدیدآورنده پیام دریافت­شده ارسال میکند، به این صورت که، برای هر متغیر که نام آن در فیلد variable-binding پیام دریافتی قرار دارد، بخش متناظر آن در GetResponse-PDU نام و مقدار آن متغیر را نشان خواهد داد.

**4.1.3.1 مثالی از پیمایش جدول**

یکی از موارد مهم استفاده از PDU-GetNextRequest پیمایش جداول مفهومی اطالعات درون MIB است. معناشناسی این نوع پیام SNMP ،همراه با مکانیسم های مختص پروتکل برای شناسایی نمونه های منحصر بفرد انواع شی در MIB ،موجب دسترسی به اشیاء مرتبط در MIB می شود، اگر آنها از یک سازمان جدولی برخوردار باشند.

|  |  |  |
| --- | --- | --- |
| Metric | NextHop | Destination |
| 5 | 89.1.1.42 | 10.0.0.99 |
| 3 | 99.0.0.3 | 9.1.2.3 |
| 5 | 89.1.1.42 | 10.0.0.51 |

ایستگاه مدیریتی به عامل SNMP یک پیام PDU-GetNextRequest حاوی ارزش شناسه شی (OBJECT IDENTIFIER) نشان داده شده به عنوان نام متغیرهای مورد درخواست را ارسال می کند :

: GetNextRequest (ipRouteDest, ipRouteNextHop, ipRouteMetric1)

عامل SNMP با یک پیام PDU-GetResponse پاسخ می دهد:

GetResponse ((ipRouteDest.9.1.2.3 = "9.1.2.3”),

(ipRouteNextHop.9.1.2.3 = "99.0.0.3”),

(ipRouteMetric1.9.1.2.3 = 3))

ایستگاه مدیریتی ادامه می­دهد:

GetNextRequest (ipRouteDest.9.1.2.3,

ipRouteNextHop.9.1.2.3,

ipRouteMetric1.9.1.2.3))

عامل SNMP پاسخ می­دهد:

GetResponse ((ipRouteDest.10.0.0.51 = "10.0.0.51”),

(ipRouteNextHop.10.0.0.51 = "89.1.1.42”),

(ipRouteMetric1.10.0.0.51 = 5))

ایستگاه مدیریتی ادامه می­دهد:

GetNextRequest (ipRouteDest.10.0.0.51,

ipRouteNextHop.10.0.0.51,

ipRouteMetric1.10.0.0.51)

عامل SNMP پاسخ می­دهد:

GetResponse ((ipRouteDest.10.0.0.99 = "10.0.0.99”),

(ipRouteNextHop.10.0.0.99 = "89.1.1.42”),

(ipRouteMetric1.10.0.0.99 = 5))

ایستگاه مدیریتی ادامه می­دهد:

GetNextRequest (ipRouteDest.10.0.0.99,

ipRouteNextHop.10.0.0.99,

ipRouteMetric1.10.0.0.99)

ازآنجا که مدخل دیگری در جدول وجود ندارد، عامل SNMP اشیایی که به ترتیب قاموسی بعد از نام شی شناخته شده وجود دارند را برمی گرداند. این پاسخ پایان جدول مسیریابی را به ایستگاه مدیریتی اعالم می کند.

**4.1.4 The GetResponse-PDU**

ساختار پیام PDU-GetResponse مشابه PDU-GetRequest می باشد به جزنشانه ای که برای نوع PDU است. در زبان 1.ASN داریم:

GetResponse-PDU ::=

[2]

IMPLICIT SEQUENCE {

request-id

RequestID,

error-status

ErrorStatus,

error-index

ErrorIndex,

variable-bindings

VarBindList

{

پیام PDU-GetResponse توسط واحد پروتکل فقط پس از دریافت پیام PDU-GetRequest ، PDU-GetNextRequest ،و یا PDU-SetRequest که در جاهای دیگر این سند توصیف شده اند، تولید می شود. پس از دریافت PDU-GetResponse ،نهاد پروتکل دریافت کننده محتوای آن را به موجودیت برنامه SNMP ارائه می دهد.

**SetRequest-PDU 4.1.5**

فرم PDU-SetRequest بجز برای نشانه نوع با PDU-GetRequest یکسان میباشد PDU . در زبان 1.ASN::

SetRequest-PDU ::=

[3]

IMPLICIT SEQUENCE {

request-id

RequestID,

error-status -- always 0

ErrorStatus,

error-index -- always 0

ErrorIndex,

variable-bindings

VarBindList

}

PDU-SetRequest توسط یک موجودیت پروتکل، تنها با درخواست موجودیت برنامه SNMP ایجاد میشود.

به محض دریافت PDU-SetRequest ،دریافت کننده براساس هرگونه قانون مناسب در لیست زیر پاسخ میدهد:

.(1)اگر برای هر شئ نام گذاری شده در حوزه اتصاالت متغیر )field bindings-variable ،)شئ برای عملیات SET ، از دیدگاه مرتبط MIB در دسترس نباشد، بنابراین دریافت کننده PDU-GetResponse را با فرم یکسان، به عامل مبداء پیام دریافت شده ارسال میکند، جز اینکه مقدار رشته وضعیت خطا به noSuchName مقدار میگیرد، و مقدار رشته ایندکس خطا، برابر با مولفه ی نام شئ در پیام دریافتی میباشد

. (2) -اگر برای هر شئ نام گذاری شده در حوزه اتصاالت متغیر )field bindings-variable ،)محتویات رشته مقدار، بر اساس زبان 1.ASN ، نوعی، طول یا مقداری که با چیزی که متغیر نیاز دارد سازگار باشد را مشخص نکند، آنگاه موجودیت دریافت کننده PDU-GetResponse را با فرم یکسان، به عامل مبداء پیام دریافت شده ارسال میکند، جز اینکه مقدار رشته وضعیت خطا به badValue مقدار میگیرد، و مقدار رشته ایندکس خطا، برابر با مولفه ی نام شئ در پیام دریافتی میباشد

. (3) اگر سایز پیام نوع get response تولید شده که به عنوان زیر توضیج داده شده از یک محدودیت محلی تجاوز کند سپس موجودیت دریافت کننده یک پیام get-response-PDU برای تولید کننده پیام ارسال می کند که دارای فرم یکسان است به جز اینکه مقدار فیلد error-status وضعبت toobig و مقدار فیلد error-index ایندکس صفر را مشخص می‌کند.

(4)اگر برای نام هرشی در فیلد variable-binding مقدار نام شی بنا به دلایلی که به وسیله قوانین فوق تحت پوشش نیست قابل تغییر نباشد ، آنگاه موجودیت دریافت کننده یک پیام get-response- PDU برای تولید کننده پیام ارسال می‌کنند که دارای فرم یکسان است به جز اینکه مقدار فیلد error-status وضعبت generr و مقدار فیلد error-index ایندکس نام شی جزء در پیام دریافت شده را مشخص می‌کند.

اگر هیچ یک از قوانین فوق اعمال نشده آنگاه برای هر نام شی در فیلد variable-binding از پیام دریافت شده مقدار متناظر به متغییر اختصاص داده می‌شود.هر متغیر انتساب داده به وسیله set-request تعیین می‌شود باید به عنوان ، اگر به طور همزمان با توجه به همه انتساب های دیگر مشخص شده در همان پیام انجام شود.

موجودیت دریافت کننده یک پیام get-response-PDU برای تولید کننده پیام ارسال می کند که دارای فرم یکسان است به جز اینکه مقدار فیلد error-status وضعبت noerror و مقدار فیلد error-index ایندکس صفر را مشخص می‌کند.

**4.1.6. The Trap-PDU**:

شکل کلی Trap-PDU به شکل زیر است:

Trap-PDU ::=

[4]

IMPLICIT SEQUENCE {

enterprise -- type of object generating

-- trap, see sysObjectID in [5]

OBJECT IDENTIFIER,

agent-addr -- address of object generating

NetworkAddress, -- trap

generic-trap -- generic trap type

INTEGER {

coldStart(0),

warmStart(1),

linkDown(2),

linkUp(3),

authenticationFailure(4),

egpNeighborLoss(5),

enterpriseSpecific(6)

},

specific-trap -- specific code, present even

INTEGER, -- if generic-trap is not

-- enterpriseSpecific

time-stamp -- time elapsed between the last

TimeTicks, -- (re)initialization of the network

-- entity and the generation of the

variable-bindings -- "interesting" information

VarBindList

}

Trap-PDU به وسیله موجودیت پروتکل که تنها در درخواست موجودیت نرم افزاری snmp تولید می‌شود.وسیله ای است که موجویت نرم افزاری snmp آدرس مقصد از موجودیت نرم افزار snmp که پیاده سازی خاص است انتخاب می کند.

به محض دریافت trap-PDU موجودیت پروتکل دریافت کننده محتویات خود را برای موجودیت نرم اقزاری snmp خود ارائه می‌کند.

اهمیت جزء variable-binding از trap-PDU پیاده سازی خاص است.

شرح مقدارهای متفاوت فیلد generic-trap :

**4.1.6.1 The coldStart Trap**

coldStart(0) به معنی این است که موجودیت پروتکل ارسال کننده دوباره راه اندازی می‌شود به گونه ای که تنظیمات عامل یا اجرای موجودیت پروتکل ممکن است تغییر کند.

**4.1.6.2** **The warmStart Trap**

warmStart(1) به معنی این است که موجودیت پروتکل ارسال کننده دوباره راه اندازی می‌شود به گونه ای که پیکربندی عامل یا اجرای پروتکل موجودیت تغییر نکند.

**4.1.6.3** **The linkDown Trap**

linkDown(2) به معنی این است که موجودیت پروتکل ارسال کننده در یکی از لینک های ارتباطی ارائه شده درپیکربندی عامل یک مشکلی را تشخیص داده شده است.

در trap\_PDU در نوع link down اولین عنصر variable-binding آن حاوی نام و مقدار شی ifindex برای واسط تحت تاثیر است.

**4.1.6.4** **The linkUp Trap**

linkUp(3) به معنی این است که موجودیت پروتکل ارسال کننده یکی از لینک های ارتباطی ارائه شده در پیکربندی عامل فعال شده است

در trap-PDU از نوع linkup اولین عنصر variable-biding آن حاوی نام و مقدار شی ifindex برای واسط تحت تاثیر است.

**4.1.6.5 The authenticationFailure Trap**

authenticationFailure(4) به معنی این است که موجودیت پروتکل ارسال کننده مخاطب یک پیام پروتکل که به درستی احراز هویت نشده است.درحالی که پیاده سازی snmp باید قادر به تولید این trap باشد ،آنها همچنین باید قادر به سرکوب انتشار چندین trap از طریق یک مکانیزم پیاده سازی خاص باشند.

**4.1.6.6 The egpNeighborLoss Trap**

egpNeighborLoss(5) به معنی این است که همسایه egp آن کسی که موجودیت پروتکل ارسال کننده یک جفت egp بود خاموش شده است و ارتباط همکار بین آنها از دست رفته است.

Trap-PDU از نوع egpNeighborLoss اولین عنصر از variable-binding آن حاوی نام و مقدار از شی egpNeighAddr برای همسایه آسیب دیده است.

**4.1.6.7**  **The enterpriseSpecific Trap**

enterpriseSpecific(6) به معنی این است که موجودیت پروتکل ارسال کننده تشخیص دهد که برخی از رویدادهای شرکت خاص رخ داده است که فیلد تله خاص ، مشخص می کند که کدام تله خاص رخ داده است.

5. Definitions

RFC1157-SNMP DEFINITIONS ::= BEGIN

IMPORTS

ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks

FROM RFC1155-SMI;

-- top-level message

Message ::=

SEQUENCE {

version -- version-1 for this RFC

INTEGER {

version-1(0)

},

community -- community name

OCTET STRING,

data -- e.g., PDUs if trivial

ANY -- authentication is being used

}

-- protocol data units

PDUs ::=

CHOICE {

get-request

GetRequest-PDU,

get-next-request

GetNextRequest-PDU,

get-response

GetResponse-PDU,

set-request

SetRequest-PDU,

trap

Trap-PDU

}

-- PDUs

GetRequest-PDU ::=

[0]

IMPLICIT PDU

GetNextRequest-PDU ::=

[1]

IMPLICIT PDU

GetResponse-PDU ::=

[2]

IMPLICIT PDU

SetRequest-PDU ::=

[3]

IMPLICIT PDU

PDU ::=

SEQUENCE {

request-id

INTEGER,

error-status -- sometimes ignored

INTEGER {

noError(0),

tooBig(1),

noSuchName(2),

badValue(3),

readOnly(4),

genErr(5)

},

error-index -- sometimes ignored

INTEGER,

variable-bindings -- values are sometimes ignored

VarBindList

}

Trap-PDU ::=

[4]

IMPLICIT SEQUENCE {

enterprise -- type of object generating

-- trap, see sysObjectID in [5]

OBJECT IDENTIFIER,

agent-addr -- address of object generating

NetworkAddress, -- trap

generic-trap -- generic trap type

INTEGER {

coldStart(0),

warmStart(1),

linkDown(2),

linkUp(3),

authenticationFailure(4),

egpNeighborLoss(5),

enterpriseSpecific(6)

},

specific-trap -- specific code, present even

INTEGER, -- if generic-trap is not

-- enterpriseSpecific

time-stamp -- time elapsed between the last

TimeTicks, -- (re)initialization of the network

-- entity and the generation of the trap

variable-bindings -- "interesting" information

VarBindList

}

-- variable bindings

VarBind ::=

SEQUENCE {

name

ObjectName,

value

ObjectSyntax

}

VarBindList ::=

SEQUENCE OF

VarBind

END

Case, Fedor, Schoffstall, & Davin [Page 32]

RFC 1157 SNMP May 1990

1. Simple Network Management Protocol [↑](#footnote-ref-1)
2. Structure of Management Information [↑](#footnote-ref-2)
3. Management Information Base [↑](#footnote-ref-3)
4. Internet Engineering Task Force [↑](#footnote-ref-4)
5. Simple Gateway Monitoring Protocol [↑](#footnote-ref-5)
6. Agents [↑](#footnote-ref-6)
7. SNMP community [↑](#footnote-ref-7)
8. Internet Network Operations Center [↑](#footnote-ref-8)
9. OBJECT IDENTIFIER [↑](#footnote-ref-9)
10. Protocol Data Unit [↑](#footnote-ref-10)