Санкт-Петербург, ул. Седова, 80, телефон (812) 449-0770, факс (812) 449-0771, E-mail: info@bilim.com

Network Working Group
Request for Comments: 1157
Obsoletes: RFC 1098

J. Case
SNMP Research
M. Fedor
Performance Systems International
M. Schoffstall
Performance Systems International
J. Davin
MIT Laboratory for Computer Science

May 1990

Простой протокол сетевого управления (SNMP)

A Simple Network Management Protocol (SNMP)

Оглавление

1. Статус документа....2
2. Введение....2
3. Архитектура SNMP....2
3.1. Задачи архитектуры....3
3.2. Элементы архитектуры....3
3.2.1. Область данных управления....3
3.2.2. Представление данных управления....3
3.2.3. Операции, поддерживаемые для данных управления....3
3.2.4. Форма и значение протокольной информации....4
3.2.5. Определение административных отношений....4
3.2.6. Форма и значение ссылок на управляемые объекты....5
3.2.6.1. Преобразование неоднозначных ссылок MIB....5
3.2.6.2. Преобразование при наличии множества версий MIB....5
3.2.6.3. Идентификация экземпляров объекта....5
3.2.6.3.1. Имена типа ifTable....6
3.2.6.3.2. Имена типа atTable....6
3.2.6.3.3. Имена типа ipAddrTable....6
3.2.6.3.4. Имена типа ipRoutingTable....6
3.2.6.3.5. Имена типа tcpConnTable....6
3.2.6.3.6. Имена типа egpNeighTable....6
4. Спецификация протокола....6
4.1. Элементы процедуры....7
4.1.1. Конструкции общего назначения....8
4.1.2. GetRequest-PDU....8
4.1.3. GetNextRequest-PDU....9
4.1.3.1. Пример обхода таблицы....9
4.1.4. GetResponse-PDU....10
4.1.5. SetRequest-PDU....10
4.1.6. Trap-PDU....11
4.1.6.1. Прерывание coldStart....11
4.1.6.2. Прерывание warmStart....11
4.1.6.3. Прерывание linkDown....11
4.1.6.4. Прерывание linkUp....12
4.1.6.5. Прерывание authenticationFailure....12
4.1.6.6. Прерывание egpNeighborLoss....12
4.1.6.7. Прерывание enterpriseSpecific....12
5. Определения....12
6. Благодарности....13
7. Литература....14
8. Вопросы безопасности....14
9. Адреса авторов....14

1. Статус документа

Данный документ является пересмотром RFC 1098 с изменением раздела "Статус документа" и исправлением нескольких опечаток в тексте. Документ определяет простой протокол, который позволяет просматривать и изменять данные системы управления сетью для удаленных узлов. В частности, вместе с дополнительными документами, описывающими структуру данных управления, этот документ обеспечивает простую, работоспособную архитектуру и систему управления для сетей на основе стека протоколов TCP/IP (в частности, Internet).

IAB[1] рекомендует делать все реализации протоколов IP и TCP управляемыми через сеть. Это означает реализацию Internet MIB (RFC-1156) и по крайней мере одного из двух рекомендованных протоколов управления - SNMP (RFC 1157) или CMOT (RFC 1095). Следует отметить, что в настоящее время протокол SNMP является стандартом Internet, а CMOT имеет статус предварительного стандарта (draft standard). Дополнительные сведения о применимости стандартов управления можно найти в документах, описывающих требования к хостам и шлюзам[2]. Текущее состояние стандартизации протоколов можно узнать из документа "IAB Official Protocol Standards"[3].

Документ может распространяться свободно.

2. Введение

Как указано в RFC 1052 (рекомендации IAB для разработки стандартов протоколов сетевого управления Internet[4]) [1] включают два варианта стратегии управления для сетей TCP/IP. В краткосрочной перспективе для управления узлами Internet использовался протокол SNMP[5], а в более долгосрочной перспективе проверялось использование модели сетевого управления OSI. Были разработаны два документа, определяющие данные, используемые для управления: RFC 1065 определяет структуру SMI[6] [2], а RFC 1066, определяет MIB[7] [3]. Оба эти документа были совместимы как с SNMP, так и с моделью сетевого управления OSI.

Эта стратегия оказалась успешной и технологии сетевого управления для Internet были разработаны и реализованы в течение нескольких месяцев. В результате этого фрагменты сети Internet стали управляемыми через сеть в соответствии с реальными потребностями того времени.

Как сказано в RFC 1109 (Report of the Second Ad Hoc Network Management Review Group) [4], различия требований SNMP и модели сетевого управления OSI оказались более значительными, нежели ожидалось. В результате требование совместимости между SMI/MIB и обоими моделями управления было снято. Эта акция перенесла на протокол SNMP ответственность за систему управления для Internet и создание документов, определяющих новые объекты MIB.

По решению IAB спецификации SNMP, SMI и первый вариант Internet MIB получили статус стандартов[8] и рекомендованы к использованию. В соответствии с этим решением IAB рекомендует всем реализациям IP и TCP поддерживать функции сетевого управления. Предполагается, что управляемые через сеть реализации протоколов будут соответствовать спецификациям SMI, MIB и SNMP.

Таким образом, современная модель управления для сетей TCP/IP включает три компоненты - информацию о структуре и идентификации сетей TCP/IP[9], которая описывает как определены управляемые объекты, содержащиеся в MIB (RFC 1155 [5]); базу MIB для управления сетями TCP/IP[10], которая описывает управляемые объекты, содержащиеся в MIB (RFC 1156 [6]); протокол SNMP, который управляет этими объектами в соответствии с приведенной ниже спецификацией.

Как сказано в RFC 1052[4] [1], IAB поручает IETF[11] создать две новых рабочих группы в сфере сетевого управления. Одна группа должна заниматься дальнейшей спецификацией и определением элементов, которые будут включаться в MIB. Другая группа должна модифицировать протокол SNMP с учетом краткосрочных потребностей производителей оборудования и сообщества операторов, а также согласовать свои результаты с результатами группы MIB.

Рабочая группа MIB подготовила два документа. один документ определяет структуру данных управления SMI[12] [2] для использования управляемыми объектами, содержащимися в MIB. Второй документ [3] определяет список управляемых объектов.

Результатом работы группы SNMP Extensions является данный документ, который включает изменения прежних определений SNMP [7], требуемые для согласования с результатами работы группы MIB. Эти изменения нужно было минимизировать в соответствии с директивой IAB, требовавшей от рабочих групп максимального сохранения простоты SNMP. Несмотря на продолжительные дебаты по поводу изменений SNMP, вносимых данным документом, получившийся в результате протокол не обеспечивает обратной совместимости со своим предшественником, протоколом SGMP[13] [8].Хотя синтаксис протокола был изменен, исходная философия и архитектура протокола сохранились. Во избежание путаницы для описанного в этом документе протокола были выделены новые порты UDP.

3. Архитектура SNMP

Применительно к SNMP архитектурная модель представляет собой набор станций сетевого управления и управляемых сетевых элементов. На станциях управления выполняются управляющие программы, которые обеспечивают мониторинг и контроль сетевых элементов. К сетевым элементам относятся хосты, шлюзы, терминальные серверы и другие устройства, включающие программный агент, который отвечает за выполнение функций, запрашиваемых станциями управления. Протокол SNMP используется для обмена информацией между станциями сетевого управления и программными агентами сетевых элементов.

[1]Internet Activities Board

[2]Текущие версии этих требований опубликованы в RFC 1122, 1123 и 1812, переводы которых имеются на сайте www.protocols.ru. *Прим. перев.*

[3]В настоящее время этот документ выпускается под названием “Internet Official Protocol Standards”. *Прим. перев.*

[4]IAB Recommendations for the Development of Internet Network Management Standards – Рекомендации IAB по разработке протоколов сетевого управления для Internet.

[5]Simple Network Management Protocol – простой протокол сетевого управления.

[6] Structure of Management Information – структура данных управления.

[7]Management Information Base – база информации управления.

[8]Standard Protocols

[9]Structure and Identification of Management Information for TCP/IP-based Internets

[10]Management Information Base for Network Management of TCP/IP-based Internets

[11]Internet Engineering Task Force

[12]Structure for Management Information

[13]Simple Gateway Monitoring Protocol – простой протокол мониторинга шлюзов.

3.1. Задачи архитектуры

SNMP неявно минимизирует количество и сложность функций управления, реализуемых самим агентом. Такой подход обусловлен по крайней мере 4 причинами:

(1) снижение расходов на разработку программных агентов управления, требуемых для поддержки протокола;

(2) сложность функций управления возрастает при поддержке их удаленного выполнения и это ведет к значительному расходу ресурсов на реализацию задач управления;

(3) сложность функций управления возрастает при поддержке их удаленного выполнения и это ведет к более жестким ограничениям для систем управления;

(4) упрощенные функции сетевого управления легче понять и использовать разработчикам систем сетевого управления.

Другой задачей протокола является функциональная парадигма расширяемости средств мониторинга и контроля в соответствии с дополнительными и возможно неожиданными аспектами работы сетей и управления ими.

Третьей задачей является обеспечения независимости архитектуры управления от архитектуры и механизмов, используемых конкретными хостами и шлюзами.

3.2. Элементы архитектуры

Архитектура SNMP формулирует решение задач сетевого управления в терминах:

(1) область данных управления, передаваемых протоколом;

(2) представление данных управления, передаваемых протоколом;

(3) операции над данными управления, поддерживаемые протоколом;

(4) форма и смысл обмена данными между объектами системы управления;

(5) определение административных отношений между объектами системы управления;

(6) форма и смысл ссылок на данные управления.

3.2.1. Область данных управления

Область данных управления, передаваемых в процессе работы SNMP в точности представлена экземплярами всех неагрегированных типов объектов, которые определены в стандартных Internet MIB или иных документах, соответствующих соглашениям, установленным в стандарте Internet SMI [5].

Поддержка агрегирования типов объектов MIB не требуется в соответствии с SMI и не реализована в протоколе SNMP.

3.2.2. Представление данных управления

Данные управления (Management information) обмен которыми осуществляется с помощью протокола SNMP, представляются в соответствии с подмножеством языка ASN.1 [9], который указан для определения неагрегированных типов в SMI.

Протокол SGMP адаптировал соглашения об использовании четко определенного (well-defined) подмножества языка ASN.1 [9]. SNMP продолжает и расширяет эту традицию, используя несколько более сложное подмножество ASN.1 для описания управляемых объектов и протокольных модулей данных (PDU), используемых для управления этими объектами. В дополнение к этому желание окончательно перейти к использованию протоколов сетевого управления на базе модели OSI ведет к определению на языке ASN.1 Internet-стандартов для SMI [5] и MIB [6]. Использование языка ASN.1 было отчасти обусловлено его успешным применением в более ранних вариантах (в частности, SGMP). Ограничения на использование ASN.1, которые были введены в SMI для упрощения, подтверждены опытом использования языка с протоколом SGMP.

Из соображений простоты SNMP использует только часть базовых правил представления ASN.1 [10], а именно – представление в форме с определенным размером[14]. Там, где это допустимо, используется представление типа non-constructor. Эти ограничения применимы ко всем аспектам кодирования ASN.1 как для протокольных модулей данных верхнего уровня, так и для содержащихся в PDU объектов данных.

3.2.3. Операции, поддерживаемые для данных управления

SNMP моделирует все функции агента сетевого управления как изменение или проверку значений переменных. Таким образом, объект протокола на логически удаленном хосте (возможно самом сетевом элементе) взаимодействует с агентом управления, располагающимся на сетевом элементе для того, чтобы прочитать (get) или изменить (set) значения переменных. Такая стратегия обеспечивает по крайней мере два преимущества:

(1) число важных функций управления, реализуемых в агенте, сокращается до двух – одна операция присваивает значение указанному параметру, а другая считывает имеющееся значение;

(2) в протокол не требуется вводить определения обязательных (императивных) команд управления – число таких команд на практике всегда растет, а семантика их в общем случае весьма сложна.

Стратегия SNMP заключается в том, что мониторинг состояния сети с любым значимым уровнем детализации выполняется главным образом путем опроса из центра мониторинга. Ограниченное число незапрашиваемых сообщений (trap – прерывание) обеспечивает синхронизацию и активизирует опросы. Ограничение числа незапрашиваемых сообщений согласуется с задачами обеспечения простоты и минимизации трафика, создаваемого системой сетевого управления.

Исключение обязательных команд из числа явно поддерживаемых функций управления не должно препятствовать выполнению тех или иных операций, желательных для агентов управления. В настоящее время большинство команд относится к числу запросов на установку значения того или иного параметра или считывание значения параметра, а функции незначительного числа поддерживаемых сегодня императивных команд могут быть легко реализованы в асинхронном режиме принятой модели управления. В такой схеме императивные команды могут быть реализованы как установка для параметров значений, которые

[14]definite-length form

будут активизировать желаемое действие. Например, вместо реализации команды перезагрузки (reboot), можно просто установить подходящее значение для параметра, определяющего число секунд, по истечении которого система будет перезагружена.

3.2.4. Форма и значение протокольной информации

Обмен данными управления между объектами системы управления реализован в протоколе SNMP с помощью сообщений. Форма и значение этих сообщений подробно рассматриваются в главе 4.

В соответствии с поставленной задачей минимизации сложности агентов управления обмен сообщениями SNMP требует лишь службы передачи дейтаграмм без гарантий доставки и каждое сообщение протокола представляется в виде единственной дейтаграммы. В данном документе рассматривается обмен сообщениями на основе протокола UDP [11], однако механизмы SNMP могут использоваться и с другими транспортными протоколами.

3.2.5. Определение административных отношений[15]

Архитектура SNMP допускает разные административные отношения между объектами, участвующими в работе протокола. Объекты, расположенные на станциях управления и сетевые элементы, обменивающиеся между собой данными по протоколу SNMP называются объектами приложений SNMP[16] или просто приложениями SNMP. Процессы, реализующие SNMP (и, таким образом, поддерживающие объекты приложений SNMP), называются протокольными объектами[17].

Агенты SNMP, связанные с неким множеством объектов приложений SNMP, называются группами SNMP[18]. Каждая группа SNMP идентифицируется символьной строкой, которая называется именем группы[19].

Сообщения SNMP порождаемые приложением SNMP, которое фактически относится к группе SNMP, указанной в компоненте community данного сообщения, называется аутентичным сообщением SNMP[20]. Набор правил, посредством которых проверяется аутентичность сообщений SNMP для конкретной группы SNMP, называется схемой аутентификации. Реализация функции, которая идентифицирует аутентичные сообщения SNMP с использованием одной или нескольких схем аутентификации, называется сервисом аутентификации.

Очевидно, что эффективное управление административными отношениями между приложениями SNMP требует сервиса аутентификации, который (с помощью шифрования или иных методов) способен идентифицировать аутентичные сообщения SNMP с высокой степенью достоверности. Некоторые реализации SNMP могут ограничиваться лишь поддержкой тривиального сервиса аутентификации, который идентифицирует все сообщения SNMP как аутентичные.

Для любого сетевого элемента подмножество объектов в MIB, имеющих отношение к этому элементу, называют представлением SNMP MIB[21]. Отметим, что имена типов объектов из представления SNMP MIB не обязаны относиться к одному субдереву пространства имен типов объектов.

Элементы из набора {READ-ONLY, READ-WRITE} называются режимами доступа SNMP.

Режим доступа SNMP вместе с представлением SNMP MIB называется профилем группы SNMP[22]. Профиль группы SNMP представляет указанные права доступа к переменным заданного представления MIB. Для каждой переменной представления MIB в данном профиле группы SNMP доступ к переменным предоставляется профилем в соответствии с приведенными ниже соглашениями:

(1) если переменная определена в MIB с режимом доступа (Access:) "none", эта переменная не может использоваться в качестве операнда любого оператора;

(2) если переменная определена в MIB с режимом доступа "read-write" или "write-only" и для данного профайла установлен режим READ-WRITE, переменная может служить операндом для get, set и trap;

(3) во всех остальных случаях переменная может использоваться в качестве операнда get и trap.

(4) В тех случаях, когда переменная "write-only"[23] используется в качестве операнда для get или trap, возвращаемое значение зависит от реализации.

Группа SNMP вместе со своим профайлом называется политикой доступа SNMP[24]. Политика доступа представляет профайл, предоставленный агентом SNMP указанной группы для других членов этой группы. Все административные отношения между приложениями SNMP определяются в архитектуре в терминах политики доступа SNMP.

Если сетевой элемент, на котором расположен агент SNMP для указанной группы, не является тем, к которому относится представление MIB для заданного профиля, политика доступа называется “политикой опосредованного доступа”[25]. Агент SNMP, связанный с политикой опосредованного доступа, называется агентом-посредником SNMP[26]. Неаккуратное определение политики опосредованного доступа может приводить к возникновению петель в системе управления, а при корректном определении политика опосредованного доступа может быть весьма полезна:

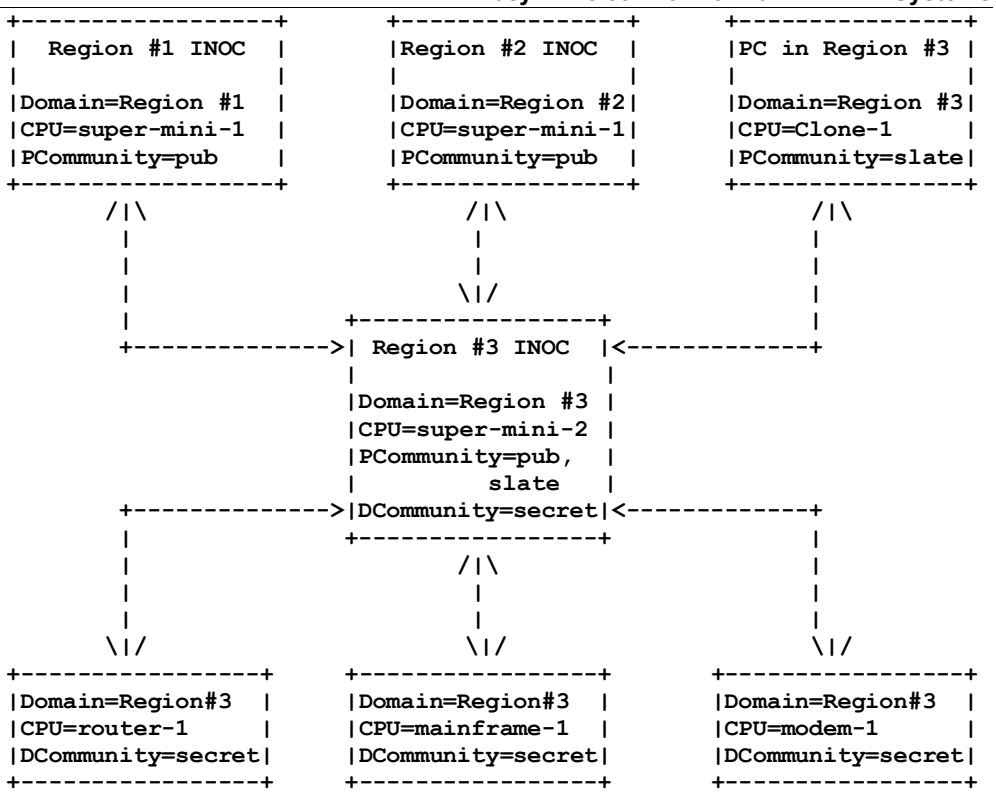
(1) Такая политика позволяет вести мониторинг и контроль для сетевых элементов, которые недоступны (не адресуются) при обычном использовании протокола управления и транспортного протокола. Таким образом, proxy-агент может обеспечивать функции преобразования протокола, позволяющие станции управления использовать согласованную модель управления для всех элементов сети, включая такие устройства, как модемы, мультиплексоры и т. п., которые предназначены для управления с использованием иных моделей.

---

[15] Administrative Relationship
[16] SNMP application entity
[17] SNMP protocol entity
[18] SNMP community
[19] SNMP community name
[20] authentic SNMP message
[21] SNMP MIB view
[22] SNMP community profile
[23] Для переменной можно задать значение, но установленное значение считывать не разрешается. *Прим. перев.*
[24] SNMP access policy
[25] SNMP proxy access policy
[26] SNMP proxy agent

(2) Обеспечивается возможность экранирования сетевых элементов с помощью продуманной политики доступа. Например, агент-посредник может реализовать изощренные механизмы контроля доступа, существенно расширяющие подмножества переменных MIB, которые можно сделать доступными для различных станций управления без усложнения самого сетевого элемента.

В качестве примера на рисунке 1 показаны отношения между станциями управления, proxy-агентами и агентами управления. В этой схеме агент-посредник представлен в виде обычного центра INOC[27] некого административного домена, использующего стандартные отношения со множеством агентов управления.



Domain: административный домен для элемента;

PCommunity: имя группы, использующей агент-посредник

DCommunity: имя прямой группы (direct community)

Рисунок 1: Пример конфигурации сетевого управления

3.2.6. Форма и значение ссылок на управляемые объекты

SMI требует, чтобы определение протокола управления включало:

(1) преобразование неоднозначных (ambiguous) ссылок MIB;

(2) преобразование ссылок MIB при наличии множества версий MIB;

(3) идентификацию отдельных экземпляров типов объектов, определенных в MIB.

3.2.6.1. Преобразование неоднозначных ссылок MIB

Поскольку действие любой операции SNMP концептуально ограничено объектами, имеющими отношение к одному сетевому элементу, и все ссылки SNMP на объекты MIB явно или неявно идентифицируются уникальными именами, не существует возможности преобразования любой ссылки SNMP на любой объект, определенный в MIB, в несколько экземпляров данного типа.

3.2.6.2. Преобразование при наличии множества версий MIB

Экземпляр объекта, указанный любой операцией SNMP, является в точности тем, который указан как часть запроса операции, или (для операции get-next) непосредственно следующий за ним в базе MIB, рассматриваемой в целом. В частности, ссылка на объект как часть некой версии стандартного Internet MIB не преобразуется в какой-либо объект, который не является частью указанной версии стандартного Internet MIB, за исключением случаев, когда запрошена операция get-next и указанный объект является лексически последним среди имен всех объектов, представленных как часть упомянутой версии стандартного Internet MIB.

3.2.6.3. Идентификация экземпляров объекта

Имена для всех типов объектов в MIB определяются явно в стандартных Internet MIB или других документах, соответствующих соглашениям об именовании SMI. Спецификация SMI требует, чтобы протоколы управления определяли механизмы для идентификации отдельных экземпляров этого типа объектов для отдельного сетевого элемента.

Каждый экземпляр любого типа объекта, определенного в MIB, идентифицируется в операциях SNMP уникальным именем, обозначаемым "variable name"[28]. В общем случае имя переменной SNMP является **идентификатором объекта**, задаваемым в формате x.y, где x – имя неагрегированного типа объекта, определенное в MIB, а y – фрагмент **идентификатора объекта**, который указанным для данного типа способом идентифицирует желаемый экземпляр.

Такая стратегия именования допускает наиболее полное использование семантики GetNextRequest-PDU (см. главу 4), поскольку имена для связанных переменных выделяются “непрерывно” в лексикографическом порядке для всех имен переменных, которые известны в MIB.

Обусловленное типом именование экземпляров объектов определено ниже для множества классов типа объектов. Экземпляры типа объекта, для которых не подходит ни одна из описанных ниже схем именования, обозначаются как **идентификаторы объектов** в форме x.0, где x – вышеупомянутое имя типа объекта в определении MIB.

Для примера предположим, что нужно идентифицировать экземпляр переменной sysDescr Класс объекта для sysDescr будет:

```
iso org dod internet mgmt mib system sysDescr
 1   3   6     1      2    1    1       1
```

[27]Internet Network Operations Center – центр управления работой сети.
[28]Имя переменной

Следовательно, тип объекта (x) будет иметь значение 1.3.6.1.2.1.1.1, к которому добавляется в конце значение субидентификатора 0. Таким образом, один из возможных экземпляров sysDescr будет обозначаться как 1.3.6.1.2.1.1.1.0.

3.2.6.3.1. Имена типа ifTable

Имя интерфейса подсети s является **идентификатором объекта** формы i, где i имеет значение экземпляра ifIndex, связанного с s.

Для каждого типа объекта t, для которого определенное имя n имеет префикс ifEntry, экземпляр i типа t именуется **идентификатором объекта** в форме n.s, где s – имя интерфейса подсети, для которого представляется информация.

Предположим в качестве примера, что нужно идентифицировать экземпляр переменной ifType, связанный с интерфейсом 2. Желаемый идентификатор будет иметь вид ifType.2.

3.2.6.3.2. Имена типа atTable

Имя AT-кэшированного[29] сетевого адреса x представляет собой **идентификатор объекта** в форме 1.a.b.c.d, где a.b.c.d – значение (в привычной форме с разделением компонент точками) типа объекта atNetAddress, связанного с x.

Имя эквивалента трансляции адреса представляет собой **идентификатор объекта** в форме s.w, где s – значение экземпляра типа объекта atIndex, связанного с e, а w – имя AT-кэшированного сетевого адреса, связанного с e.

Для каждого типа объекта t, для которого определенное имя n имеет префикс atEntry, экземпляр i типа t именуется **идентификатором объекта** в форме n.y, где y – имя эквивалента трансляции адреса, информацию о котором представляет i.

Предположим в качестве примера, что нужно найти физический адрес записи в таблице трансляции адресов (кэш ARP), связанный с IP-адресом 89.1.1.42 и интерфейсом 3. Желаемый идентификатор будет иметь вид atPhysAddress.3.1.89.1.1.42.

3.2.6.3.3. Имена типа ipAddrTable

Имя сетевого элемента с IP-адресом x **идентификатор объекта** в форме a.b.c.d, где a.b.c.d (в привычной форме с разделением компонент точками) – экземпляр типа объекта ipAdEntAddr, связанного с x.

Для каждого типа объекта t, для которого определенное имя n имеет префикс ipAddrEntry, экземпляр i типа t именуется **идентификатором объекта** в форме n.y, где y – имя сетевого элемента с адресом IP, информацию о котором представляет i.

Предположим в качестве примера, что нужно найти маску записи в таблице интерфейсов IP, связанную с адресом 89.1.1.42. Желаемый экземпляр будет иметь идентификатор ipAdEntNetMask.89.1.1.42.

3.2.6.3.4. Имена типа ipRoutingTable

Имя IP-маршрута x представляет собой **идентификатор объекта** в форме a.b.c.d, где a.b.c.d – значение (в привычной форме с разделением компонент точками) экземпляра типа ipRouteDest, связанного с x.

Для каждого типа объекта t, который имеет определенное имя n с префиксом ipRoutingEntry, экземпляр i типа t именуется **идентификатором объекта** в форме n.y, где y – имя IP-маршрута, для которого i представляет информацию.

Предположим в качестве примера, что нужно найти запись для следующего интервала в таблице пересылки IP, связанную с адресатом 89.1.1.42. Искомый идентификатор будет иметь вид ipRouteNextHop.89.1.1.42.

3.2.6.3.5. Имена типа tcpConnTable

Имя TCP-соединения x представляет собой **идентификатор объекта** в форме a.b.c.d.e.f.g.h.i.j, где a.b.c.d – значение (в привычной форме с разделением компонент точками) экземпляра типа tcpConnLocalAddress, связанного с x, f.g.h.i – значение (в привычной форме с разделением компонент точками) экземпляра типа the tcpConnRemoteAddress, связанного с x, e – значение экземпляра типа tcpConnLocalPort, связанного с x, а j – значение экземпляра типа tcpConnRemotePort, связанного с x.

Для каждого типа объекта t, который имеет определенное имя n с префиксом tcpConnEntry, экземпляр i типа t именуется **идентификатором объекта** в форме n.y, где y – имя соединения TCP, для которого i представляет информацию.

Предположим для примера, что нужно найти состояние соединения TCP между локальным адресом 89.1.1.42 на порту TCP 21 и удаленным адресом 10.0.0.51 на порту TCP 2059. Нужный идентификатор будет иметь вид tcpConnState.89.1.1.42.21.10.0.0.51.2059.

3.2.6.3.6. Имена типа egpNeighTable

Имя EGP-соседа x представляет собой **идентификатор объекта** в форме a.b.c.d, где a.b.c.d – значение (в привычной форме с разделением компонент точками) экземпляра типа egpNeighAddr, связанного с x.

Для каждого типа объекта t, который имеет определенное имя n с префиксом egpNeighEntry, экземпляр i типа t именуется **идентификатором объекта** в форме n.y, где y – имя соседа EGP neighbor, для которого i представляет информацию.

Предположим в качестве примера, что нужно найти состояние соседа для IP-адреса 89.1.1.42. Искомый идентификатор будет иметь вид egpNeighState.89.1.1.42.

4. Спецификация протокола

Протокол сетевого управления представляет собой прикладной протокол, посредством которого переменные MIB агента управления проверяются или изменяются.

Предполагается, что обмен данными между объектами протокола осуществляется путем передачи/приема сообщений, каждое из которых передается независимо от других в форме одной дейтаграммы UDP. В сообщениях используются правила представления ASN.1 (см. параграф 3.2.2). Сообщение включает идентификатор версии, имя группы SNMP и модуль данных протокола (PDU[30]). Объект протокола принимает через порт UDP с номером 161 на хосте, с которым он связан, все сообщения за исключением прерываний (т. е., все сообщения кроме тех, которые содержат Trap-PDU). Сообщения, которые содержат прерывания, следует принимать через порт UDP с номером 162. Реализация протокола не обязана принимать сообщения, размер которых превышает 484 октета. Однако по возможности желательна поддержка дейтаграмм больших размеров.

[29] AT – address translation. Адрес кэшированный в системе трансляции. *Прим. перев.*

[30] protocol data unit

От всех реализаций протокола SNMP требуется поддержка 5 типов PDU: GetRequest-PDU, GetNextRequest-PDU, GetResponse-PDU, SetRequest-PDU и Trap-PDU.

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

 IMPORTS
      ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
              FROM RFC1155-SMI;
          -- сообщение верхнего уровня
          Message ::=
                  SEQUENCE {
                       version        -- номер версии (1 для данной спецификации)
                          INTEGER {
                              version-1(0)
                          },

                      community      -- имя группы
                          OCTET STRING,

                      data           -- например, PDU в простейшем случае
                          ANY        -- будет использоваться аутентификация
                  }
          -- модули данных протокола
          PDUs ::=
                  CHOICE {
                      get-request
                          GetRequest-PDU,

                      get-next-request
                          GetNextRequest-PDU,

                      get-response
                          GetResponse-PDU,

                      set-request
                          SetRequest-PDU,

                      trap
                          Trap-PDU
                       }

-- отдельные PDU и наиболее часто
-- используемые данные рассматриваются ниже

END
```

4.1. Элементы процедуры

В этом параграфе рассматриваются действия протокольного объекта, реализующего SNMP. Отметим однако, что это описание не является рекомендациями по внутренней архитектуре реализации протокола, соответствующей данной спецификации.

В последующем тексте используется термин “транспортный адрес”. Для случая UDP этот адрес состоит из IP-адреса и номера порта UDP. Однако для поддержки SNMP могут использоваться и другие типы транспортного сервиса. В этих случаях определение транспортного адреса изменяется в соответствии с используемым протоколом.

Операции верхнего уровня для протокольного объекта, генерирующего сообщения, включают:

(1) Сначала создается соответствующий модуль данных PDU (например, GetRequest-PDU) как объект ASN.1.

(2) Созданный объект ASN.1 вместе с именем группы и транспортными адресами отправителя и получателя передается службе, которая реализует желаемую схему аутентификации. Сервис аутентификации возвращает другой объект ASN.1.

(3) После этого протокольный объект создает объект ASN.1 Message, используя имя группы и полученный после аутентификации объект ASN.1.

(4) Новый объект ASN.1 преобразуется в последовательность данных с использованием базовых правил представления ASN.1 и объект передается с использованием транспортного сервиса удаленному партнеру.

На приемной стороне операции верхнего уровня включают:

(1) Простейший анализ принятой дейтаграммы для построения объекта ASN.1, соответствующего объекту ASN.1 Message. При неудачной попытке анализа дейтаграмма отбрасывается без выполнения дальнейших операций.

(2) Далее проверяется номер версии SNMP. Если версии не совпадают, дейтаграмма отбрасывается без выполнения дальнейших операций.

(3) После этого протокольный элемент передает имя группы и пользовательские данные из объекта ASN.1 Message вместе с транспортными адресами отправителя и получателя службе, реализующей нужную схему аутентификации. После аутентификации возвращается другой объект ASN.1 или информация об отказе. В последнем случае протокольный элемент отмечает этот отказ, генерирует прерывание (необязательно) и отбрасывает дейтаграмму без выполнения дальнейших операций.

(4) Протокольный элемент выполняет простейший анализ объекта ASN.1, возвращенного после аутентификации, для построения объекта ASN.1, соответствующего объекту ASN.1 PDU. При неудачной попытке анализа дейтаграмма отбрасывается без выполнения последующих операций. В остальных случаях с использованием имени группы SNMP выбирается подходящий профиль и выполняется обработка PDU. Если в результате этой обработки возвращается сообщение, то адрес транспортный отправителя для этого сообщения должен совпадать с транспортным адресом получателя исходного запроса.

4.1.1. Конструкции общего назначения

Перед описанием типов PDU, используемых протоколом, полезно рассмотреть некоторые их часто используемых конструкций ASN.1:

```
-- данные запроса/отклика
RequestID ::= INTEGER

ErrorStatus ::=
       INTEGER {
              noError(0),
              tooBig(1),
              noSuchName(2),
              badValue(3),
              readOnly(4)
              genErr(5)
       }

       ErrorIndex ::= INTEGER

-- связка
VarBind ::=
       SEQUENCE {
              name
                     ObjectName,

              value
                     ObjectSyntax
       }

       VarBindList ::=
              SEQUENCE OF
                     VarBind
```

Идентификаторы запросов RequestID используются для того, чтобы различать переданные запросы. Используя RequestID, приложение SNMP может также находить корреляции между входящими откликами и запросами, отклики на которые еще не получены. При использовании транспортного сервиса без гарантий доставки дейтаграмм RequestID обеспечивает также простой способ детектирования дубликатов.

Отличное от нуля значение ErrorStatus показывает, что в процессе обработки запроса произошла та или иная ошибка. В таких случаях значение ErrorIndex может обеспечивать дополнительную информацию об ошибке, показывая номер переменной (в списке), с которой связана ошибка.

Термин переменная используется для обозначения экземпляра управляемого объекта. Связка[31] или VarBind – это пара, включающая имя переменной и ее значение. VarBindList представляет собой простой список имен переменных и соответствующих значений. Некоторые PDU содержат только имя переменной без ее значения (например, GetRequest-PDU). В этом случае относящаяся к значению часть связки игнорируется протокольным объектом. Однако относящаяся к значению часть связки все равно должна использовать корректный синтаксис и представление ASN.1. Рекомендуется для таких связок указывать значение ASN.1 NULL.

4.1.2. GetRequest-PDU

GetRequest-PDU имеет формат:

```
GetRequest-PDU ::= [0]
       IMPLICIT SEQUENCE {
              request-id
                     RequestID,

              error-status        -- всегда 0
                     ErrorStatus,

              error-index         -- всегда 0
                     ErrorIndex,

              variable-bindings
                     VarBindList
       }
```

Модули данных GetRequest-PDU генерируются протокольным объектом только по запросу приложения SNMP.

При получении GetRequest-PDU протокольный объект отвечает в соответствии со всеми применимыми правилами из приведенного ниже списка:

[31] Variable binding

(1) Если для любого объекта, поименованного в связке “имя-значение”, имя объекта не совпадает в точности с именем одного из объектов, доступных для операции get в соответствующем представлении MIB, принимающий объект передает отправителю запроса аналогичное сообщение GetResponse-PDU, указывая в поле кода ошибки значение noSuchName (нет такого имени), а в поле error-index – индекс имени вышеупомянутого объекта в полученном сообщении.

(2) Если для любого объекта, поименованного в связке “имя-значение”, объект представляет собой *агрегированный* тип (в соответствии с определением SMI), принимающий объект передает отправителю аналогичное сообщение GetResponse-PDU, указав в поле error-status значение noSuchName, а в поле error-index – индекс имени вышеупомянутого объекта в принятом сообщении.

(3) Если размер GetResponse-PDU, созданного в соответствии с приведенным ниже описанием, превышает локальное ограничение, принимающий объект передает отправителю аналогичное сообщение GetResponse-PDU, указав в поле error-status значение tooBig, а в поле error-index - 0.

(4) Если для любого объекта, поименованного в связке “имя-значение”, значение объекта не может быть найдено по причинам, которые отличаются от перечисленных выше, принимающий объект передает отправителю аналогичное сообщение GetResponse-PDU, указав в поле error-status значение genErr, а в поле error-index – индекс имени вышеупомянутого объекта в полученном сообщении.

Если ни одно из перечисленных выше правил не применимо, принимающий протокольный объект передает отправителю полученного сообщения такое сообщение GetResponse-PDU, в котором для каждого объекта, поименованного в связке “имя-значение” полученного сообщения, соответствующая компонента GetResponse-PDU представляет имя и значение переменной. В поле error-status помещается значение noError, а в поле the error-index - 0. Значение поля request-id в ответном сообщении GetResponse-PDU совпадает с идентификатором в принятом сообщении.

4.1.3. GetNextRequest-PDU

Формат GetNextRequest-PDU отличается от формата GetRequest-PDU лишь полем индикации типа PDU. ASN.1-форма имеет вид:

```
GetNextRequest-PDU ::= [1]
       IMPLICIT SEQUENCE {
              request-id
                     RequestID,
              error-status        -- всегда 0
                     ErrorStatus,

              error-index         -- всегда 0
                     ErrorIndex,

              variable-bindings
                     VarBindList
       }
```

Модули данных GetNextRequest-PDU генерируются протокольным объектом только по запросу приложения SNMP.

При получении GetNextRequest-PDU протокольный объект отвечает в соответствии со всеми применимыми правилами из приведенного ниже списка:

(1) Если для любого объекта, поименованного в связке “имя-значение”, имя объекта не совпадает в точности с именем одного из объектов, доступных для операции get в соответствующем представлении MIB, принимающий объект передает отправителю запроса аналогичное сообщение GetResponse-PDU, указывая в поле кода ошибки значение noSuchName (нет такого имени), а в поле error-index – индекс имени вышеупомянутого объекта в полученном сообщении.

(2) Если размер GetResponse-PDU, созданного в соответствии с приведенным ниже описанием, превышает локальное ограничение, принимающий объект передает отправителю аналогичное сообщение GetResponse-PDU, указав в поле error-status значение tooBig, а в поле error-index - 0.

(3) Если для любого объекта, поименованного в связке “имя-значение”, значение объекта не может быть найдено по причинам, которые отличаются от перечисленных выше, принимающий объект передает отправителю аналогичное сообщение GetResponse-PDU, указав в поле error-status значение genErr, а в поле error-index – индекс имени вышеупомянутого объекта в полученном сообщении.

Если ни одно из перечисленных выше правил не применимо, принимающий протокольный объект передает отправителю полученного сообщения такое сообщение GetResponse-PDU, в котором для каждого объекта, поименованного в связке “имя-значение” полученного сообщения, соответствующая компонента GetResponse-PDU представляет имя и значение переменной. В поле error-status помещается значение noError, а в поле the error-index - 0. Значение поля request-id в ответном сообщении GetResponse-PDU совпадает с идентификатором в принятом сообщении.

4.1.3.1. Пример обхода таблицы

Одним из основных вариантов применения GetNextRequest-PDU является обход[32] концептуальных таблиц информации в MIB. Семантика этого типа сообщений SNMP вкупе с обеспечиваемыми протоколом механизмами идентификации отдельных экземпляров типа объекта обеспечивает возможность доступа к связанным объектам MIB как при просмотре строк таблицы.

Пример обмена данными SNMP, приведенный ниже, показывает как приложение SNMP может определить адрес получателя и следующий маршрутизатор (next hop gateway) для каждой записи в таблице маршрутизации конкретного элемента сети. Предположим, что таблица маршрутизации включает три записи:

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

[32]Просмотр множества значений. *Прим. перев.*

Станция управления передает агенту SNMP пакет GetNextRequest-PDU, содержащий значения **идентификаторов объектов** как имена запрашиваемых переменных:

```
GetNextRequest ( ipRouteDest, ipRouteNextHop, ipRouteMetric1 )
```

Агент SNMP возвращает сообщение GetResponse-PDU:

```
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 ))
```

Станция управления генерирует следующее сообщение:

```
GetNextRequest ( ipRouteDest.9.1.2.3,  ipRouteNextHop.9.1.2.3, ipRouteMetric1.9.1.2.3 )
```

Агент SNMP возвращает:

```
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 ))
```

Станция управления передает еще одно сообщение:

```
GetNextRequest ( ipRouteDest.10.0.0.51, ipRouteNextHop.10.0.0.51, ipRouteMetric1.10.0.0.51 )
```

Агент SNMP возвращает отклик:

```
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 ))
```

Станция управления передает очередной запрос:

```
GetNextRequest ( ipRouteDest.10.0.0.99, ipRouteNextHop.10.0.0.99, ipRouteMetric1.10.0.0.99 )
```

Поскольку в таблице маршрутизации больше нет записей, агент SNMP возвращает тот объект, который является следующим в лексическом порядке имен известных объектов. Такой отклик говорит управляющей станции о том, что в таблице маршрутизации больше нет записей.

4.1.4. GetResponse-PDU

Формат GetResponse-PDU аналогичен формату GetRequest-PDU и отличается лишь идентификатором типа. Представление ASN.1 имеет вид:

```
GetResponse-PDU ::= [2]
       IMPLICIT SEQUENCE {
              request-id
                     RequestID,
              error-status
                     ErrorStatus,

              error-index
                     ErrorIndex,

              variable-bindings
                     VarBindList
       }
```

Сообщения GetResponse-PDU генерируются протокольными объектами только при получении запросов GetRequest-PDU, GetNextRequest-PDU или SetRequest-PDU, описанных в этом документе.

При получении GetResponse-PDU принимающая сторона представляет содержимое сообщения приложению SNMP.

4.1.5. SetRequest-PDU

Формат сообщений SetRequest-PDU аналогичен формату GetRequest-PDU и отличается лишь значением идентификатора типа. Представление ASN.1 имеет вид:

```
SetRequest-PDU ::= [3]
       IMPLICIT SEQUENCE {
              request-id
                     RequestID,

              error-status        -- всегда 0
                     ErrorStatus,

              error-index         -- всегда 0
                     ErrorIndex,

             variable-bindings
                     VarBindList
       }
```

Сообщения SetRequest-PDU генерируются протокольным объектом только по запросу приложения SNMP.

При получении SetRequest-PDU принимающий объект отвечает на это сообщение в соответствии с приведенными ниже правилами:

(1) Если для любого объекта, поименованного в связке “имя-значение”, запись (операция set) не поддерживается в имеющих отношение к делу представлениях MIB, принимающий объект передает отправителю запроса аналогичное сообщение GetResponse-PDU, указывая в поле кода ошибки значение noSuchName (нет такого имени), а в поле error-index – индекс имени вышеупомянутого объекта в полученном сообщении.

(2) Если для любого объекта, поименованного в поле связки “имя-значение”, содержимое этого поля не соответствует в точности представлению ASN.1, типу, размеру и значению, которые согласуются с запрошенной переменной, принимающий объект передает отправителю запроса аналогичное сообщение GetResponse-PDU, указывая в поле кода ошибки значение badValue (нет такого имени), а в поле error-index – индекс имени вышеупомянутого объекта в полученном сообщении.

(3) Если размер GetResponse-PDU, созданного в соответствии с приведенным ниже описанием, превышает локальное ограничение, принимающий объект передает отправителю аналогичное сообщение GetResponse-PDU, указав в поле error-status значение tooBig, а в поле error-index – 0.

(4) Если для любого объекта, поименованного в связке “имя-значение”, значение объекта не может быть изменено по причинам, которые отличаются от перечисленных выше, принимающий объект передает отправителю аналогичное сообщение GetResponse-PDU, указав в поле error-status значение genErr, а в поле error-index – индекс имени вышеупомянутого объекта в полученном сообщении.

Если неприменимо ни одно из перечисленных выше правил, тогда для каждого объекта, поименованного в поле связки “имя-значение” принятого сообщения, заданное значение присваивается указанной переменной. Каждое присваивание значения переменной должно выполняться с учетом значений других переменных, устанавливаемых этим же сообщением (как будто присваиваются значения всем переменным разом).

После этого принявший сообщение объект передает отправителю сообщение GetResponse-PDU, аналогичное полученному запросу с установкой в поле error-status значения noError, а в поле error-index – значение 0.

4.1.6. Trap-PDU

Сообщения Trap-PDU имеют формат:

```
Trap-PDU ::= [4]

        IMPLICIT SEQUENCE {
                enterprise          -- тип объекта, генерирующего прерывание
                                    -- см. sysObjectID в документе [5]
                OBJECT IDENTIFIER,

                agent-addr          -- адрес объекта,
                NetworkAddress,     -- генерирующего прерывание

                generic-trap        -- тип прерывания
                        INTEGER {
                                coldStart(0),
                                warmStart(1),
                                linkDown(2),
                                linkUp(3),
                                authenticationFailure(4),
                                egpNeighborLoss(5),
                                enterpriseSpecific(6)
                        },

                specific-trap       -- специфический код прерывания,
                        INTEGER,    -- указываемый даже в тех случаях,
                                    -- когда прерывание не относится к enterpriseSpecific

                time-stamp          -- время, прошедшее с момента (ре) инициализации
                TimeTicks,          -- сетевого элемента до генерации прерывания

                variable-bindings   -- представляющая интерес информация
                VarBindList
        }
```

Сообщения Trap-PDU генерируются протокольными объектами только по запросу приложения SNMP. Это означает, что адрес получателя, задаваемый приложением SNMP, зависит от реализации.

При получении сообщения Trap-PDU принимающий протокольный объект представляет содержимое этого сообщения своему приложению SNMP.

Значение поля variable-bindings в сообщениях Trap-PDU зависит от реализации.

Базовые типы прерываний описаны ниже

4.1.6.1. Прерывание coldStart

Прерывание coldStart(0) говорит, что передающий сообщение протокольный объект заново инициализирует себя (перезагружается) и, следовательно, конфигурация агента или реализация протокольного объекта может измениться.

4.1.6.2. Прерывание warmStart

Прерывание warmStart(1) сообщает, что протокольный объект заново инициализирует себя, но конфигурация агента и реализация протокольного объекта не будут изменяться.

4.1.6.3. Прерывание linkDown

Прерывание linkDown(2) сообщает, что передающий объект обнаружил отказ на одном из коммуникационных каналов, представленных в конфигурации агента.

Сообщения Trap-PDU типа linkDown содержат в качестве первого элемента variable-bindings имя и значение экземпляра ifIndex для соответствующего интерфейса.

4.1.6.4. Прерывание linkUp

Прерывание linkUp(3) говорит о том, что передающий сообщение объект детектировал активизацию одного из коммуникационных каналов, указанных в конфигурации агента.

Сообщения Trap-PDU типа linkUp содержат в качестве первого элемента variable-bindings имя и значение экземпляра ifIndex для соответствующего интерфейса.

4.1.6.5. Прерывание authenticationFailure

Прерывание authenticationFailure(4) говорит о том, передающий сообщение протокольный объект получил протокольное сообщение с некорректной аутентификацией. Хотя реализации SNMP должны поддерживать генерацию прерываний этого типа, они должны также обеспечивать механизм отключения генерации таких прерываний.

4.1.6.6. Прерывание egpNeighborLoss

Прерывание egpNeighborLoss(5) говорит о том, что сосед EGP, для которого передающий протокольный объект был партнером (EGP peer), помечен как неработающий (down and the peer relationship no longer obtains.

Сообщения Trap-PDU типа egpNeighborLoss содержат в качестве первого элемента variable-bindings имя и значение экземпляра egpNeighAddr для соответствующего соседа.

4.1.6.7. Прерывание enterpriseSpecific

Прерывание enterpriseSpecific(6) говорит о том, что передающий протокольный объект обнаружил некоторые события типа enterprise-specific. Поле specific-trap идентифицирует прерывание более точно.

5. Определения

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

        IMPORTS
                ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
                        FROM RFC1155-SMI;

                -- сообщение верхнего уровня

                Message ::=
                        SEQUENCE {
                                version              -- номер версии (1 для данной спецификации)
                                        INTEGER {
                                                version-1(0)
                                        },

                                community            -- имя группы (community name)
                                        OCTET STRING,

                                data                 -- например, PDU (при использовании
                                        ANY          -- тривиальной аутентификации)
                        }

                -- 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            -- иногда игнорируется
                        INTEGER {
                                noError(0),
                                tooBig(1),
                                noSuchName(2),
                                badValue(3),
                                readOnly(4),
                                genErr(5)
                        },

                error-index             -- иногда игнорируется
                        INTEGER,

                variable-bindings       -- эти значения иногда игнорируются
                        VarBindList
        }

Trap-PDU ::= [4]
        IMPLICIT SEQUENCE {
                enterprise              -- тип объекта, генерирующего  generating
                                        -- прерывание (см. sysObjectID в [5])


                OBJECT IDENTIFIER,
        agent-addr                      -- адрес объекта, генерирующего
                NetworkAddress,         -- прерывание

        generic-trap                    -- тип прерывания
                INTEGER {
                        coldStart(0),
                        warmStart(1),
                        linkDown(2),
                        linkUp(3),
                        authenticationFailure(4),
                        egpNeighborLoss(5),
                        enterpriseSpecific(6)
                },

        specific-trap           -- специфический код, включаемый даже для прерываний,
                INTEGER,        -- не относящихся к числу enterpriseSpecific

        time-stamp              -- время, прошедшее с момента (ре)инициализации
                TimeTicks,      -- сетевого объекта до момента генерации прерывания

        variable-bindings       -- представляющая интерес информация
                VarBindList
}

-- variable bindings
VarBind ::=
        SEQUENCE {
                name
                        ObjectName,

                value
                        ObjectSyntax
        }

VarBindList ::=
        SEQUENCE OF
                VarBind

END
```

6. Благодарности

В подготовке этого документа участвовали члены рабочей группы IETF SNMP Extensions:

Karl Auerbach, Epilogue Technology

K. Ramesh Babu, Excelan

Amatzia Ben-Artzi, 3Com/Bridge

Lawrence Besaw, Hewlett-Packard

Jeffrey D. Case, University of Tennessee at Knoxville

Anthony Chung, Sytek

James Davidson, The Wollongong Group

James R. Davin, MIT Laboratory for Computer Science

Mark S. Fedor, NYSERNet

Phill Gross, The MITRE Corporation

Satish Joshi, ACC

Dan Lynch, Advanced Computing Environments

Keith McCloghrie, The Wollongong Group

Marshall T. Rose, The Wollongong Group (председатель)

Greg Satz, cisco

Martin Lee Schoffstall, Rensselaer Polytechnic Institute

Wengyik Yeong, NYSERNet

7. Литература

[1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, IAB, April 1988.

[2] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", RFC 1065, TWG, August 1988.

[3] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1066, TWG, August 1988.

[4] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, IAB, August 1989.

[5] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, Performance Systems International and Hughes LAN Systems, May 1990.

[6] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets", RFC 1156, Hughes LAN Systems and Performance Systems International, May 1990.

[7] Case, J., M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol", Internet Engineering Task Force working note, Network Information Center, SRI International, Menlo Park, California, March 1988.

[8] Davin, J., J. Case, M. Fedor, and M. Schoffstall, "A Simple Gateway Monitoring Protocol", RFC 1028, Proteon, University of Tennessee at Knoxville, Cornell University, and Rensselaer Polytechnic Institute, November 1987.

[9] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.

[10] Information processing systems - Open Systems Interconnection, "Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)", International Organization for Standardization, International Standard 8825, December 1987.

[11] Postel, J., "User Datagram Protocol", RFC 768[33], USC/Information Sciences Institute, November 1980.

8. Вопросы безопасности

Вопросы безопасности в этом документе не рассматриваются.

9. Адреса авторов

Jeffrey D. Case

SNMP Research

P.O. Box 8593

Knoxville, TN 37996-4800

Phone: (615) 573-1434

Email: case@CS.UTK.EDU

Mark Fedor

Performance Systems International

Rensselaer Technology Park

125 Jordan Road

Troy, NY 12180

Phone: (518) 283-8860

[33]Перевод этого документа имеется на сайте www.protocols.ru, *Прим. перев.*

Email: fedor@patton.NYSER.NET

Martin Lee Schoffstall

Performance Systems International

Rensselaer Technology Park

165 Jordan Road

Troy, NY 12180

Phone: (518) 283-8860

Email: schoff@NISC.NYSER.NET

James R. Davin

MIT Laboratory for Computer Science, NE43-507

545 Technology Square

Cambridge, MA 02139

Phone: (617) 253-6020

EMail: jrd@ptt.lcs.mit.edu

Перевод на русский язык

Николай Малых
nmalykh@bilim.com