چکیده:

مستند پیش‌رو معرف پروتکل SNMP نسخه2 است. این مستند قواعد نحوی (syntax) و اجزای روندهای ارسال، دریافت و پردازش بسته های SNMP PDU را معرفی کرده است. این نسخه از مستند، مستند RFC 1905 را منسوخ می کند.

1. مقدمه:

چارچوب مدیریت SNMP در زمان نگارش این متن مشمول 5 جزء اصلی است:

- معماری کلی، توصیف شده در STD 62, RFC 3411
- مکانیزم هایی برای توصیف و نام‌گذاری اشیاء و رخدادها برای اهداف مدیریتی. اولین نسخه‌ی این ساختار اطلاعات مدیریتی که SMI نامیده می شود در STD 16, [RFC 1155] [RFC1155], STD 16, RFC 1212] [RFC1212] توصیف شده است. دومین نسخه که SMIv2 نامیده می شود در STD 58, RFC 2578] [RFC2578], STD 58, RFC 2579] [RFC2579] توصیف شده است.
- پروتکل های پیغام برای انتقال اطلاعات مدیریتی. اولین نسخه‌ی پروتکل پیغام SNMP که SNMPv1 نامیده می شود در STD 15, RFC 1157] [RFC1157] توصیف شده است. دومین نسخه پروتکل SNMP، که در پروتکل استاندار اینترنت نیست، SNMPv2c نامیده می شود در STD 62 and [RFC1901] RFC 1901, RFC 3417] [RFC3417]. توصیف شده است. سومین نسخه پروتکل پیغام SNMP که SNMPv3 نامیده می شود در RFC 3417] [RFC3417], RFC 3414] [RFC3414] and RFC 3412] [RFC3412]. توصیف شده است.
- عملیات‌های پروتکل برای دستیابی به اطلاعات مدیریتی. اولین مجموعه از عملیات‌های پروتکل وقالب‌های PDU وابسته به آن در STD 15, RFC 1157] [RFC1157]. توصیف شده است. دومین مجموعه از عملیات‌های پروتکل و قالب‌های PDU وابسته به آن در این مستند توصیف شده است.
- مجموعه ای از کاربردهای اساسی در STD 62, RFC 3413 و مکانیزم ها کنترل دسترسی بر پایه مشاهده در STD 62, RFC 3415] [RFC3415] توصیف شده است.

جزئیات بیشتر درباره مقدمه چارچوب مدیریتی SNMP در زمان نگارش این متن در RFC 3410] [RFC3410] قابل دسترسی و مطالعه است.

اشیاء مدیریت شده توسط منابع اطلاعات مجازی که پایگاه اطلاعات مدیریتی یا MIB نامیده می شوند، قابل دسترسی هستند. اشیاء در MIB توسط مکانیزم های تعریف شده در SMI معرفی می شوند.

این مستند که نسخه دوم عملیات پروتکل SNMP است، به معرفی عملیات این پروتکل با توجه به PDU های ارسال شده و دریافت شده توسط پروتکل پیغام، می پردازد.

2. دید کلی:

موجودیت های SNMP که پشتیبانی تولید کننده‌ی دستورات یا برنامه های دریافت کننده اعلان ها را دارا هستند (مدیرها) با موجودیت های SNMP که قابلیت پاسخگویی به دستورات یا برنامه های تولیدکننده اعلان‌ها(همان عامل ها) در ارتباط هستند. هدف اصلی این پروتکل انتقال اطلاعات مدیریتی و عملیات‌ها است.

اطلاعات مدیریتی:

واژه variable (متغیر) نشان دهنده که یک نمونه از انواع داده ای غیرترکیبی تعریف شده در چهارمین مجموعه از SMI یا قراردادهای مبتنی برپایه SMI است. واژه variable binding (زوج متغیر) به معنای یک جفت از

متغیر و ارزش آن متغیر است. اما اگر یکی از شرایط استثناء خاص حین پردازش بازیابی درخواست رخ دهد، زوج متغیر به معنای یک جفت از اسم و علامت آن خطا است.

لیست variable binding یک لیست ساده از زوج متغیرها است.

نام متغیر OBJECT INDENTIFIER است که اجتماعی از OBJECT INDENTIFIERهای انواع داده ای مرتبط با آن در درخت و OBJECT INDENTIFIER خود نمونه است. به OBJECT INDENTIFIER انواع داده ای مرتبط که در درخت مشخص می شود، پیشوند OBJECT INDENTIFIER گفته می شود.

بازارسالی درخواست ها:

در تمام درخواستهای این پروتکل، گیرنده در شرایط نرمال نیاز به تولید و ارسال پاسخ به تولیدکننده درخواست دارد. در صورت دریافت نشدن پاسخ به یک درخواست در بازه زمانی مشخص، درخواست باید دوباره ارسال شود. البته این موضوع در صلاحدید تولیدکننده درخواست است که مورد به صورت نرمال بستگی به اضطراری بودن درخواست دارد. اما به هرحال برنامه نیاز دارد تا رفتاری مسئولانه در برابر فرکانس و مدت زمان بازارسالی داشته باشد. [BCP 41 [RFC2914 بحثهای اساسی پیرامون موضوع کنترل ازدحام را مطرح میکند.

اندازه پیام:

حداکثر اندازه پیام SNMP برابراست با حداقل:

1. حداکثر اندازه پیام که موجودیت مقصد SNMP میتواند دریافت کند.
2. حداکثر اندازه پیام که موجودیت مبدا SNMP میتواند تولید کند.

حالت اول این است که این مقدار از قبل برای هر گیرنده ای مشخص است، و درصورت نداشتن این دانش وظیفه دامنه انتقال است که حین ارسال پیام آن را مشخص کند. مورد دوم این است که این مقدار توسط افرادی که آن را پیاده سازی کرده اند، به صورت مطلی مشخص و محدود شده است.

هر نگاشت انتقال در SNMP مشخص کننده حداقل اندازه پیامی است که طراحی SNMP میتواند آن را تولید یا مصرف کند. به رغم این که پیاده سازی ها می توانند در صورت امکان اندازه های بزرگتر را نیز پشتیبانی کنند اما این مقدار نباید بزرگتر از مقدار قابل پذیرش برای موجودیت گیرنده پیام SNMP باشد.

یکی از اهداف GetBulkRequest-PDU که در این پروتکل تعریف شده است، کمینه کردن تعداد تعویض های مورد نیاز برای بازیابی حداکثر اندازه اطلاعات مدیریتی است. بنابراین این نوع PDU اجازه می دهد تا موجودیت مدیر درخواست کند تا پاسخ ها به بزرگی مشخص شده در محدودیت اندازه پیام باشد. این محدودیتها شامل حد اندازه پیام در مدیر و عامل است.

اما ممکن است در مواردی اندازه حداکثر پیام بزرگتر از MTU شبکه ای باشد که این پیام می خواهد از آن عبور کند. در چنین شرایطی پیام باید تکه تکه شود. تکه تکه کردن پیام به طور کلی به عنوان یک عمل مضر در کنار دیگر مشکلات شناخته می شود زیرا کاهش اطمینان در انتقال پیام می شود. بنابراین موجودیت SNMP که GetBulkRequest-PDU را ارسال می کند، باید پارامترهای آن را بگونه ای تنظیم کند تا ریسک تکه تکه شدن پیام کاهش یابد. به طور خاص در شرایطی که شبکه شلوغ است باید مقدار کوچک برای max-repetitions استفاده شود.

نگاشت انتقال:

باید به این نکته توجه داشت که پیام های مبادله شده SNMP نیاز دارند تا از UDP استفاده کنند، زیرا که هر پیام به صورت کامل و مستقل شامل یک دیتاگرام انتقال است. نگاشتهای انتقال خاص و کد کردن قوانین در جای دیگری آورده شده است [RFC3417] اما نگاشتی که ترجیح داده شده است UDP است.

نگاشت انواع داده :SMIv2

SMIv2، 11 نوع داده ای پایه را که شامل (INTEGER ,OBJECT IDENTIFIER ,OCTET STRING, Opaque ,TimeTicks ,Gauge32,Unsigned32 ,Counter32 ,Integer32,IpAddress Counter64) و ساختار BIT می شود، تعریف کرده است. انواع پایه ای SMIv2 به نوع مرتبط در SimpleSyntax و ApplicationSyntax در ASN.1 در تعریف پروتکل SNMP نگاشت می شوند. باید توجه شود که INTEGER و Integer32 در انواع پایه ای SIMv2 به نوع integer-value در SimpleSyntax نگاشت می شود. به طور مشابه Gauge32 و Unsigned32 در انواع پایه ای SMIv2 به unsigned-integer-value در ApplicationSyntax نگاشت می شود.

ساختار BITS در SIMv2 به string-value در SimpleSyntax نگاشت می شود. مقدار BITS به صورت OCTET STRING کد می شود که در آن تمام بیت‌های نامیده شده که در تعریف bitstring آمده را می توان در اکتت های 8 بیتی قرار داد و آن را از بیت هشتم تا بیت اول پر کرد و این کار را به تعداد مورد نیاز بیتها ادامه داد. در آخر اگر اکتتی بود که بیتهایی از آن باقی مانده است، در هنگام تولید آن را برابر صفر و در هنگام استفاده آن ها را نادیده می گیریم.

3. تعاریف:

نحو PDU با استفاده از نشان گذاری ASN.1 تعریف شده است.

```
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 {

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
```

- مورد استفاده و صرف (Report-PDU semantic) در این مستند تعریف نشده است. هر چارچوب مدیریتی SNMP که از این PDU استفاده می کند باید مورد استفاده و صرف آن را مشخص کند.

```
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

4.ویژگی های پروتکل

4.1 ساختار های مشترک

قدار فیلد request-id در Response-PDU ، همان مقدار request-id در request-PDU ای است که این پاسخ مربوط به آن است. با استفاده از مقدار request-id یک کاربرد میتواند بین (احتمالا چند) درخواست های پاسخ داده نشده و در نتیجه پاسخ دریافتی مرتبط با درخواست پاسخ داده نشده تمایز قائل شود .در مواردی که UDP مورد استفاده قرار گرفته است ، request-id راه های ساده ای برای مشخص کردن پیغام های تکراری توسط شبکه تامین می کند . استفاده از request-id مشابه در یک باز انتشار درخواست این اجازه را فراهم می کند که پاسخ برای خود پیغام تعلق گیرد یا برای باز انتشار شده ی آن. اما به جهت محاسبه ی RTT برای ارسال و پردازش یک تراکنش شامل درخواست و پاسخ ،کاربرد نیازمند این است که از request-id متفاوت برای ریکوست باز نشر شده استفاده کند. در اکثر مواقع استفاده از روش دوم پیشنهاد می شود.

یک مقدار غیر صفر در فیلد ی error-status در Response-PDU ، برای نشان دادن وقوع یک خطا و جلوگیری از پردازش آن دستور به کار می رود .در این موارد مقدار غیر صفر در فیلد error-index در Response-PDU اطلاعات اضافه ای به وسیله‌ی شناساندن این که کدام زوج متغیر خطا را ایجاد کرده فراهم می‌کند. اولین زوج متغیر در لیست زوج متغیر ها جایگاه اول را دارد. دومین زوج جایگاه دوم و الی اخر.

SNMP مقدار OBJECT IDENTIFIER ها را به 128 sub-identifier محدود می‌کند به طوری که هر sub-identifier حداکثر محدودیت 2**31 -1 را دارد.

4.2 پردازش PDU

در عناصر رویه های زیر هر فیلد PDU که توسط رویه مرتبط به آن اشاره نشده است توسط موجودیت دریافت کننده‌ی SNMP نادیده گرفته می‌شود. اما همه ی اجزای PDU شامل فیلد هایی که نادیده گرفته میشوند نیز باید از قواعد نحوی ASN 1 و encoding معتبر استفاده کنند. برای مثال ، بعضی PDUها (GetRequest) فقط با اسم متغیر در نظر گرفته میشوند نه با مقدار آن. در این شرایط بخش مقداری متغیر توسط موجودیت دریافت کننده ی SNMP نادیده گرفته میشود. مقدار unspecified برای استفاده در چنین زوج هایی تعریف شده است.

در هنگام تولید ارتباط مدیریتی، "wrapper" جهت کپسول کردن PDU با توجه به عناصر رویه ی چارچوب اجرایی در دست تولید میشود. تعریف max-bindings نشان دهنده ی حد بالای تعداد زوج متغیر ها است. در عمل، سایز پیغام توسط الزام حداکثر سایز پیام نیز محدود میشود. یک پیاده سازی منطبق باید از بیشترین متغیر های ممکنه در PDUو BulkPDU تا جایی که از حد بالای اندازه بسته ی SNMP نمیگذرد پشتیبانی کند. اما نباید بیشتر از 2147483647 زوج متغیر شود.

به هنگام دریافت ارتباط مدیریتی، عناصر رویه چارچوب اجرایی در دست استفاده به کار گرفته میشوند و اگر رویه ها نشان دهند که عملیاتی که در پیام است باید به صورت محلی اجرا شود، این رویه ها باید نمایه ی MIB را نیز که برای عملیات آشکار است را نشان دهند.

4.2.1 GetRequest-PDU

یک GetRequest-PDU پیرو درخواست یک کاربرد تولید و منتشر می‌شود. به محض دریافت GetRequest-PDU موجودیت دریافت کننده ی SNMP اقدام به پردازش زوج متغیر ها میکند تا پیغام پاسخ را تولید کند. همه ی فیلد های

Response-PDU باید مقدار مشابهی متناظر با مقادیر دریافتی از پیغام درخواست داشته باشند به جز در موارد زیر. موجودیت پروتکل دریافت کننده مطابق با هر قانون قابل اجرا در لیست زیر پاسخ می دهد:

1. اگر نام زوج متغیر دقیقا مشابه نام متغیر در دسترس توسط درخواست باشد ،پس مقدار زوج متغیر باید با مقدار متغیر نامیده شده مقدار دهی شود.
2. در غیر اینصورت اگر نام زوج متغیر پیشوند OBJECT IDENTIFIER دقیقا مشابه با پیشوند OBJECT IDENTIFIER هیچ متغیر در دسترسی در درخواست یکسان نبود، پس مقدار فیلدش به noSuchObject مقدار دهی میشود.
3. در غیر این صورت مقدار زوج متغیر با noSuchInstnace مقدار دهی میشود.

اگر پردازش هر زوج متغیری به دلیلی غیر از دلایل بالا شکست بخورد، Response-PDU با مقادیر مشابه در request-id و فیلد های زوج متغیر های دریافتی از GetRequest-PDU به همراه مقدار genErr فیلد error-status مرتب میشود و مقدار error-index با جایگاه زوج متغیر شکست خورده مقدار دهی میشود.

در غیر این صورت،مقدار فیلد error-status در پیغام پاسخ با noError مقدار دهی میشود و مقدار error-index متناظرش صفر میشود.

پاسخ تولید شده سپس در یک پیام کپسول میشود . اگر سایز پیام در بازه ی مجاز اندازه سازنده و پاسخ دهنده بود ، پیام به سمت درخواست کننده ارسال می‌شود.

در غیر این صورت یک پیغام پاسخ جایگزین با همان request-id ساخته میشود که مقدار error-status ان با tooBig مقدار دهی شده و مقدار error-index آن صفر خواهد بود. و فیلد های زوج متغیر آن خالی خواهد بود. اگر در نهایت سایز این پیغام در حدود اندازه مجاز بود ارسال میشود و گرنه شمارنده ی snmpSilentDrops شروع به زیاد شدن میکند و پیام drop میشود.

GetNextRequest-PDU 4.2.2

یک پیغام GetNextRequest-PDU به درخواست کاربرد ساخته و فرستاده می‌شود. به محض دریافت یک GetNextRequest-PDU موجودیت دریافت کننده ی SNMP تک تک زوج متغیر های لیست زوج متغیر ها را به جهت تولید یک پاسخ پردازش میکند. همه ی فیلد های Response-PDU مقادیر مشابه متناظر با فیلد های درخواست دریافتی دارند به جز موارد زیر. هر الزام متغیر به روش زیر پردازش میشود.

1. متغیری که در لیست lexicographically مرتب شده از نام متغیر هایی که قابل دسترس توسط درخواست هستند و نام آنها اولین جانشین lexicographic نام زوج متغیر در GetNextRequest-PDU است. فیلد نام و مقدار زوج متغیر متناظر در Response-PDU به عنوان نام و مقدار متغیر پیدا شده قرار میگیرند.
2. اگر نام زوج متغیر درخواستی به صورت `lexicographically` جلوتر از نام متغیری که قابل دسترس توسط درخواست است نبود ، به طور مثال هیچ جانشین `lexicographic` نبود، پس زوج متغیر متناظر تولید شده در `Response-PDU` دارای فیلد با مقدار `endOfMibView` است. و فیلد نام آن با نام زوج متغیر پیغام درخواست پر میشود.

اگر پردازش هر کدام از زوج متغیر به هردلیلی به جز دلایل بالا شکست بخورد سپس پیغام پاسخ با همان مقادیر در زوج متغیر ها و شناسه ی پیغام `getNextRequset-PDU` بازسازی میشود.مقدار `error-status` آن نیز با `genErr` پرشده و مقدار `error-index` با جایگاه زوج متغیر شکست خورده پر میشود.

در غیر این صورت مقدار فیلد خطا در `Response-PDU` با `noError` و `error-index` با صفر پر میشود.

پیغام پاسخ ساخته شده در پیام نهایی کپسوله میشود. اگر سایز پیام نهایی کوچکتر مساوی محدودیت های فرستنده و گیرنده بود، به سمت ارسال کننده ی درخواست گسیل میشود.

در غیر این صورت یک پیغام پاسخ جایگزین با همان request-id ساخته میشود که مقدار error-status ان با tooBig مقدار دهی شده و مقدار و مقدار error-index آن صفر خواهد بود. و فیلد های زوج متغیر آن خالی خواهد بود. اگر در نهایت سایز این پیغام در حدود اندازه مجاز بود ارسال میشود و گرنه شمارنده ی snmpSilentDrops شروع به زیاد شدن میکند و پیام drop میشود.

4.2.2.1 مثالی از پیمایش جدول

یکی از کاربرد های استفاده از getNextRequest-PDU پیمایش جداول انتزاعی اطلاعات موجود در MIB است. معانی این درخواست به همراه روش شناسایی نمونه های اشیا در MIB ، دسترسی به اشیا مرتبطه در MIB را فراهم میکنند.

در تبادل پیام پروتکلی ترسیم شده در زیر، یک کاربرد ، ادرس فیزیکی وابسته به رسانه و نوع نگاشت ادرس برای یک موجودیت در جدول ترجمه ی ادرس عناصر مشخص IP net-to-media را دریافت میکند. همچنین مقدار sysUpTime را نیز دریافت میکند. فرض کنید جدول پاسخ دهنده سه موجود ثبت شده داشته باشد.

Interface-Number	Network-Address	Physical-Address	Type
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

SNMP entity که از برنامه تولید کننده دستورات پشتیبانی میکند با فرستادن یک GetNextRequest-PDU ای که مقدار OBJECT IDENTIFIER مشخص شده را به عنوان متغیر درخواست شده شامل میشود , شروع میکند:

GetNextRequest (sysUpTime,

ipNetToMediaPhysAddress,

ipNetToMediaType)

SNMP entity که از برنامه پاسخ دهنده دستورات پشتیبانی میکند با یک Response-PDU پاسخ میدهد:

Response ((sysUpTime.0 = "123456"),

(ipNetToMediaPhysAddress.1.9.2.3.4 = "000010543210"),

(ipNetToMediaType.1.9.2.3.4 = "dynamic"))

SNMP entity که از برنامه تولید کننده دستورات پشتیبانی میکند ادامه میدهد:

GetNextRequest (sysUpTime,

ipNetToMediaPhysAddress.1.9.2.3.4,

ipNetToMediaType.1.9.2.3.4)

SNMP entity که از برنامه پاسخ دهنده دستورات پشتیبانی میکند پاسخ میدهد:

Response ((sysUpTime.0 = "123461"),

(ipNetToMediaPhysAddress.1.10.0.0.51 = "000010012345"),

(ipNetToMediaType.1.10.0.0.51 = "static"))

SNMP entity که از برنامه تولید کننده دستورات پشتیبانی میکند ادامه میدهد:

GetNextRequest (sysUpTime,

ipNetToMediaPhysAddress.1.10.0.0.51,

ipNetToMediaType.1.10.0.0.51)

SNMP entity که از برنامه پاسخ دهنده دستورات پشتیبانی میکند پاسخ میدهد:

Response ((sysUpTime.0 = "123466"),

(ipNetToMediaPhysAddress.2.10.0.0.15 = "000010987654"),

(ipNetToMediaType.2.10.0.0.15 = "dynamic"))

SNMP entity که از برنامه تولید کننده دستورات پشتیبانی میکند ادامه میدهد:

GetNextRequest (sysUpTime,

ipNetToMediaPhysAddress.2.10.0.0.15,

ipNetToMediaType.2.10.0.0.15)

از آنجایی که هیچ entry دیگری در جدول نیست SNMP entity که از برنامه پاسخ دهنده به دستورات پشتیبانی میکند با متغیری که مقدار بعدی در ترتیب قاموسی نام اشیا قابل دسترس است پاسخ میدهد. مثلا:

Response ((sysUpTime.0 = "123471"),

(ipNetToMediaNetAddress.1.9.2.3.4 = "9.2.3.4"),

(ipRoutingDiscards.0 = "2"))

در نظر داشته باشید چگونه با رسیدن به انتهای ستون ipNetToMediaPhysAddress دومین variable binding از برنامه پاسخ دهنده به دستورات هم اکنون به اولین در ستون بعدی "wrapped" شده است. در نظر داشته باشید چگونه با رسیدن به انتهای ستون ipNetToMediaPhysAddress برای سومین variable binding برنامه پاسخ دهنده به دستورات با شئ در دسترس بعدی که بیرون از جدول است جواب داده است. این جواب نشان دهنده انتهای جدول برای برنامه تولید کننده دستورات است.

4.2.3 GetBulkRequest

یک GetBulkRequest-PDU به خاطر درخواست یک برنامه تولید و فرستاده میشود. هدف GetBulkRequest-PDU برای درخواست انتقال مقدار بالنسبه زیادی از داده شامل ولی نه محدود به کارایی و سرعت در بازیابی جداول بزرگ میشود.

در دریافت یک GetBulkRequest-PDU , SNMP entity گی رنده برای تولید یک Response-PDU هر variable binding در لیست variable binding را پردازش میکند و فیلد request-id را همان مقدار در دستور request میگذارد.

برای نوع GetBulkRequest-PDU پردازش موفق هر variable binding در request باعث تولید صفر یا بیشتر variable binding در Response-PDU میشود. بطور دقیق تر , نگاشت یک به یک بین variable binding های GetRequest-PDU, GetNextRequest-PDU, SetRequest-PDU و Response-PDU حاصل نگاشت بین variable binding های یک GetBulkRequest-PDU و Response-PDU ی حاصل را اجرا نمیکند.

مقدار فیلدهای "non-repeaters" و "max-repetitions" در request پردازش درخواست شده را مشخص میکند.

یک variable binding در Response-PDU برای اولین N variable binding در request درخواست شده و variable binding M برای هر variable binding R باقی مانده در request . در نتیجه تعداد variable binding هایی که توسط request فرستاده شده برابر با N + (M*R) خواهد بود که N مینیمم :

- مقدار فیلد "non-repeaters" در request
- تعداد variable binding ها در request

و M مقدار فیلد "max-repetitions" در request و R ماکزیمم:

- تعداد زوج متغیر ها در درخواست – N
- 0

خواهند بود.

SNMP entity گیرنده یک Response-PDU با حداکثر به تعداد variable binding های درخواست شده که با request فرستاده شده اند را تولید میکند. Request-id باید همان مقدار GetBulkRequest-PDU دریافت شده را بگیرد.

اگر N از 0 بیشتر بود اولین تا Nامین variable binding در Response-PDU به طریق زیر تولید میشوند:

1. متغیری که به ترتیب قاموسی نام همه متغیرها که با این درخواست در دسترس است و نامش اولین جانشین قاموسی در نام variable binding در GetBulkRequest-PDU , تعیین میشود. فیلد نام و مقدار variable binding متناظر در Response-PDU با نام و مقدار متغیر تعیین شده , قرار داده میشود.
2. اگر نام variable binding درخواست شده به لحاظ قاموسی قبل از نام هیچ متغیر در دسترس با این درخواست نبود , یعنی هیچ جانشین قاموسی نبود آنگاه variable binding متناظر تولید شده در Response-PDU مقدار فیلد value را مساوی "endOfMibView" قرار میدهد و فیلد نام را با نام Variable binding در درخواست مساوی قرار میدهد.

اگر M و R غیر صفر باشند (N+1) امین variable binding و متغیرهای بعدی Response-PDU هرکدام به روش

مشابهی تولید میشوند. برای هر تکرار i (i بزرگتر از 0 و کوچکتر مساوی M)و برای هر متغیر تکرار شده r (r بزرگتر از 0 و کوچکتر مساوی N) (R + ((i-1) * R)) + r) امین variable binding در Response-PDU به ترتیب زیر تولید میشود:

1. متغیری که در لیست ترتیب قاموسی نام های همه متغیر های در دسترس با این درخواست که نامش i امین جانشین قاموس ی (N + r) امین نام variable binding در GetBulkRequest-PDU دریافتی تعیین میشود و فیلد نام و مقدار variable binding با نام و مقدار متغیر تعیین شده برابر میشود.
2. اگر هیچ i امین جانشینی نبود variable binding متناظر تولید شده در Response-PDU فیلد مقدارش با"endOfMibView" برابر و فیلد نامش یا با آخرین جانشین یا اگر هیچ جانشینی نباشد با نام (N+r) امینvariable binding در request برابر میشود.

با اینکه حداکثر تعداد variable binding ها در Response-PDU توسط N + (M*R) محدود میشود ممکن است پاسخ به یکی از سه دلیل زیر تعداد کمتری variable binding (حتی تا 0) تولید کند:

1. اگر سایز پیامی که قرار است در Response-PDU کپسول شود و حاوی تعداد درخواست شده variable binding است از یک محدودیت محلی یا "max message size" تولید کننده بیشتر باشد , در آن صورت پاسخ با تعداد کمتری از variable binding ها تولید میشود. این تعداد کمتر مجموعه مرتب شده variable binding ها با حذف برخی از آنها از انتهای مجموعه است به طوری که سایز پیام کپسول شده تقریبا برابر و نه بزرگتر از محدودیت محلی یا "max message size" تولید کننده باشد. توجه کنید که تعداد variable binding های حذف شده هیچ ربطی به مقادیر N,M,R ندارد.
2. پاسخ ممکن است با تعداد کمتری از variable binding ها تولید شود اگر به ازای برخی مقادیر تکرار i (i بز رگتر از 0 و کوچکتر از M)تمام variable binding های تولید شده مقدار "endOfMibView" را داشته باشند. در این مواقع ممکن است variable binding ها بعد از ((i*R)+N) امین variable binbing بریده شوند.
3. در صورتی که پردازش یک درخواست با تعداد زیادی تکرار مقدار قابل توجهی زمان برای پردازش نسبت به یک درخواست معمول بگیرد در آن صورت ممکن است برنامه پاسخ دهنده به دستورات ممکن است به آن درخواست با تعداد کمتری از تکرارها خاتمه بدهد(با اتمام حداقل 1 تکرار).

اگر پردازش یکی از variable binding ها به دلیل دیگری بجز موارد بالا با مشکل روبه رو شد , Response-PDU با مقدار مشابه request-id و فیلدهای variable binding که در GetBulkRequest-PDU , که مقدار error-status آن "genErr" و مقدار error-index آن اندیس متغیری که در پیام اصلی در variable binding دچار مشکل شد , دوباره قالب بندی میشود . در غیر این صورت مقدار فیلد error-status در "noError" Response-PDU و مقدار فیلد error-index 0 خواهد بود.

Response-PDU تولید شده (احتمالا با فیلد خالی variable binding) در یک پیام کپسول می شود. اگر سایز پیام حاصله کوچکتر یا مساوی هردو محدودیت های محلی و "max message size" تولید کننده بود به تولید کننده GetBulkRequest-PDU ارسال میشود در غیر این صورت شمارنده "snmpSilentDrops" یکی زیاد شده و پیام حاصله درو انداخته میشود.

۴.۲.۳.۱. مثالی دیگر از پیمایش جدول

این مثال نشان میدهد که چگونه می توان از GetBulkRequest-PDU به عنوان جایزینی برای GetNextRequest-PDU استفاده کرد. همان پیمایش جدول IP Net-to-media که در قسمت ۴.۲.۲.۱ نشان داده شده است با مبادلات کمتری حاصل می شود.

موجودیت SNMP که از برنامه تولیدکننده فرمان پشتیبانی می کند با ارسال یک GetBulkRequest-PDU با مقدار حداکثر تکرار 2 و شامل مقادیر OBJECT IDENTIFIER به عنوان نام متغیر های درخواست شده شروع می شود:

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

```
                    ipNetToMediaPhysAddress,
ipNetToMediaType )
```

موجودیت SNMP که از برنامه پاسخگویی فرمان پاسخگویی میکند با Response-PDU 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 تولید کننده فرمان پشتیبانی می کند ادامه پیدا میکند با :

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

موجودیت SNMP که از برنامه پاسخگویی فرمان پاسخگویی میکند پاسخ می دهد با:

```
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 داشته باشید که چگونه مانند مثال اول ، variable binding در پاسخ نشان می دهد که به انتهای جدول رسیده است. چهارمین variable binding با بازگشت اطلاعات از ستون موجود بعدی این کار را انجام می دهد. پنجمین variable binding با برگرداندن اطلاعات از اولین شی موجود در فهرست از نظر جدول ، این کار را انجام می دهد. این پاسخ انتهای جدول را برای برنامه تولید کننده دستورات نشان میدهد.

۴.۲.۴ Response-PDU

Response-PDU فقط با دریافت GetRequest-PDU ، GetNextRequest-PDU ، GetBulkRequest-PDU ، SetRequest-PDU یا InformRequest-PDU توسط یک موجودیت SNMP تولید می شود.

اگر فیلد error-status در Respons-PDU صفر نباشد، مقدار فیلد های variable bindings در لیست variable binding نادیده گرفته می شوند.

اگر هر دو فیلد error-status و error-index در Respons-PDU غیر صفر باشند، آنگاه مقدار فیلد error-index عبارت است از ایندکس variable binding (در لیست variable-binding درخواست مربوطه) که برای آن درخواست انجام نشد. اولین variable binding در لیست variable-binding درخواست، ایندکس ۱ ، دومین ایندکس ۲ و به همین ترتیب قابل دسترس اند.

یک موجودیت SNMP سازگار که از یک تولید کننده ی فرمان پشتیبانی می کند باید بتواند به طور صحیح یک Response-PDU را با یک فیلد error-status برابر با "noSuchValue" , "badVal" یا "readOnly" دریافت و مدیریت کند. (بخش های ۱.۳ و ۴.۳ از [RFC2576] ببینید)

پس از دریافت Respons-PDU، موجودیت دریافت کننده SNMP محتویات خود را به برنامه ای ارائه می دهد که درخواست را با همان مقدار request-id ارائه داده است. برای اطلاعات بیشتر [RFC3412] را ببینید.

۴.۲.۵. SetRequest-PDU

SetRequest-PDU بنا به درخواست یک برنامه تولید و انتقال می یابد.

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

1. اگر قسمت **نام** variable binding یک متغیر موجود یا ناموجود را مشخص کند ،اگر این درخواست سلب دسترسی شود (چون از نظر MIB مناسب نیست) در این صورت مقدار "**noAccess**" در فیلد error-status قرار می گیرد و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.
2. در غیر این صورت ،اگر **نام** variable binding وجود داشته باشد ، و بدون توجه به مقدار جدید مشخص شده قادر به ایجاد یا اصلاح آن نباشد ، مقدار فیلد error-status در Response-PDU برابر "**notWritable**" قرار داده می شود و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.
3. در غیر اینصورت ، اگر **مقدار** variable binding ، مطابق با زبان ASN.1، نوعی نا سازگار با تعریف آن داشته باشد ، مقدارerror-status در Response-PDU برابر "**wrongType**" قرار داده می شود و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود. این خطا موقعی روی می دهد که متغیری از نوع INTEGER را در string قرار دهیم .
4. در غیر اینصورت ، اگر **مقدار** variable binding مطابق با زبان ASN.1 ، طولی بیشتر از آن مقدار که تعیین شده است اختیار کند ، آنگاه مقدار error-status در Response-PDU برابر "**wrongLength**" قرار داده می شود و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.
5. در غیر این صورت ، اگر **مقدار** variable binding دارای یک کدگذاری ASN.1 باشد که با برچسب ASN.1 آن فیلد مغایر است ، مقدار error-status در Response-PDU برابر "**wrongEncoding**" قرار می گیرد و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود. (توجه داشته باشید که همه استراتژی های اجرا این خطا را ایجاد نمی کنند.)
6. در غیر این صورت ، اگر **مقدار** variable binding مقداری را مشخص کند که تحت هیچ شرایطی نمی تواند به متغیر اختصاص یابد برای مثال این می تواند زمانی اتفاق بیفتد که یک read-write به عنوان enumeration تعریف‌شده باشد ، و شما سعی می کنید آن را روی مقداری تنظیم کنید که جزء انواع ذکر شده نیست. ، مقدار error-status در Response-PDU برابر "**wrongValue**" قرار می گیرد و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.
7. در غیر این صورت ، اگر **نام** variable binding متغیری را مشخص کند که وجود ندارد و هرگز در MIB نمی تواند ایجاد شود (حتی اگر برخی از متغیرهای مشترک با همان پیشوند OBJECT IDENTIFIER ممکن است در برخی شرایط قادر به ایجاد باشند) ، مقدار error-status در Response-PDU برابر "**noCreation**" قرار می گیرد و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.
8. در غیر این صورت ، اگر **نام** variable binding ، متغیری را مشخص کند که وجود نداشته باشد ، اما تحت شرایط فعلی نمی تواند ایجاد شود (حتی اگر می تواند در شرایط دیگری ایجاد شود) برای مثال نام متغیر با نام متغیری دیگر یکسان باشد ، سپس مقدار فیلد error-status در Response-PDU برابر "**inconsistentName**" قرار می گیرد و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.
9. در غیر این صورت ، اگر **نام** variable binding متغیری را مشخص کند که وجود دارد اما مهم نیست که چه مقدار جدیدی مشخص شده باشد ، می توان مقدار error-status در Response-PDU را روی "**notWritable**" قرار داد ، مقدار-error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.
10. در غیر این صورت ، اگر **مقدار** variable binding مقداری را مشخص کند که می تواند تحت شرایط دیگر توسط متغیر نگه داشته شود ، اما در حال حاضر متناقض است (با خود MIB یا با مقدار سایر متغیر ها)و یا قادر به اختصاص به متغیر نیست

، مقدار error-status در Response-PDU برابر "**inconsistentValue**" قرار می گیرد ، ومقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.

11. هنگامی که در طی مراحل فوق ، اختصاص مقدار مشخص شده توسط فیلد مقدار variable binding به متغیر مشخص شده ، نیاز به تخصیص منبعی دارد که در حال حاضر در دسترس نباشد ، مقدار error-status در Response-PDU برابر "**sourceUnavailable**" قرار می گیرد ، و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.
12. اگر پردازش variable binding به دلایلی غیر از موارد ذکر شده در بالا انجام نشود ، مقدار error-status در Response-PDU برابر "**genErr**" قرار می گیرد ،و مقدار error-index ، ایندکس فیلدی که خطا داشته است مقدار دهی می شود.
13. در غیر این صورت ، اعتبارسنجی variable binding موفقیت آمیز است.

منبع مثال ها :

- https://community.cisco.com/t5/network-management/different-error-states-when-using-wrong-mib-in-snmp-v1-and-v2c/td-p/1277654
- https://docs.microsoft.com/en-us/windows/win32/snmp/snmp-error-codes

۴.۲.۶. SNMPv2-Trap-PDU

SNMPv2-Trap-PDU توسط یک موجودیت SNMP به نمایندگی از برنامه سازنده اعلان ، تولید و منتقل می شود. SNMPv2-Trap-PDU اغلب برای اطلاع رسانی به یک برنامه گیرنده اطلاع رسانی در یک موجودیت SNMP با منطق از راه دور استفاده می شود بدین منظور که یک واقعه رخ داده است یا شرایطی وجود دارد.در این مکانیسم هیچ تاییدی بر تحویل اعلان وجود ندارد.

مقصد (های) ارسال شده برای SNMPv2-Trap-PDU به روش وابسته به اجرا توسط موجودیت SNMP تعیین می شود. دو variable binding ابتدایی در لیست ، sysUpTime.0 و snmpTrapOID.0 هستند. اگر بند OBJECTS در فراخوانی ماکرو مربوط به NOTIFICATION-TYPE موجود باشد ، پس از آن هر یک از متغیر های نمونه ، به ترتیب در variable binding کپی میشوند. اگر اطلاعات اضافی ای درج شده باشند ، به ترتیب در فیلد variable-binding کپی می شوند.

۴.۲.۷. The InformRequest-PDU

InformRequest-PDU توسط یک موجودیت SNMP به نمایندگی از برنامه سازنده اعلان تولید و منتقل می شود.

InformRequest-PDU اغلب برای اطلاع رسانی به برنامه گیرنده اعلان مبنی بر وقوع یک رویداد یا وجود یک شرط استفاده می شود. این یک مکانیسم تأیید اعلان تأیید شده است ، اگرچه هیچ تضمینی برای تحویل وجود ندارد.

مقصد (های) ارسال شده برای `InformRequest-PDU` توسط برنامه مبانی اعلان مشخص شده است. دو variable binding ابتدایی در لیست ، sysUpTime.0 و snmpTrapOID.0 هستند. اگر بند OBJECTS در فراخوانی ماکرو مربوط به NOTIFICATION-TYPE موجود باشد ، پس از آن هر یک از متغیر های نمونه ، به ترتیب در variable binding کپی میشوند. اگر اطلاعات اضافی ای درج شده باشند ، به ترتیب در فیلد variable-binding کپی می شوند.

به محض دریافت یک InformRequest-PDU ، واحد SNMP دریافت کننده اندازه یک پیام را محصور می کند که یک Response-PDU را با همان request-id ، error-status ، error-index و variable-bindings به عنوان InformRequest-PDU تعیین می کند. اگر اندازه پیام تعیین شده از یک محدودیت محلی یا حداکثر اندازه پیام مبدأ بزرگتر باشد ، یک Response-PDU ایجاد می شود ، به مبدأ InformRequest-PDU منتقل می شود و پردازش InformRequest-PDU بلافاصله پس از آن پایان می یابد. این Response-PDU با همان مقادیر در قسمت request-id خود مطابق با InformRequest-PDU دریافت شده قالب بندی می شود ، مقدار فیلد

error-status برابر با "tooBig" قرار داده می شود ، مقدار فیلد error-index آن برابر با صفر و فیلد variable binding خالی می ماند.