# چکیده:

مستند پیشرو معرف پروتکل SNMP ورژن 2 می­باشد. این مستند صرف (syntax) و المان­های روند ارسال، دریافت و پردازش بسته­های SNMP PDU را معرفی کرده است. این مستند، مستند RFC 1905 را منسوخ می­کند.

# مقدمه:

چارچوب مدیریت SNMP در زمان نگارش این متن دارای 5 جزء اصلی می­باشد:

* معماری کلی، توصیف شده در STD 62, RFC 3411
* مکانیزم­هایی برای توصیف و نام گذاری اشیاء و اتفاقات برای اهداف مدیریتی. اولین ورژن این ساختار اطلاعات مدیریتی که SMI نامیده می­شود در STD 16, RFC 1155 [RFC1155], STD 16, RFC 1212 [RFC1212] توصیف شده است. دومین ورژن که نامیده می­شود در STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] توصیف شده است.
* پروتکل­های پیغام برای انتقال اطلاعات مدیریتی. اولین ورژن پروتکل پیام در SNMP که ... نامیده می­شود در STD 15, RFC 1157 [RFC1157] توصیف شده است. دومین ورژن پروتکل پیام در SNMP، که در پرتکل استاندار اینترنت نیست، ... نامیده می­شود در RFC 1901[RFC1901] and STD 62, RFC 3417 [RFC3417]. توصیف شده است. سومین ورژن پروتکل پیام در SNMP که ... نامیده می­شود در RFC 3417 [RFC3417], RFC 3412 [RFC3412] and RFC 3414 [RFC3414]. توصیف شده است.
* عملیات پروتکل برای دستیابی به اطلاعات مدیریتی. اولین مجموعه از عملیات پروتکل و فرمت­های PDU وابسته به آن در STD 15, RFC 1157 [RFC1157]. توصیف شده است. دومین مجموعه از عملیات پروتکل و فرمت­های PDU وابسته به آن در این مستند توصیف شده است.
* مجموعه­ای از برنامه­های اساسی توصیف شده در STD 62, RFC 3413 و مکانیزم­ها کنترل دسترسی بر پایه مشاهده توصیف شده در STD 62, RFC 3415 [RFC3415]

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

اشیاء مدیریت شده توسط منابع اطلاعات مجازی که پایگاه اطلاعات مدیریتی یا MIB نامیده می­شوند، قابل دسترسی هستند. اشیاء در MIB توسط مکانیزم­های تعریف شده در SMI معرفی می­شوند.

این مستند که ورژن دوم عملیات پروتکل SNMP می­باشد، به معرفی عملیات این پروتکل با توجه به PDU های ارسال شده و دریافت شده توسط پروتکل پیام، می­پردازد.

# دید کلی:

موجودیت­های SNMP که قابلیت تولید دستورات یا برنامه­های دریافت­کننده اطلاعات ( همان مدیرها) با موجودیت­های SNMP که قابلیت پاسخگویی به دستورات یا برنامه­های تولیدکننده اطلاعات( همان عامل­ها) در ارتباط هستند. هدف اصلی این پروتکل انتقال اطلاعات مدیریتی و عملیات می­باشد.

## اطلاعات مدیریتی:

واژه variable (متغیر) نشان دهنده که یک نمونه از انواع داده­ای غیرترکیبی تعریف شده در چهارمین مجموعه از SMI یا قراردادهای متنی برپایه SMI می­باشد. واژه variable binding ( زوج متغیر) به معنای یک جفت از متغیر و ارزش آن متغیر می­باشد. 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 می­باشد.

لیست variable binding یک لیست ساده از زوج متغیرها می­باشد.

نام متغیر OBJECT INDENTIFIER است که ترکیبی از OBJECT INDENTIFIERهای انواع داده­ای مرتبط با آن در درخت به علاوه OBJECT INDENTIFIER مشخص برای خود نمونه می­باشد. به OBJECT INDENTIFIER انواع داده­ای مرتبط که در درخت مشخص می­شود، OBJECT INDENTIFIER پیشوند گفته می­شود.

## بازارسالی درخواست­ها:

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 می­باشد 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 گیرنده­ای مشخص است، و درصورت نداشتن این مقدار وظیفه دامنه انتقال است که حین ارسال پیام آن را مشخص کند. حالت دوم این است که این مقدار توسط افرادی که آن را پیاده­سازی کرده­اند، به صورت محلی مشخص و محدود شده است.

هر نگاشت انتقال در SNMP مشخص کننده حداقل اندازه پیامی می­باشدکه طراحی SNMP می­تواند آن را تولید یا مصرف کند. به رغم این که پیاده­سازی­ها می­توانند در صورت امکان اندازه­های بزرگتر را نیز ساپورت 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 تعویض­های مورد نیاز برای بازیابی حداکثر اندازه اطلاعات مدیریتی می­باشد. بنابراین این نوع PDU اجازه می­دهد تا موجودیت مدیر درخواست کند تا پاسخ­ها به بزرگی مشخص شده در محدودیت اندازه پیام باشد. این محدودیتها شامل حد اندازه پیام در مدیر و عامل می­باشد.

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

## نگاشت انتقال:

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

## نگاشت انواع داده SMIv2:

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

ساختار BITS در SIMv2 به string-value در SimpleSyntax نگاشت می­شود. مقدار BITS به صورت OCTET

STRING کد می­شود که در آن تمام نام­ها بیت که در تعریف bitstring آمده را می­توان در اکتت­های 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 آن­ها را نادیده می­گیریم.

# تعاریف:

نحو PDU با استفاده از نشان­گذاری 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 (semantic) Report-PDU در این مستند تعریف نشده است. هر چارچوب مدیریتی SNMP که از این PDU استفاده می­کند 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