دانشکده مهندسی کامپیوتر

ترجمه‌ی

RFC 3416

نام دانشجویان:

ملیحه عظمی

زهرا دبیری

نام درس:

مدیریت شبکه‌های کامپیوتری

استاد درس:

دکتر زینب موحدی


بسم الله الرحمن الرحيم

Request for Comments: 3416
Network Working Group
STD: 62
Obsoletes: 1905
Category: Standards Track

Editor of this version:
R. Presuhn
BMC Software, Inc.
Authors of previous version:
J. Case
SNMP Research, Inc.
K. McCloghrie
Cisco Systems, Inc.
M. Rose
Dover Beach Consulting, Inc.
S. Waldbusser
International Network Services
December2002

نسخه‌ی دوم عملیات پروتکلی برای پروتکل ساده‌ی مدیریت شبکه

وضعیت سند جاری

این سند برای جامعه‌ی اینترنت یک پروتکل ردیابی در استانداردهای شبکه را بیان می‌کند و بدین منظور از سازمان‌ها درخواست می‌کند که پیشنهادها و نظرهای خود را برای بهبود آن ارائه کنند تا روی آن‌ها بحث شود. برای آگاهی از وضعیت استاندارد سازی این پروتکل، به نسخه‌ی جاری استانداردهای پروتکلی رسمی اینترنت (STD1) رجوع کنید. انتشار این سند بدون محدودیت امکان‌پذیر است.

چکیده

این سند نسخه‌ی دوم عملیات پروتکلی پروتکل ساده‌ی مدیریت شبکه (SNMP) را توضیح می‌دهد. این سند نحو و عناصر رویه‌های ارسال، دریافت و پردازش بسته‌های SNMP را توضیح می‌دهد. سند جاری RFC 1905 را منسوخ می‌کند.

فهرست مطالب

عنوان **صفحه**

۱-۱- مقدمه

در زمان نوشتن این سند، چهارچوب مدیریتی SNMP از پنج عنصر اصلی تشکیل شده است:

- یک معماری کلی که در STD 62، RFC 3411 [RFC3411] ارائه شده است.
- سازوکارهایی برای توصیف و نام‌گذاری اشیاء و رویدادها برای اهداف مدیریتی ارائه شده‌اند. اولین نسخه از این ساختار اطلاعات مدیریتی (SMI)، SMI نسخه‌ی یک یا SMIv1 نام دارد و در STD 16، RFC 1155 [RFC1155]، STD 16، RFC 1212 [RFC1212] و RFC 1215 [RFC1215] تعریف شده است. نسخه‌ی دوم که SNMPv2 نام دارد در STD 58، RFC 2578 [RFC2578]، STD 58، RFC 2579 [RFC2579] و STD 58، RFC 2580 [RFC2580] توصیف شده است.
- پروتکل‌های پیام‌رسانی برای انتقال اطلاعات مدیریتی. اولین نسخه از پروتکل پیام‌رسانی SNMP SNMPv1 نام دارد و در STD 15، RFC 1157 [RFC1157] توضیح داده شده است. نسخه‌ی دوم از پروتکل پیام‌رسانی SNMP، SNMPv2c نام دارد و جزء پروتکل‌های ردیابی استاندارد اینترنت نیست. این نسخه در RFC 1901 [RFC1901] و STD 62 توضیح داده شده است. نسخه‌ی سوم نیز SNMPv3 نام دارد و در STD 62، RFC 3417 [RFC3417]، RFC 3412 [RFC3412] و RFC 3414 [RFC3414] به توضیح آن پرداخته شده است.
- پروتکل‌های عملیاتی برای دسترسی به اطلاعات مدیریتی. اولین مجموعه از پروتکل‌های عملیاتی و ساختار پیام‌های مرتبط با آن در STD 15، RFC 1157 [RFC1157] توصیف شده‌اند. در این سند دومین مجموعه از پروتکل‌های عملیاتی و ساختار پیام‌های مرتبط با آن توضیح داده می‌شوند.
- مجموعه‌ای از برنامه‌های کاربردی پایه در STD 62، RFC 3413 [RFC 3413] توصیف شده‌اند. در STD 62، RFC 3415 [RFC3415] نیز، مکانیزم کنترل دسترسی مبتنی بر دید[۱] توضیح داده شده است.

برای آشنایی دقیق‌تر با چهارچوب مدیریتی SNMP می‌توانید در زمان نوشتن این سند، به RFC 3415 [RFC3415] مراجعه کنید.

دسترسی به اشیاء مدیریت شده از طریق یک پایگاه اطلاعات مجازی به نام پایگاه اطلاعات مدیریتی[۲] یا MIB صورت می‌گیرد. اشیاء داخل این پایگاه با استفاده از مکانیزم‌های تعریف شده در SMI تعریف می‌شوند.

این سند (نسخه‌ی دوم از پروتکل ساده‌ی مدیریت شبکه) به عملیاتی از پروتکل می‌پردازد که مربوط به ارسال و دریافت پیام‌های منتقل شده توسط پروتکل پیام‌رسانی هستند.

---

[۱] view-based access control mechanism

[۲] Management Information Base

۲-۱- ایده‌ی اصلی

موجودیت‌های SNMP که از برنامه‌های تولیدکننده‌ی فرمان یا دریافت‌کننده‌ی اعلان پشتیبانی می‌کنند (که به صورت سنتی مدیر نامیده می‌شوند) با موجودیت‌های SNMP که از برنامه‌های پاسخ‌دهنده‌ی فرمان یا تولیدکننده‌ی اعلان پشتیبانی می‌کنند (که به صورت سنتی کارگزار[۲] نامیده می‌شوند) ارتباط برقرار می‌کنند. هدف این پروتکل انتقال اطلاعات و عملیات مدیریتی است.

۱-۲-۱- اطلاعات مدیریتی

کلمه‌ی "متغیر" به نمونه‌ای از یک نوع شئ غیرتجمعی[۳] شاره دارد که با توجه به قراردادهای بیان شده در SMI [RFC2578] یا قراردادهای متنی تعریف شده براساس SMI [RFC2579]، تعریف می‌شوند. کلمه‌ی "انقیاد متغیر[۴]" به زوج‌سازی نام یک متغیر و مقدار نظیر آن اشاره دارد. با این حال اگر حین پردازش یک درخواست بازیابی، انواع مشخصی از شرایط غیرعادی رخ دهند، یک انقیاد متغیر یک نام را با اشاره‌گری به آن وضعیت غیرعادی زوج‌سازی می‌کند.

یک لیست variable-binding، لیست ساده‌ای از انقیاد متغیرها است.

نام یک متغیر یک OBJECT IDENTIFIER است که از الحاق OBJECT IDENTIFIER نظیر object-type و یک قطعه‌ی OBJECT IDENTIFIER که نشانگر نمونه‌ی آن شئ است، ساخته می‌شود. OBJECT IDENTIFIER نظیر object-type، پیشوند OBJECT IDENTIFIER آن متغیر نام دارد.

۲-۲-۱- بازارسال درخواست‌ها

در شرایط عادی در این پروتکل، گیرنده باید برای تمام انواع درخواست‌ها پاسخی را تولید کند و به ارسال‌کننده‌ی درخواست بفرستد. تصمیم‌گیری در خصوص بازارسال پاسخ‌هایی که در محدوده‌ی زمانی مناسب دریافت نشده‌اند برعهده‌ی ارسال‌کننده‌ی درخواست است و به صورت معمول به اورژانسی بودن درخواست وابسته است. با این حال چنین برنامه‌ای باید درخصوص تعیین فرکانس و طول بازه‌ی زمانی برای بازارسال به صورتی مناسب تصمیم‌گیری کند. برای آشنایی با اصول کنترل ازدحام فوق به BCP 41 [RFC2914] مراجعه کنید.

---

[۱] notification

[۲] agent

[۳] Non-aggregate object type

[۴] Variable binding

۳-۲-۱- اندازه‌ی پیام‌ها

بزرگ‌ترین اندازه یک پیام SNMP توسط حداقل‌های زیر تعیین می‌گردد:

۱) بزرگ‌ترین اندازه پیامی که موجودیت مقصد SNMP می‌تواند بپذیرد؛ و

۲) بزرگ‌ترین اندازه پیامی که موجودیت مبدا SNMP می‌تواند تولید کند.

اولین مورد به ازای هر گیرنده تعریف می‌شود و در شرایطی که این اطلاعات در دسترس نباشد، توسط دامنه‌ی انتقال که حین ارسال پیام مورد استفاده قرار می‌گیرد، تعیین می‌گردد. دومین مورد نیز توسط محدودیت‌های محلی وابسته به پیاده‌سازی تعیین می‌شوند.

هر نگاشت انتقال برای SNMP، نشانگر حداقل اندازه‌ی پیامی است که یک پیاده سازی SNMP باید بتواند تولید یا مصرف کند. گرچه پیاده‌سازی‌ها تشویق می‌شوند تا در صورت امکان از مقادیر بزرگ‌تری از اندازه‌ی بسته‌ها پشتیبانی کنند، یک پشتیبانی سازگار نباید هیچگاه پیام‌هایی تولید کند که اندازه‌ی آن‌ها بزرگ‌تر از اندازه‌ی مجاز پیام‌ها در موجودیت SNMP دریافت‌کننده باشد.

یکی از اهداف پیام GetBulkRequest که در این پروتکل مشخص شده است، کمینه نمودن تعداد مبادله‌های پروتکلی مورد نیاز برای دریافت حجم زیادی از اطلاعات مدیریتی است. این نوع پیام به یک موجودیت SNMP که از برنامه‌های تولیدکننده‌ی فرمان پشتیبانی می‌کند، اجازه می‌دهد تا پاسخ‌هایی را درخواست کنند که اندازه‌ی آن‌ها با توجه به محدودیت در اندازه‌ی پیام‌ها بیشینه باشد. این محدودیت‌ها شامل محدودیت‌هایی روی اندازه‌ی پیام‌هایی است که موجودیت SNMP پاسخ‌دهنده می‌تواند تولید کند و اندازه‌ی پیام‌هایی که موجودیت SNMP تولیدکننده‌ی فرمان می‌تواند دریافت کند.

با این حال ممکن است این اندازه‌ی بیشینه برای پیام‌ها، بزرگتر از بزرگ‌ترین واحد قابل انتقال[۱](MTU) در مسیر پیمایش شده توسط آن‌ها باشد. در این شرایط این بسته‌ها قطعه‌بندی[۲]می‌شوند. از آن‌جا که قطعه‌بندی موجب کاهش اعتمادپذیری در انتقال پیام‌ها می شود، به صورت معمول مضر است. بنابراین یک موجودیت SNMP که پیام GetBulkRequest را ارسال می‌کند، باید پارامترهای نظیر را بگونه‌ای مقداردهی کند که از مخاطره‌ی قطعه شدن پیام جلوگیری به عمل آورد. به خصوص در شرایط دشوار برای شبکه، باید مقادیر کوچکی را برای max-repetition انتخاب نمود.

٤-۲-۱- نگاشت‌های لایه‌ی انتقال

توجه به این نکته مهم است که مبادله‌ی پیام‌های SNMP تنها به یک سرویس دیتاگرام غیرقابل اعتماد نیاز دارد که در آن، هر پیام به صورت کامل و مستقل در یک دیتاگرام قرار می‌گیرد. نگاشت‌های لایه‌ی انتقال و

---

[۱] Maximum transmission unit

[۲] fragmentation

قوانین کدگذاری[1] در [RFC3417] توضیح داده شده‌اند. با این حال، استفاده از پروتکل UDP[2] ترجیح داده می‌شود [RFC768].

۵-۲-۱- نگاشت‌های نوع داده‌ی SMIv2

SMIv2 [RFC2578]، یازده نوع پایه (INTEGER, OCTET STRING, OBJECT IDENTIFIER, INInteger32, IpAddress, Cpunter32, Gauge32, Unsigned32, TimeTicks, Opaque, Counter64) و نوع ساخت یافته‌ی BITS را تعریف می‌کند. نوع های پایه‌ی SMIv2 به نوع متناظر در انتخاب های SimpleSyntax و ApplicationSyntax از تعریف پروتکل SNMP توسط ASN.1 نگاشت می‌شوند. توجه کنید که نوع های پایه‌ی INTEGER و Integer32 از SMIv2 به نوع انتخابی integer-value از انتخاب SympleSyntax نگاشت می‌شوند. به صورت مشابه، نوع‌های پایه‌ای Gauge32 و Unsigned32 از SMIv2 به نوع انتخابی unsigned-integer-value از انتخاب ApplicationSyntax نگاشت می‌شوند.

نوع ساخت‌یافته‌ی BITS به نوع انتخابی string-value از انتخاب SimpleSyntax نگاشت می‌شود. یک مقدار BITS به صورت یک OCTET STRING کد می‌شود. این کدگذاری بدین صورت است که در آن، تمام بیت‌های نام‌دار[3] در bitstring، که از اولین بیت آغاز و تا بیت آخر ادامه می‌یابد، در بیت هشتم (بیت پرارزش) تا بیت اول (بیت کم ارزش) از اولین هشت‌تایی قرار می‌گیرند و سایر بیت‌ها به ترتیب در هشت‌تایی‌های بعدی، تا جایی که مورد نیاز است، تا آخرین هشت‌تایی قرار می‌گیرند. اگر بیت‌هایی از آخرین هشت‌تایی بدون استفاده بمانند، فرستنده به جای آن‌ها صفر می‌گذارد و گیرنده این بیت‌ها را نادیده می‌گیرد.

۳-۱- تعاریف

نحو یک PDU با استفاده از نمادگذاری ASN.1 تعریف می‌شود.

```
SNMPv2-PDU DEFINITIONS ::= BEGIN

ObjectName ::= OBJECT IDENTIFIER

ObjectSyntax ::= CHOICE {
                        simple SimpleSyntax,
                        application-wide ApplicationSyntax }

SimpleSyntax ::= CHOICE {
                        integer-value INTEGER (-2147483648..2147483647),
                        string-value OCTET STRING (SIZE (0..65535)),
                        objectID-value OBJECT IDENTIFIER }

ApplicationSyntax ::= CHOICE {
```

---

[1] encoding

[2] User Datagram Protocol

[3] Named bits

```
                                ipAddress-value IpAddress,
                                counter-value Counter32,
                                timeticks-value TimeTicks,
                                arbitrary-value Opaque,
                                big-counter-value Counter64,
                                unsigned-integer-value Unsigned32 }

IpAddress ::= [APPLICATION 0] IMPLICIT OCTET STRING (SIZE (4))


Counter32 ::= [APPLICATION 1] IMPLICIT INTEGER (0..4294967295)

Unsigned32 ::= [APPLICATION 2] IMPLICIT INTEGER (0..4294967295)

Gauge32 ::= Unsigned32

TimeTicks ::= [APPLICATION 3] IMPLICIT INTEGER (0..4294967295)

Opaque ::= [APPLICATION 4] IMPLICIT OCTET STRING

Counter64 ::= [APPLICATION 6]
              IMPLICIT INTEGER (0..18446744073709551615)

-- protocol data units

PDUs ::= CHOICE {
                  get-request       GetRequest-PDU,
                  get-next-request  GetNextRequest-PDU,
                  get-bulk-request  GetBulkRequest-PDU,
                  response          Response-PDU,
                  set-request       SetRequest-PDU,
                  inform-request    InformRequest-PDU,
                  snmpV2-trap       SNMPv2-Trap-PDU,
                  report            Report-PDU }
-- PDUs

GetRequest-PDU ::= [0] IMPLICIT PDU

GetNextRequest-PDU ::= [1] IMPLICIT PDU

Response-PDU ::= [2] IMPLICIT PDU

SetRequest-PDU ::= [3] IMPLICIT PDU

-- [4] is obsolete

GetBulkRequest-PDU ::= [5] IMPLICIT BulkPDU


InformRequest-PDU ::= [6] IMPLICIT PDU

SNMPv2-Trap-PDU ::= [7] IMPLICIT PDU

-- Usage and precise semantics of Report-PDU are not defined
-- in this document. Any SNMP administrative framework making
-- use of this PDU must define its usage and semantics.


Report-PDU ::= [8] IMPLICIT PDU

max-bindings INTEGER ::= 2147483647

PDU ::= SEQUENCE {
        request-id INTEGER (-214783648..214783647),
```

```
        error-status          -- sometimes ignored
            INTEGER {
                  noError(0),
                  tooBig(1),
                  noSuchName(2),    -- for proxy compatibility
                  badValue(3),      -- for proxy compatibility
                  readOnly(4),      -- for proxy compatibility
                  genErr(5),
                  noAccess(6),
                  wrongType(7),
                  wrongLength(8),
                  wrongEncoding(9),
                  wrongValue(10),
                  noCreation(11),
                  inconsistentValue(12),
                  resourceUnavailable(13),
                  commitFailed(14),
                  undoFailed(15),
                  authorizationError(16),
                  notWritable(17),
                  inconsistentName(18)
      },

error-index             -- sometimes ignored
      INTEGER (0..max-bindings),
variable-bindings       -- values are sometimes ignored
VarBindList
}

BulkPDU ::=             -- must be identical in
SEQUENCE {              -- structure to PDU
request-id        INTEGER (-214783648..214783647),
non-repeaters     INTEGER (0..max-bindings),
max-repetitions   INTEGER (0..max-bindings),
variable-bindings       -- values are ignored
VarBindList
}

-- variable binding

VarBind ::= SEQUENCE {
            name ObjectName,

            CHOICE {
            value       ObjectSyntax,
            unspecified       NULL,       -- in retrieval requests

                                          -- exceptions in responses
            noSuchObject      [0] IMPLICIT NULL,
            noSuchInstance    [1] IMPLICIT NULL,
            endOfMibView      [2] IMPLICIT NULL
      }
}
-- variable-binding list

VarBindList ::= SEQUENCE (SIZE (0..max-bindings)) OF VarBind

END
```

۴-۱- ویژگی‌های پروتکل

۱-٤-۱- ساختارهای رایج

مقدار فیلد request-id در پیام پا سخ، م ساوی مقدار فیلد request-id در پیام درخوا ست متناظر این پا سخ است. یک برنامه می‌تواند با استفاده از این فیلد، درخواست‌های منتظر را از هم تشخیص دهد و به این ترتیب پاسخ‌های دریافتی نظیر هر یک از درخواست‌ها را بیابد. در شرایطی که یک سرویس دیتاگرام غیرقابل اعتماد اســتفاده شــود، request-id برای تشــخیص پیام‌هایی که دومرتبه ارســال شــده‌اند، بکار می‌رود. اســتفاده از request-id م شابه در بازار سال یک درخوا ست نیز این امکان را می‌دهد که پا سخ بسته‌ی ار سال شده یا باز ار سال شده نیاز درخوا ست ار سال شده را برآورده کنند. با این حال برای محا سبه‌ی زمان رفت و برگ شت (RTT)[1]برای انتقال و پردازش یک تراکنش درخواســت- پاســخ، این برنامه باید از مقدار request-id متفاوتی برای پیام بازارسالی استفاده کند. این راهبرد برای استفاده در اغلب شرایط بکار می‌رود.

مقدار ناصــفر فیلد error-status در پیام پاســخ نشــان‌دهنده‌ی وقوع خطا در بســته‌ی ارســالی اســت که مانع پردازش آن شده ا ست. در این شرایط، مقدار نا صفر فیلد error-index در پیام پا سخ نشان می‌دهد که کدام انقیاد متغیر در یک لیســت موجب خطا شــده اســت. هر انقیاد متغیر توســط مقدار شــاخص[2]آن شــناســایی می‌شود. اولین شاخص در یک انقیاد متغیر یک، دومین دو و بقیه به همین ترتیب هستند.

SNMP مقادیر OBJECT IDENTIFIER را به حداکثر صد و بیست و هشت زیرشاخص[3]محدود می‌کند که حداکثر مقدار هر زیرشاخص 1-32**2 است.

۲-٤-۱- پردازش پیام

در عناصــر رویه‌ی زیر، تمامی فیلدهای یک پیام که توســط رویه‌ی فوق مورد ارجاع قرار نگرفته‌اند، توســط موجودیت SNMP دریافت‌کننده نادیده گرفته می‌شــوند. با این حال، تمامی اجزای یک پیام، شــامل آن‌هایی که مقادیر شان توسط موجودیت دریافت‌کننده نادیده گرفته می شوند، باید از نحو و کدگذاری معتبر ASN.1 پشــتیبانی کنند. برای مثال در برخی پیام‌ها (برای مثال GetRequest) تنها نام متغیرها و نه مقدار آن‌ها مهم اســت. در این شــرایط، بخش مقدار در انقیاد متغیر توســط موجودیت SNMP دریافت‌کننده نادیده گرفته می‌شود. مقدار unspecified برای فیلد مقدار در چنین انقیادهایی در نظر گرفته می‌شود.

در برقراری یک ارتباط مدیریتی، wrapper پیام‌ها برای کپ سوله کردن PDU با ا ستفاده از "عنا صر رویه" ی چهارچوب مدیریتی مورد ا ستفاده ساخته می شود. تعریف "max-bindings" یک کران بالا برای تعداد انقیاد

---

[1] Round Trip Time

[2] index

[3] sub-idenifier

متغیرها تعریف می‌کند. همچنین اندازه‌ی یک پیام توسط محدودیت‌های روی بیشینه اندازه‌ی پیام‌ها محدود می‌شـــود. یک پیاده‌ســـازی ســـازگار باید در هر پیام PDU یا BulkPDU حداکثر تعداد متغیرها را با رعایت محدودیت اندازه درSNMP مقداردهی کند. اما نباید بیش از ۲۱۴۷۴۸۳۶۴۷ انقیاد متغیر را انجام دهد.

در دریافت یک ارتباط مدیریتی، "عناصـر رویه"ی چهارچوب مدیریتی مورد اسـتفاده پیروی می‌شـوند و اگر این رویه‌ها نشـــان دهند که عملیاتی که در این پیام قرار دارند باید به صـــورت محلی اجرا گردند، این رویه‌ها نشانگر MIB view ی قابل مشاهده برای عملیات هستند.

۳-٤-۱- پیام GetRequest

یک پیغام GetRequest با درخواست یک برنامه تولید و منتقل می‌گردد.

موجودیت SNMP دریافت‌کننده با دریافت یک پیغام GetRequest، هر انقیاد متغیر در یک لیست از انقیادها را پردازش می‌کند تا یک پیام پاسخ تولید نماید. مقادیر تمامی فیلدهای پیام پاسخ با پیام درخواست دریافتی یکسان است و تنها در فیلدهای زیر با هم متفاوتند. هر انقیاد متغیر به صورت زیر پردازش می‌شود:

۱) اگر نام استفاده شده در انقیاد متغیر دقیقا با نام متغیر قابل دسترسی با این درخواست یکسان باشد، فیلد مقدار در انقیاد متغیر به متغیر نام‌برده داده می‌شود.

۲) در غیر اینصـــورت، اگر پیشـــوند OBJECT IDENTIFIER نام اســـتفاده شـــده در انقیاد متغیر، با پیشوند OBJECT IDENTIFIER هر متغیر قابل دسترسی در این درخواست دقیقا یکسان نباشد، مقدار این فیلد به "noSuchObject" تغییر می‌یابد.

۳) در غیر اینصورت، فیلد مقدار در انقیاد متغیر به "noSuchInstance" تنظیم می‌شود.

اگر پردازش یک انقیاد متغیر به دلایلی به جز موارد اشاره شده در بالا با شکست مواجه شود، یک پیام پاسخ با فرمت جدید ساخته می‌شود که در آن، فیلدهای request-id و variable-binding مقادیر نظیر این فیلدها در پیام GetRequest را دارند و مقدار فیلد error-status به "genErr" تغییر می‌یابد و مقدار فیلد error-index نیز به شاخص انقیاد متغیر شکست‌خورده داده می‌شود.

در غیر اینصـــورت، مقدار فیلد error-status در پیام پاســـخ به "noError" تغییر می‌یابد و فیلد error-index نیز مقدار صفر می‌گیرد.

پیام پاسـخ تولید شـده در یک پیام کپسـوله می‌شـود. اگر اندازه‌ی پیام حاصـل کوچک‌تر یا مسـاوی با هردو محدودیت محلی و بزرگترین اندازه پیام پشـــتیبانی شـــده توســـط تولیدکننده باشـــد، به تولیدکننده‌ی پیام GetRequest ارسال می‌شود.

در غیر اینصـــورت، یک پیام پاســـخ جایگزین ســـاخته می‌شـــود. این پیام مقدار فیلد request-id در پیام GetRequest دریافتی را در فیلد request-id خود دارد. همچنین مقدار فیلد error-status آن "tooBig"، مقدار فیلد errorIndex آن صفر و فیلد variable-bindings آن خالی است. این پیام پاسخ جایگزین در یک

پیام دیگر کپسوله می‌شود. درصورتیکه اندازه‌ی پیام حاصل کوچک‌تر یا مساوی هردو محدودیت‌های محلی و اندازه‌ی بزرگ‌ترین پیام پشتیبانی شده توسط تولید کننده باشد، به تولیدکننده‌ی پیام GetRequest ارسال می شود. در غیر اینصورت شمارنده‌ی snmpSilenceDrops [RFC3418] افزایش می‌یابد و پیام حاصل دور انداخته می‌شود.

٤-٤-١- پیام GetNextRequest

یک پیام GetNextRequest با درخواست یک برنامه تولید و منتقل می‌شود.

با دریافت یک پیام GetNextRequest، موجودیت SNMP دریافت کننده هر انقیاد متغیر در لیست variable-binding را پردازش می‌کند تا یک پیام پاسخ تولید کند. تمامی فیلدهای پیام پاسخ مقادیر فیلدهای پیام دریافت شده را دارند و تنها در موارد زیر با یکدیگر متفاوتند. هر انقیاد متغیر به صورت زیر پردازش می‌شود:

۱) مکان متغیر در لیستی از نام‌های تمام متغیرها که با ترتیب حرف به حرف مرتب گشته‌اند، مشخص می‌گردد. می‌توان با این درخواست به این متغیر دسترسی یافت و نام آن اولین نام در ترتیب حرف به حرف پس از نام ذکر شده در انقیاد متغیر در پیام GetNextRequest دریافتی است. فیلدهای نام و مقدار در پیام پاسخ، نام و مقدار متغیر فوق را می‌گیرند.

۲) اگر نام متغیر درخواست شده در انقیاد متغیر، پیش از نام هیچ متغیر دیگر قابل دسترس با این درخواست در ترتیب حرف به حرف نباشد، یعنی هیچ نام دیگری پس از آن در ترتیب حرف به حرف نیامده باشد، انقیاد متغیر فوق که در پیام پاسخ صورت می‌گیرد دارای فیلد مقدار با مقدار "endOfMibView" است و فیلد نام در آن به نام انقیاد متغیر در پیام درخواست، تنظیم می‌شود.

۳) اگر پردازش یک انقیاد متغیر به دلیلی خارج از موارد بالا با شکست مواجه شود، پیام پاسخ دارای فیلدهای request-id و variable-binding یکسانی با پیام GetNextRequest دریافتی است. مقدار فیلد error-status آن "genErr" و مقدار فیلد error-index آن به شاخص انقیاد متغیر خطا خورده تنظیم می‌شود.

در غیر اینصورت مقدار فیلد error-status در پیام پاسخ به "noError" و فیلد error-index نیز به صفر تنظیم می‌شوند.

سپس پیام پاسخ تولید شده در یک پیام کپسوله می شود. اگر اندازه‌ی پیام حاصل کوچک‌تر یا مساوی هردو محدودیت‌های محلی و بیشینه‌ی اندازه‌ی پیام پشتیبانی شده توسط تولیدکننده باشد، پیام پاسخ به تولیدکننده‌ی پیام GetNextRequest ارسال می‌گردد.

در غیراینصورت، یک پیام پاسخ جایگزین ساخته می‌شود. این پیام دارای فیلد request-id یکسان با پیام GetNextRequest دریافتی است و مقدار فیلد error-status آن "tooBig"، مقدار فیلد error-index

آن صفر و فیلد variable-bindings آن خالی است. پیام پاسخ جایگزین سپس در یک بسته کپسوله می‌شود. اگر اندازه‌ی پیام حاصل کوچک‌تر یا مساوی هردو محدودیت‌های‌های محلی و بیشینه اندازه‌ی یک پیام در تولیدکننده باشد، این پیام به تولیدکننده‌ی پیام GetNextRequest ارسال می‌گردد. در غیر اینصورت، شمارنده‌ی snmpSilentDrops [RFC3418] یک واحد افزایش می‌یابد و پیام حاصل دور انداخته می‌شود.

◘ مثالی از پیمایش جدول

یک کاربرد مهم از پیام GetNextRequest، پیمایش جدول‌های اطلاعاتی متصور[1] در یک MIB است. نحو این نوع درخواست همراه با روش شناسایی نمونه‌های منفرد از اشیاء در MIB، دسترسی به اشیاء فوق از MIB که به صورت جدولی سازمان‌دهی شده‌اند را فراهم می‌کند.

در تبادل پروتکلی نمایش داده شده در زیر، یک برنامه آدرس فیزیکی وابسته به رسانه و نوع نگاشت آدرس برای هر سطر از جدول ترجمه‌ی آدرس IP شبکه- رسانه [RFC1213] نظیر یک عنصر شبکه مشخص را بدست می‌آورد. این برنامه همچنین مقدار sysUpTime [RFC3418] که نگاشت‌ها در آن وجود داشتند را بدست می‌آورد. فرض کنید جدول آدرس IP- رسانه عنصر پاسخ‌دهنده‌ی فرمان سه سطر داشته باشد:

شماره واسط	آدرس شبکه	آدرس فیزیکی	نوع
1	10.0.0.51	00:00:10:01:23:45	static
1	9.2.3.4	00:00:10:54:32:10	dynamic
2	10.0.0.15	00:00:10:98:76:54	dynamic

موجودیت SNMP که از یک برنامه‌ی تولیدکننده‌ی فرمان پشتیبانی می‌کند، کار خود را با ارسال یک پیام GetNextRequest که شامل مقادیر OBJECT IDENTIFIER نظیر نام متغیرهای درخواستی است، شروع می‌کند.

```
GetNextRequest ( sysUpTime,
                 ipNetToMediaPhysAddress,
                 ipNetToMediaType )
```

موجودیت SNMP که از یک برنامه‌ی پاسخگوی فرمان پشتیبانی می‌کند یک پیام پاسخ به صورت زیر ارسال می‌کند:

```
Response (( sysUpTime.0 = "123456" ),
            ( ipNetToMediaPhysAddress.1.9.2.3.4 = "000010543210" ),
            ( ipNetToMediaType.1.9.2.3.4 = "dynamic" ))
```

[1] Conceptual

موجودیت SNMP که از یک برنامه‌ی تولیدکننده‌ی فرمان پشتیبانی می‌کند به صورت زیر ادامه می‌دهد:

```
GetNextRequest ( sysUpTime,
            ipNetToMediaPhysAddress.1.10.0.0.51,
            ipNetToMediaType.1.10.0.0.51 )
```

موجودیت SNMP که از یک برنامه‌ی پاسخگوی فرمان پشتیبانی می‌کند با پیام زیر پاسخ می‌دهد:

```
Response (( sysUpTime.0 = "123466" ),
            ( ipNetToMediaPhysAddress.2.10.0.0.15 = "000010987654" ),
            ( ipNetToMediaType.2.10.0.0.15 = "dynamic" ))
```

موجودیت SNMP که از برنامه‌ی تولیدکننده‌ی فرمان پشتیبانی می‌کند با ارسال پیام زیر ادامه می‌دهد:

```
GetNextRequest ( sysUpTime,
               ipNetToMediaPhysAddress.2.10.0.0.15,
               ipNetToMediaType.2.10.0.0.15 )
```

از آنجا که سطرهای بی شتری در جدول وجود ندارند، موجودیت SNMP پا سخگوی فرمان با متغیرهایی که بعد از نام اشیاء قابل دسترسی در ترتیب حرف به حرف می‌آیند، پاسخ می‌دهد. برای مثال:

```
Response (( sysUpTime.0 = "123471" ),
          ( ipNetToMediaNetAddress.1.9.2.3.4 = "9.2.3.4" ),
          ( ipRoutingDiscards.0 = "2" ))
```

توجه کنید که چگونه با رسیدن به انتهای ستون ipNetToMediaPhysAddress، دومین انقیاد متغیر از برنامه‌ی پاسخگوی فرمان، در اولین سطر از ستون بعدی پیچیده شده است.

همچنین توجه کنید که چگونه با رسیدن به انتهای ipNetToMedia برای سومین انقیاد متغیر، برنامه‌ی پاسخگوی فرمان با شئ موجود بعدی پاسخ داده است که این شئ خارج از جدول است. این پاسخ به برنامه‌ی تولیدکننده‌ی فرمان اطلاع می‌دهد که جدول به پایان رسیده است.

۵-٤-۱- پیام GetBulkRequest

یک پیام GetBulkRequest بنابر درخواست یک برنامه‌ی کاربردی تولید می‌شود و منتقل می‌گردد. هدف این پیام درخواست نمودن انتقال حجم زیادی از داده، شامل (و نه محدود به) درخواست سریع و کارآمد جداول بزرگ است.

با دریافت یک پیام GetBulkRequest، موجودیت SNMP دریافت‌کننده، هر انقیاد متغیر در لیست variable-binding را پردازش می‌کند تا یک پیام پاسخ با مقدار فیلد request-id مشابه پیام درخواست تولید کند.

برای نوع پیام GetBulkRequest، پردازش موفقیت‌آمیز هر انقیاد متغیر در درخواست، صفر یا تعداد بیشتری انقیاد متغیر در پیام پاسخ تولید می‌کند. به این ترتیب وضعیت نگاشت یک به یک بین انقیاد متغیرها در پیام GetRequest، GetNextRequest و SetRequest و پیام‌های پاسخ حاصل برای نگاشت بین انقیاد متغیرهای پیام GetBulkRequest و پیام پاسخ حاصل برقرار نیست.

مقادیر فیلدهای non-repeaters و non-repetitions در پیام درخواست، پردازش مورد درخواست را مشخص می‌کنند. در پیام پاسخ برای N انقیاد متغیر اول در پیام درخواست یک پیام و برای هر یک از R انقیاد متغیر

باقی‌مانده در درخواســـت، M انقیاد متغیر درخواســـت می‌شـــود. به این ترتیب تعداد کل انقیاد متغیرهای درخواست شده توسط N+M*R نشان داده می‌شود و در آن، N کمترین میان ۱) مقدار فیلد non-repeaters در پیام درخواست و ۲) تعداد انقیاد متغیرها در درخواست، M مقدار فیلد max-repetitions در درخواست و R بزرگ‌ترین میان ۱) تعداد انقیاد متغیرها در پیام درخواست – N و ۲) صفر است.

موجودیت SNMP دریافت‌کننده یک پیام پاســخ با حداکثر تعداد کل انقیاد متغیرهای درخواســت شــده که توســط پیام درخواســت تبادل شــده اســت، می‌ســازد. Request-id پیام پاســخ باید با request-id پیام GetBulkRequest دریافتی یکسان باشد. اگر N بزرگ‌تر از صفر باشد، اولین تا Nامین انقیاد متغیر در پیام پاسخ به صورت زیر تولید می‌شوند:

۱) یک متغیر مکان‌یابی می شود. این کار بدین صورت است که در یک لیست مرتب شده از نام تمام متغیرهای قابل دسترسی توسط این درخواست به صورت حرف به حرف[۱] مکان آن مشخص می‌گردد و نام آن اولین کلمه بعد از نام انقیاد متغیر در پیام GetBulkRequest در ترتیب حرف به حرف است. فیلدهای نام و مقدار در انقیاد متغیر در پیام پاسخ به نام و مقدار متغیر مکان‌یابی شده تنظیم می‌شود.

۲) اگر نام انقیاد متغیر درخواســت شــده پیش از نام هیچ یک از متغیرهای قابل دســترســی توســط این درخواست نیامده باشد، در واقع هیچ نام دیگری در ترتیب حرف به حرف پس از این نام نیامده باشد، انقیاد متغیر تولید شده در پیام پاسخ مقدار فیلد مقدارش به "endOfMibView" و مقدار فیلد نام آن به نام در انقیاد متغیر در پیام درخواست تنظیم می‌شود.

اگر M و N ناصفر باشند، (N+1) امین و نیز انقیاد متغیرهای بعدی در پیام پاسخ به صورت مشابه تولید می‌گردند. برای هر تکرار i، به طوریکه i بزرگ‌تر یا مساوی صفر و کوچک‌تر یا مساوی M است و برای هر متغیر تکراری r که بزرگ‌تر یا مساوی صفر و کوچک‌تر یا مساوی R است، (N+((i-1)*R)+r)امین انقیاد متغیر در پیام پاسخ به صورت زیر تولید می‌گردد:

۱) متغیری که در لیستی از نام‌های تمام متغیرهای قابل دسترسی توسط این درخواست که به صورت حرف به حرف مرتب شده‌اند، مکان‌یابی می‌شود. نام این متغیر iامین بعدی در ترتیب حرف به حرف پس از (N+r)امین نام انقیاد متغیر در پیام GetBulkRequest دریافتی اســـت. فیلدهای نام و مقدار انقیاد متغیر به نام و مقدار متغیر مکان‌یابی شده تنظیم می‌شود.

۲) اگر در ترتیب حرف به حرف هیچ نامی پس از iامین نام وجود نداشته باشد، انقیاد متغیر متناظر که در پیام پاسخ آمده است، دارای فیلد مقدار برابر با "endOfMibView" و فیلد نام مساوی با آخرین نام بعدی در ترتیب حرف به حرف و یا در صـــورت وجود نداشـــتن هیچ بعدی‌ای در ترتیب حرف به

---

[۱] lexicographic

حرف، (N+1)امین نام در انقیاد متغیر در پیام درخواست است.

گرچه بیشنیه تعداد انقیاد متغیرها در پیام پاسخ توسط N+(M*R) محدود می شود، ممکن است پیام پاسخ با تعداد کمتری از انقیاد متغیرها که صفر نیز می‌تواند باشد، تولید گردد. علت این مسئله هر یک از سه مورد زیر می‌تواند باشد:

۱) اگر اندازه‌ی پیام پاسخ پیچنده‌ی پیام پاسخ که شامل تعداد درخواستی از انقیاد متغیرها است بزرگ‌تر یا مساوی محدودیت محلی یا بیشترین اندازه‌ی پیام‌های تولیدکننده باشد، پیام پاسخ با تعداد کمتری از انقیاد متغیرها ساخته می شود. این تعداد کمتر در حقیقت یک مجموعه‌ی مرتب از انقیاد متغیرها است که تعدادی از انقیاد متغیرها در انتهای مجموعه از آن حذف شده‌اند، به طوریکه اندازه‌ی پیامی که پیام پاسخ را می‌پیچد به صورت تقریبی مساوی اما نه بزرگ‌تر محدودیت محلی یا محدودیت روی بیشینه اندازه‌ی پیام در موجودیت تولیدکننده باشد. دقت کنید که تعداد انقیاد متغیرهای حذف شده هیچ ارتباطی با مقادیر N، M یا R ندارد.

۲) اگر برای برخی مقادیر تکرار i، بطوریکه i بزرگ‌تر از صفر یا کوچک‌تر یا مساوی M است، تمامی فیلدهای انقیاد متغیر تولید شده دارای مقدار "endOfMibView" باشند، پاسخ نیز می‌تواند با تعداد کم‌تری انقیاد متغیر ساخته شود. در این شرایط، انقیاد متغیرها ممکن است پس از ((N+(i*R))امین انقیاد متغیر صورت گیرد.

۳) در شرایطی که پردازش یک درخواست که تعداد زیادی repetitions دارد، مستلزم زمان پردازش بسیار بیشتری نسبت به یک درخواست عادی باشد، یک برنامه‌ی پاسخگوی فرمان ممکن است به پیام درخواست را با تعداد repetitionهای کمتری پردازش کند. البته با این شرط که حداقل یک repetition را کامل کند.

اگر پردازش یک انقیاد متغیر به دلیلی خارج از موارد لیست بالا متوقف شود، پیام پاسخ با فرمت جدید ساخته می‌شود که فیلدهای request-id و variable-bindings آن برابر با مقدار این فیلدها در پیام درخواست شده هستند. مقدار فیلد error-status نیز به "genErr" و مقدار فیلد error-index به شاخص انقیاد متغیر در درخواست اصلی تنظیم می‌شود که متناظر با انقیاد متغیرهای شکست خورده است.

در غیر اینصورت، مقدار فیلد error-status در پیام پاسخ به "noError" و مقدار فیلد error-index به صفر تنظیم می‌شود.

پیام پاسخ تولید شده (که ممکن است فیلد variable-bindings آن خالی باشد) در یک پیام کپسوله می‌شود. اگر اندازه‌ی پیام حاصل کوچک‌تر یا مساوی هردو محدودیت محلی و اندازه‌ی بزرگ‌ترین پیام پشتیبانی شده توسط تولیدکننده باشد، به تولیدکننده‌ی پیام GetBulkRequest ارسال می‌گردد. در غیر اینصورت، شمارنده‌ی snmpSilentDrops [RFC3418] افزایش می‌یابد و پیام حاصل دور انداخته

می‌شود.

◘ یک مثال دیگر از جدول Traversal

این مثال نشـــان می‌دهد که چگونه پیام GetBulkRequest می‌تواند به عنوان جایگزینی برای پیام GetNextRequest استفاده شود. بدین ترتیب همان پیمایش جدول IP شبکه- رسانه که در بخش قبل نشان داده شد، با تعداد تبادل‌های کمتر بدست می‌آید.

موجودیت SNMP که از برنامه تولیدکننده فرمان پشتیبانی می کند با ارسال یک پیام GetBulkRequest با مقدار max-repetition دو، و شـامل مقادیر OBJECT IDENTIFIER که به صـورت نام متغیر درخواسـت شده نشن داده شده است، کار خود را آغاز می‌کند.

```
GetBulkRequest [ non-repeaters = 1, max-repetitions = 2 ]
        ( sysUpTime,
          ipNetToMediaPhysAddress,
          ipNetToMediaType )
```

موجودیت SNMP که از برنامه پاسخگوی فرمان پشتیبانی می‌کند با پیام Response پاسخ می دهد:

```
Response (( sysUpTime.0 =  "123456" ),
        ( ipNetToMediaPhysAddress.1.9.2.3.4 = "000010543210" ),
      ( ipNetToMediaType.1.9.2.3.4 =  "dynamic" ),
        ( ipNetToMediaPhysAddress.1.10.0.0.51 = "000010012345" ),
      ( ipNetToMediaType.1.10.0.0.51 =  "static" ))
```

موجودیت SNMP که از برنامه تولیدکننده‌ی فرمان پشتیبانی می کند، بدین ترتیب ادامه می‌دهد:

```
GetBulkRequest [ non-repeaters = 1, max-repetitions = 2 ]
          ( sysUpTime,
            ipNetToMediaPhysAddress.1.10.0.0.51,
            ipNetToMediaType.1.10.0.0.51 )
```

موجودیت SNMP که از برنامه‌ی پاسخگوی فرمان پشتیبانی می‌کند به صورت زیر پاسخ می‌دهد :

```
Response (( sysUpTime.0 =  "123466" ),
        ( ipNetToMediaPhysAddress.2.10.0.0.15 = "000010987654" ),
        ( ipNetToMediaType.2.10.0.0.15 = "dynamic" ),
        ( ipNetToMediaNetAddress.1.9.2.3.4 = "9.2.3.4" ),
        ( ipRoutingDiscards.0 =  "2" ))
```

توجه کنید که چگونه (همانند مثال اول) انقیاد متغیرها در پیام پاســـخ نشـــان می دهد که جدول به پایان رسیده است. چهارمین انقیاد متغیر، با برگرداندن اطلاعات از ستون موجود بعدی انجام می شود؛ پنجمین انقیاد متغیر، با بازگشـــت اطلاعات از اولین شـــیء موجود که به ترتیب حرف به حرف در جدول قرار گرفته است، انجام می شود. این پاسخ، رسیدن به انتهای جدول را به برنامه تولیدکننده‌ی فرمان اطلاع می‌دهد.

پیام Response

پیام Response توسـط یک موجودیت SNMP تنها پس از دریافت پیام GetRequest، GetNextRequest ، GetBulkRequest ، SetRequest یا InformRequest تولید می شود، همانطور که در جاهای دیگر این سند توضیح داده شده است.

اگر فیلد error-status در پیام Response غیر صفر باشد، فیلدهای مقدار مربوط به متغیر در لیست انقیاد متغیرها، نادیده گرفته می شوند.

اگر هر دو فیلد error-status و فیلد error-index در پیام Response غیر صفر با شند، سپس مقدار فیلد error-index، شاخص متغیری در لیست variable-binding درخواست نظیر است که درخواست آن شکست خورده است. اولین انقیاد متغیر در لیست variable-binding درخواست، شاخص اول را دارد، دومین شاخص دوم را دارد و به همین ترتیب.

یک موجودیت SNMP که از یک برنامه‌ی تولیدکننده‌ی فرمان پشتیبانی می کند باید قادر باشد به درستی یک پیام Response را با فیلد error-status برابر با "noSuchName " ، "badValue" یا "readOnly" اداره کند. (به بخش های ۱,۳ و ۴,۳ [RFC2576] مراجعه کنید.)

با دریافت یک پیام Response، موجودیت SNMP دریافت‌کننده، محتویات خود را به برنامه ای که این درخواست را با همان مقدار request-id ایجاد کرده است، ارائه می دهد. برای جزئیات بیشتر، [RFC3412] را ببینید.

٦-٤-١- پیام SetRequest

یک پیام به درخواست یک برنامه SetRequest تولید و ارسال می شود.

پس از دریافت یک پیام SetRequest، موجودیت SNMP دریافت کننده اندازه‌ی پیامی که پیام Response را کپسوله می‌کند، مشخص می‌کند. فیلدهای request-id و variable-binding در پیام کپسوله‌کننده برابر با مقدار این فیلدها در پیام SetRequest دریافتی است. همچنین بزرگ‌ترین اندازه‌ی ممکن برای فیلدهای error-status و error-index برابر با بزرگ‌ترین اندازه‌ی ممکن در پیام SetRequest دریافتی است.. اگر اندازه پیام مشخص شده بیشتر از محدودیت محلی یا حداکثر اندازه پیام پشتیبانی شده توسط مبدأ باشد، سپس یک پیام Response جایگزین تولید می شود، و به فرستنده‌ی پیام SetRequest فرستاده می شود و پردازش پیام SetRequest بلافاصله پس از آن متوقف می شود . این پیام Response دارای فیلد request-id یکسان با پیام اصلی است، مقدار فیلد error-status به "tooBig"، مقدار error-index به صفر تنظیم می‌شوند و فیلد variable-bindings خالی می‌ماند. اگر اندازه پیام حاصل کمتر یا برابر محدودیت محلی و حداکثر اندازه پیام اصلی باشد، به فرستنده پیام SetRequest ارسال می شود. در غیر اینصورت، شمارنده snmpSilentDrops [RFC3418] افزایش می یابد و پیام حاصل از آن حذف می شود. صرفنظر از این، پردازش پیام SetRequest نیز متوقف می شود.

در غیر این صورت، موجودیت دریافت کننده SNMP، هر variable-binding را در لیست variable-binding ، برای تولید پیام Response پردازش می کند. همه فیلدهای پیام Response مقادیر مشابهی با فیلدهای متفاوتی از درخواست دریافت شده دارند به غیر از موارد زیر:

variable-binding به صـــورت مفهومی به عنوان یک عملیات دو مرحله ای پردازش می شـــوند. در مرحله اول، هر variable-binding تایید می شـود؛ اگر تمام اعتبارســنجی ها موفق باشــند، هر متغیر در مرحله دوم تغییر می یابد. البته، برای اجرای اولین یا دوم یا هر دو این مراحل مفهومی به عنوان مراحل تکامل چندگانه آزادی عمل وجود دارد. در واقع، در بعضی موارد، ممکن است مراحل تکامل چندگانه برای اطمینان از انسجام لازم باشد.

اعتبار سنجی های زیر در مرحله اول بر روی هر variable-binding تا زمانی که همه موفق شوند یا تا زمانی که یک شکست نباشد اجرا می شوند:

۱) اگر نام variable-binding نام یک متغیر موجود یا غیر موجود را تعیین کند این درخواست / دسترسی ممنوع است زیرا در MIB نمی باشد، سپس مقدار فیلد error-status در پیام Response برابر "noAccess" می شود ، و مقدار فیلد index-index آن به عنوان متغیر ناموفق نامحدود تنظیم خواهد شد.

۲) در غیر این صورت، اگر هیچ متغیری وجود نداشته باشد که یک پیشوند OBJECT IDENTIFIER همانند نام متغیر را داشته باشد و بتواند بدون توجه به مقدار جدید تعیین شده ایجاد یا اصلاح شود، مقدار فیلد error-status برابر notWritable تنظیم می شود ، و مقدار فیلد index-index آن به نام متغیر fail شده تعلق می‌گیرد.

۳) در غیر این صورت، اگر فیلد variable binding، مطابق با زبان ASN.1، نوعی است که با آن مورد نیاز برای تمام متغیرهایی که یک پیشوند OBJECT IDENTIFIER را همانند نام متغیر مورد استفاده قرار می دهند، مشخص می کند، سپس مقدار پیام Response فیلد وضعیت error-state به "wrongType" تنظیم شده است و مقدار فیلد index-index آن به عنوان متغیر ناموفق نامحدود تنظیم شده است.

۴) درغیر اینصورت، اگر فیلد مقدار متغیر variable binding براساس زبان ASN.1 مشخص شود، طولی که با مقدار مورد نیاز برای همه متغیرهایی که همان پیشوند OBJECT IDENTIFIER را به عنوان متغیر binding's name به اشتراک میگذارند، متناقض هستند، سپس مقدار فیلد وضعیت خطای Response-PDU's به مقدار "wrongLength" تنظیم میشود، و مقدار فیلد error-index آن به index فیلد variable binding تنظیم می‌شود.

۵) در غیر این صورت، اگر فیلد variable binding حاوی کدگذاری ASN.1 باشد که با برچسب ASN.1 آن فیلد ناسازگار باشد، مقدار فیلد error-status در پیام Response روی «errorEncoding» تنظیم شده و مقدار فیلد error-index آن به index فیلد variable binding متغیر fail شده تنظیم می شود. (توجه داشته باشید که تمام استراتژی های پیاده سازی این خطا را تولید نمی کند.)

۶) در غیر این صورت، اگر فیلد value-binding یک متغیر که در هیچ شرایطی نمی تواند به متغیر اختصاص داده شود به متغیر داده شود، مقدار فیلد error-status در پیام Response بر روی «wrongValue» تنظیم شده و مقدار فیلد error-index آن برابر index مربوط به انقیاد متغیر شکست خورده تنظیم شده می‌شود.

۷) در غیر این صورت، اگر نام variable binding متغیری را ایجاد کند که وجود ندارد و هرگز نمی تواند ایجاد شود (حتی اگر برخی از متغیرهایی که یک پیشوند OBJECT IDENTIFIER به اشتراک گذاشته شوند ممکن باشد در بعضی شرایط ایجاد شود)، سپس مقدار پیام Response فیلد وضعیت error-state به "noCreation" تنظیم شده، و مقدار فیلد error-index آن به index مربوط به variable-binding شکست خورده تنظیم می‌شود.

۸) در غیر این صورت، اگر نام binding-variable متغیری را تعیین کند که وجود ندارد، اما نمی تواند تحت شرایط فعلی ایجاد شود (حتی اگر در شرایط دیگر ایجاد شود)، مقدار فیلد error-status در پیام Response بر روی " inconsistentName"و مقدار فیلد error-index آن به index مربوط به variable-binding شکست خورده تنظیم می‌شود.

۹) در غیر این صورت، اگر نام variable-binding متغیری را تعیین کند که وجود دارد، اما قابل تغییر نیست فارغ از اینکه چه مقدار جدیدی تعیین شده ، سپس مقدار وضعیت پیام error-state Response بر روی "notWritable" تنظیم شده و مقدار فیلد error-index آن به index مربوط به variable-binding شکست خورده تنظیم می‌شود.

۱۰)در غیر این صورت، اگر نام variable-binding متغیری را تعیین کند که می تواند تحت شرایط دیگر توسط متغیر نگهداری شود، اما در حال حاضر ناسازگار است یا در غیر اینصورت قادر به اختصاص به متغیر نیست، سپس مقدار error-state مربوط به پیام Response برابر "inconsistentValue" تنظیم می شود و مقدار فیلد error-index آن به شاخص مربوط به variable-binding variable شکست خورده تنظیم می‌شود.

۱۱)هنگامی که در طی مراحل فوق، اختصاص مقدار توسط فیلد variable-binding به متغیر مشخص شده، نیاز به تخصیص یک منبع دارد که در حال حاضر در دسترس نیست، سپس مقدار error-status در پیام Response برابر " resourceUnavailable "و مقدار فیلد error-index آن به شاخص مربوط به variable-binding شکست خورده تنظیم می‌شود.

۱۲)اگر پردازش variable-binding به دلیل دیگری غیر از موارد فوق شکست بخورد، سپس مقدار error-status در پیام Response بر روی genErr تنظیم می شود و مقدار فیلد error-index آن به شاخص مربوط به پvariable-binding شکست خورده تنظیم می‌شود.

۱۳)در غیر این صورت، اعتبارسنجی variable-binding موفق می شود.

در پایان فاز اول، اگر اعتبار سنجی تمام variable-binding ها موفق شود، مقدار فیلد error-status در پیام Response به "noError" تنظیم می شود و مقدار فیلد index-index آن صفر است و پردازش بشرح زیر ادامه دارد.

برای هر variable-binding در درخواست، متغیر نامی در صورت لزوم ایجاد می شود و مقدار مشخص شده به آن اختصاص داده می شود. هر یک از این تخصیص متغیرها به طور همزمان با توجه به تمام تخصیص های دیگر مشخص شده در همان درخواست رخ می دهد. با این حال، اگر یک متغیر همزمان یک درخواست را بیش از یک بار ، با مقادیر مختلف نام‌گذاری کند ، سپس تخصیص واقعی به آن متغیر، implementation-

specific است.

اگر هر یک از این تخصیص ها شکست خورده باشد (حتی پس از تمام اعتبارسنجی های قبلی)، تمام انتصاب های دیگر لغو می شود، و پاسخ پیام اصلاح می شود تا مقدار فیلد error-status در آن را به "commitFailed" تنظیم کند و مقدار فیلد error-index آن به شاخص مربوط به variable-binding شکست خورده تنظیم می‌شود.

اگر و فقط اگر تمام تخصیص ها را لغو نکنیم، سپس پیام Response اصلاح می شود تا مقدار فیلد -error status در آن به «undoFailed» تنظیم شود و مقدار فیلد index-index آن به صفر برسد . توجه داشته باشید که پیاده سازی ها به شدت تشویق می شوند تا تمام اقدامات احتمالی را برای جلوگیری از استفاده از "commitFailed" یا "undoFailed" انجام دهند.

در نهایت، پیام Response تولید شده به یک پیام کپسوله شده تبدیل و به فرستنده پیام SetRequest ارسال می شود.

۷-٤-۱- پیام SNMPv2-Trap

پیام SNMPv2-Trap توسط یک موجودیت SNMP تولید شده و توسط یک برنامه اعلان ارسال می شود. پیام SNMPv2-Trap اغلب برای اعلان یک برنامه گیرنده اعلانات در یک موجودیت کنترل از راه دور منطقی SNMP است که یک رویداد اتفاق افتاده یا اینکه در شرایط موجود است. هیچ تأییدی در ارتباط با این مکانیزم تحویل اعلان وجود ندارد.

مقصدی (هایی) که یک پیام SNMPv2-Trap برای آن ها ارسال می شود، توسط یک موجودیت SNMP تعیین می شود. دو دسته انقیاد متغیر در لیست variable-binding پیام SNMPv2 Trap به ترتیب sysUpTime.0 [RFC3418] و snmpTrapOID.0 [RFC3418] هستند. اگر OBJECTS clause در فراخوانی مربوط به TYPE NOTIFICATION وجود داشته باشد، هر یک از متغیرهای مربوطه به عنوان نمونه ای از این notification، به ترتیب، به فیلد variable binding کپی می شود. اگر هر متغیر اضافی شامل می شود (در گزینه ای از ساختار SNMP)، سپس هر کدام به فیلد variable bindings کپی می شود.

۸-٤-۱- پیام InformRequest

یک پیام InformRequest بو سیله ی یک موجودیت SNMP از طرف برنامه سازنده اعلانتولید و ارسال می شود. پیام InformRequest اغلب برای اعلان یک برنامه دریافت کننده اعلانکه یک رویداد که اتفاق افتاده است یا وضعیت موجود مورد استفاده قرار می گیرد. این یک مکانیسم تحویل اعلانتأیید شده است، ، البته، تضمین تحویل وجود ندارد.

مقصدی (هایی) که یک پیام InformRequest برای آن ها ارسال می شود توسط برنامه ایجاد کننده

اعلانمشخص می‌شوند. دو دسته variable binding اول در لیست variable binding پیام InformRequest به ترتیب sysUpTime.0 [RFC3418] و snmpTrapOID.0 [RFC3418] هستند. اگر OBJECTS clause در فراخوانی مربوطه NOTIFICATION-TYPE macro وجود داشته باشد، هر یک از متغیرهای مربوطه به عنوان نمونه ای از این notification، به ترتیب، به فیلد variable-bindings کپی می شـــود. اگر هر متغیر اضافی را شامل شود (در گزینه ای از ساختار SNMP)، سپس هر کدام به فیلد variable-bindings کپی می شود.

پس از دریافت پیام InformRequest ، موجودیت SNMP دریافتی اندازه پیام کپسوله شدهResponse را با مقادیر مشـــابه در request-id ، error-status، error-index و variable-bindings به عنوان پیام InformRequest دریافت شـده تعیین می کند. اگر اندازه پیام مشـخص شـده بیشـتر از محدودیت محلی یا حداکثر اندازه پیام اصـــلی اســـت، ســـپس یک واکنش پیام متناوب تولید می شـــود، به فرســـتنده پیام InformRequest ارسال می شود و پردازش پیام InformRequest بلافاصله پس از آن متوقف می شود . این پاسـخ جایگزین پیام با مقادیر مشـابه در request-id field خود به عنوان پیام InformRequest دریافت می شود، با مقدار فیلد error-index خود را صفر تنظیم می کند و یک فیلد variable-bindings خالی جایگزین می شود. اگر اندازه پیغام نتیجه کمتر یا برابر محدودیت محلی و حداکثر اندازه پیام اصلی باشد، به فرستنده پیام InformRequest ارسال می شود. در غیر اینصورت، شمارنده snmpSilentDrops [RFC3418] افزایش می یابد و پیام حاصل از آن حذف می شود. صرف نظر از این، پردازش پیام InfoRequest متوقف می شود.

در غیر این صورت، یک موجودیت SNMP دریافتی:

۱) مطالب خود را به نرم افزار مناسب ارائه می کند.
۲) یک پیام Response با همان مقادیر در فیلدهای request-id و variable-bindings به عنوان پیام Receive InformRequest تولید می کند، که مقدار فیلد error-status آن که به "noError" و مقدار فیلد error-index آن را صفر تعیین می کند. و
۳) پیام Response تولید شده را به فرستنده پیام InformRequest ارسال می کند.

۵-۱- اطلاع از مالکیت معنوی

IETF در مورد اعتبار یا محدوده هر مالکیت معنوی یا حقوق دیگر مربوط به پیاده سازی که ممکن است ادعا شوند یا استفاده از فن آوری توصیف شده در این سند یا میزان مجاز بودن هر مجوز که ممکن است یا ممکن نیست در دسترس باشند هیچ جایگاهی ندارد. نشان نمی دهد تلاش برای شناسایی صحت چنین حقوقی را انجام شده باشد. اطلاعات در مورد روش های IETF در رابطه با حقوق در مستندات مربوط به استانداردهای پیگیری و استانداردهای ارتباطات در BCP-11 یافت می شود. کپی های ادعای حقوقی که برای انتشار و هر گونه تضمین مجوزها در اختیار قرار گرفته است و یا نتیجه تلاش برای کسب مجوز عمومی یا اجازه استفاده

از چنین حقوق مالکیت توسط اجراکنندگان یا کاربران این مشخصات می تواند از دبیرخانه IETF به دست آید.

IETF هر یک از طرفین ذینفع را دعوت می کند تا به هر گونه حق نسخه برداری، اختراع ثبت شده یا پرونده های ثبت اختراع و یا سایر حقوق مالکیت که ممکن است شامل تکنولوژی هایی باشد که ممکن است برای انجام این استاندارد مورد نیاز باشد، توجه کنند.