УДК 681.3.06

Закрытое акционерное общество
Институт инфотелекоммуникаций

М.Д. Рожнов, Ю.М. Шерстюк

Протокол управления XMLNMP

Версия 1.0

Санкт-Петербург

2010

Содержание

Стр.

Перечень сокращений

АУ	– агент управления
ОМУ	– объект мониторинга и управления
DTD	– Document Type Definition
EMS	– Elements Management System
MIB	– Management Information Block
SNMP	– Simple Network Management Protocol
TMN	– Telecommunication Management Network
URI	– Uniform Resource Identifier
XML	– eXtensible Markup Language
XMLNMP	– XML Network Management Protocol

Предисловие

Протокол SNMP (Simple Network Management Protocol) версий 1 и 2 (v1 и v2c), получивший большое распространение как протокол управления для средств телекоммуникаций и активных элементов вычислительных сетей обладает рядом известных недостатков. К числу основных недостатков SNMP обычно относят следующие:

- низкие скорость опроса значений и коэффициент использования транспортный среды;
- ограниченность допустимых типов данных значений объектов информационной базы управления;
- низкая безопасность протокола;
- возможность потерь данных вследствие применения в качестве транспортного протокола UDP.

По этой причине в 2006 в ходе разработки ряда агентов управления для средств инфотелекоммуникаций в ЗАО «Институт инфотелекоммуникаций» («ИНФОТЕК») авторами был предложен и апробирован протокол XMLNMP.

Последующий положительный опыт применения XMLNMP и рост интереса к нему со стороны различных организаций – разработчиков привели к осознанию необходимости публикации его описания в виде отдельной брошюры.

Следует отметить, что в ЗАО «Институт инфотелекоммуникаций» разработан ряд программных средств, существенно упрощающих разработку XMLNMP-агентов и менеджеров с использованием языка программирования *Python*, собранных в единое изделие “Библиотека поддержки протокола XMLNMP” (*Рожнов М.Д. Библиотека поддержки протокола XMLNMP*. Свидетельство о государственной регистрации программы для ЭВМ № 2008613106 от 27.06.2008 (Федеральная служба по интеллектуальной собственности, патентам и товарным знакам)).

Взаимодействие с агентами управления, поддерживающими протокол XMLNMP, в полной мере реализовано в изделиях ЗАО «ИНФОТЕК»:

- “Технологическое управление инфотелекоммуникациями”;
- “Средства тестирования наборов данных технологического управления”;
- “Комплекс средств подготовки наборов данных технологического управления”.

Общие сведения о протоколе XMLNMP впервые опубликованы в материалах X Санкт-Петербургской международной конференции “Региональная информатика – 2006”: *Рожнов М.Д.* XMLNMP - протокол управления сетевыми элементами // X Санкт-Петербургская международная конференция "Региональная информатика – 2006 (РИ-2006)", Санкт-Петербург, 24-26 октября 2006г.: Материалы конференции. – СПб.: СПОИСУ, 2006. - с. 91-92.

1. Назначение и основные возможности протокола

Протокол XMLNMP (XML Network Management Protocol) представляет собой протокол сетевого управления на основе расширяемого языка разметки XML (eXtensible Markup Language).

Протокол относится к протоколам прикладного уровня модели взаимодействия открытых систем.

Протокол поддерживает модель взаимодействия “агент – менеджер” и ориентирован на применение в системах управления, в которых управляющая система представлена программой – менеджером, а управляемая система представлена программой – агентом. В качестве управляемой системы выступает объект мониторинга и управления (ОМУ) – рис. 1.1.

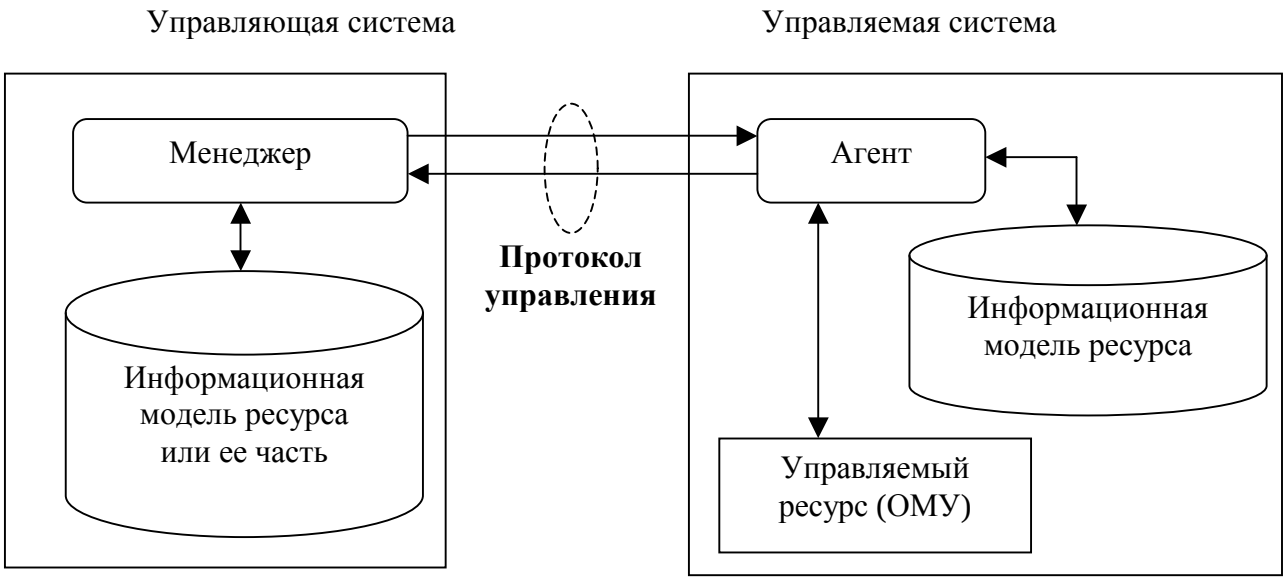

Рисунок 1.1

Считается, что роль менеджера выполняет программное приложение, которое с помощью соответствующих запросов следит за текущим состоянием ОМУ, выдает команды на его изменение (команды управления) и принимает уведомления. Приложение, установленное на элементе вычислительной (телекоммуникационной) сети, отвечающие на запросы о состоянии, выполняющее команды управления и посылающее уведомления, выполняет роль агента (агента управления). Таким образом, агент управления (АУ) предоставляет возможность доступа управляющей системе к ОМУ, который рассматривается как некий ресурс, имеющий свою информационную модель.

Информационная модель ресурса представима базой информации управления (Management Information Block (MIB)), поддерживаемой АУ, и значениями параметров ресурса, имеющимися в MIB (параметрами MIB). В информационную модель могут входить и параметры собственно АУ.

Менеджер устанавливает взаимосвязь с агентом для мониторинга ОМУ и оказания на него управляющих воздействий. Возможное нарушение такой взаимосвязи может быть обнаружено обеими сторонами. Мониторинг осуществляется путем опроса значений параметров MIB АУ. Кроме того, АУ может сообщать об изменении в состоянии ОМУ путем отправки извещений менеджерам, зарегистрированным агентом. Управление осуществляется путем установки значений параметров MIB АУ.

Протокол предусматривает следующие основные виды взаимодействий между агентом и менеджером:

- запрос менеджером сведений о наличии и характеристиках параметров, имеющихся в MIB АУ, и значений этих параметров;
- запрос менеджера на установку новых значений параметров MIB и отчет АУ о результатах установки этих значений;
- извещение менеджера агентом об изменении состояния ОМУ (о событиях в функционировании ОМУ и, возможно, АУ, существенных с позиций мониторинга и управления ресурсом и функционирования АУ).

Основные свойства и особенности протокола заключаются в следующем:

- обмен данными между АУ и менеджером представляет собой передачу специальным образом отформатированных структур данных, представленных на языке, основанном на XML;
- в качестве транспорта для передачи XML-пакетов используется протокол HTTP версий 1.0 и 1.1 (возможен обмен данными через HTTP-proxy серверы);
- аутентификация агента менеджером и менеджера агентом осуществляется с помощью средств протокола HTTP;
- предусмотрен широкий набор допустимых типов данных параметров MIB;
- АУ способен сообщать метаданные о поддерживаемых им параметрах MIB;
- имеется возможность получить и изменить значения одного или нескольких параметров MIB (включая таблицы и их части как единые объекты) с помощью одного запроса;
- существует возможность передачи в извещении таблиц и их частей;
- используется расширенная диагностика возможных ошибок взаимодействия менеджера и АУ;
- предусмотрена возможность извещения менеджера о том, что из-за проблем с транспортом данных в сети им не получены извещения, сформированные АУ.

Основной областью применения протокола является его использование в системах управления элементами (Elements Management System (EMS)), предусмотренными для уровня управления элементами в логической многоуровневой архитектуре сети управления электросвязью (Telecommunication Management Network (TMN)).

2. Транспортный протокол

Общая схема взаимодействия между агентом и менеджером заключается в следующем:

- менеджер формирует запрос, передает его агенту, ожидает и получает ответ агента (результат выполнения запроса);
- агент формирует извещение (являющееся запросом агента к менеджеру), передает его менеджеру, ожидает и получает ответ менеджера, свидетельствующий о том, что извещение принято менеджером.

В случае, когда время ожидания ответа на запрос (уведомление) превысило некую установленную величину, отправителем фиксируется факт отсутствия связи с получателем.

Действия менеджера и агента управления при отсутствии (пропадании) связи с агентом определяются алгоритмом его функционирования, зависящим от реализации.

Протокол XMLNMP предполагает, что обмен данными между АУ и менеджером осуществляется путем пересылки данных, представленных в соответствии с нотацией XML-приложения, синтаксис которого в форме Document Type Definition (DTD) приведен в приложении. Для обмена данными в качестве транспортного протокола должен использоваться протокол HTTP версии 1.0 или 1.1. Возможен обмен данными через HTTP-proxy сервера.

Для выполнения операции обмена данными инициатором взаимодействия должен использоваться запрос *POST* протокола HTTP, в котором *Uniform Resource Identifier* (URI) определяет адрес обработчика запроса. Заголовок *Content-type* должен содержать значение *text/xml*.

Аутентификация агента менеджером и менеджера агентом должна осуществляться с помощью средств протокола HTTP. В случае ошибки аутентификации получателем должны возвращаться коды 401 и 407.

При отправке запроса менеджером к агенту последний должен вернуть код, характеризующий результат обработки HTTP-запроса на уровне транспортного протокола:

- код *200* свидетельствует об успешном выполнении запроса;
- код *500* возвращается в случае внутренней ошибки агента;
- код *400* следует интерпретировать как внутреннюю ошибку менеджера;
- остальные коды ошибок следует рассматривать как ошибки транспорта.

При отправке запроса агентом менеджеру последний должен вернуть код, характеризующий результат обработки HTTP-запроса на уровне транспортного протокола:

- код *204* свидетельствует об успешном выполнении запроса;
- код *500* возвращается в случае внутренней ошибки менеджера;
- код *400* следует интерпретировать как внутреннюю ошибку агента;
- остальные коды ошибок следует рассматривать как ошибки транспорта.

3. Модель данных

3.1. Параметры информационной модели

В рамках протокола XMLNMP предполагается, что АУ поддерживает информационную модель ОМУ, которую можно рассматривать как упрощенный вариант MIB, определенной в RFC 1155 – существенным считается только тот факт, что MIB содержит ряд параметров, каждый из которых может быть скалярным параметром (скаляром) или табличным параметром (таблицей).

Скалярный параметр идентифицируется своим именем и имеет некоторое значение. В зависимости от характеристики доступности скалярного параметра менеджер может иметь возможность только запрашивать его значение (доступ на чтение, ***readonly***) или запрашивать и изменять его значение (доступ на чтение и запись, ***readwrite***). Характеристика доступности значений скаляра для менеджера указывается в MIB.

Табличный параметр представляет собой список строк, количество которых потенциально произвольно. Таблица идентифицируется своим именем и может допускать добавление и/или удаление строк (поддержка операций добавления/удаления строк является одной из характеристик таблицы).

Таблица содержит фиксированное количество столбцов, идентифицируемых своими именами. Перечень и порядок следования столбцов таблицы задается в MIB. В таблице должен присутствовать по крайней мере один столбец, являющийся индексным. Идентификация строки таблицы осуществляется путем перечисления значений ее ячеек, принадлежащих всем индексным столбцам таблицы в порядке следования индексных столбцов, предусмотренном в MIB. Перечисление имен индексных столбцов в предусмотренном в MIB порядке их следования представляет собой индекс таблицы. Значения строк таблицы должны быть упорядочены по значениям индекса таблицы.

Столбец таблицы идентифицируется своим именем и имеет некоторое значение, которое по аналогии со скалярным параметром может быть доступно менеджеру только на чтение или на чтение и запись. Характеристика доступности значений столбца для менеджера указывается в MIB. Для столбца таблицы может быть предусмотрено значение "по умолчанию", подлежащее использованию при добавлении строк в таблицу для заполнения соответствующей ячейки добавляемой строки.

Ячейка таблицы идентифицируется именем столбца и значением индекса таблицы для строки, в которой находится данная ячейка.

Строку, столбец и ячейку таблицы наряду со скалярами и таблицами также можно рассматривать как параметр MIB. Соответственно, скаляры и ячейки таблицы являются элементарными параметрами, а строки, столбцы и таблицы – структурированными.

Множество имен скаляров, таблиц и их столбцов образует единое пространство имен параметров, поэтому указанные имена должны быть уникальными в пределах MIB.

3.2. Типы данных

Для каждого скалярного параметра и столбца таблицы в MIB задается тип данных, который определяет ограничения допустимых значений и правила их представления с использованием XML.

Типы данных, предусмотренные в протоколе XMLNMP (типы данных протокола), приведены в таблице 3.1.

Разработчик АУ может определить в MIB и реализовать в АУ дополнительные (пользовательские) типы данных, являющиеся производными от типов данных протокола или пользовательских типов данных, определенных в той же MIB ранее. Определение пользовательского типа данных позволяет ужесточить ограничения на допустимые значения. Тип данных протокола, от которого в конечном итоге порожден пользовательский тип данных, является родительским протокольным типом данных для этого пользовательского типа.

Значения элементарных параметров передаются между агентом и менеджером как содержимое тега **val**. Синтаксис тега **val**:

```
<!ELEMENT val (#PCDATA)>
<!ATTLIST val type CDATA #REQUIRED>
```

В качестве значения атрибута ***type*** передается наименование типа данных протокола. Если для описания типа данных скаляра или ячейки таблицы используется пользовательский тип данных, то в качестве значения атрибута указывается тип данных протокола, являющийся предком этого типа данных.

Таблица 3.1

Типы данных протокола XMLNMP

Наименование типа данных	Описание типа данных	Представление с использованием XML
Integer	Целое значение в диапазоне $(-2^{128}) \div (2^{128}-1)$	Последовательность десятичных цифр – возможно, с предшествующим знаком
Unsigned	Неотрицательное целое значение в диапазоне $0 \div (2^{128}-1)$	Последовательность десятичных цифр
Float	Вещественное значение в диапазоне -1.79769313486231157e308÷ 1.79769313486231157e308	Значение представляется в научной нотации: последовательность десятичных цифр (возможно со знаком и точкой, разделяющей целую и дробную части числа), соответствующая мантиссе числа, символ ‘e’ (строчная латинская буква *e*), последовательность десятичных цифр (возможно со знаком), соответствующая порядку числа
String	Последовательность от 0 до 65536 печатных (отображаемых) символов	Последовательность печатных символов, соответствующих кодировке, указанной в заголовке XML
Bytes	Последовательность от 0 до 65536 байтов	Строка, содержащая последовательность байтов, закодированная с ис-

Наименование типа данных	Описание типа данных	Представление с использованием XML
		пользованием алгоритма *base64*

Продолжение табл. 3.1

Наименование типа данных	Описание типа данных	Представление с использованием XML
Set	Множество (перечисление) возможных целых значений (до 256 элементов перечисления), каждому из которых поставлена в соответствие строка (до 255 символов). Параметр такого типа может иметь значение, равное одному из возможных значений или сумме двух и более возможных значений (при условии, что каждое возможное значение входит в сумму не более одного раза)	Последовательность чисел (каждое представлено последовательностью десятичных цифр), составляющих значение параметра, разделенных запятыми (символ ‘,’)
Date	Дата в диапазоне от 01/01/1 до 31/12/9999	Строка формата *YYYYMMDD*, где *YYYY* – четыре десятичных цифры, соответствующие году, *MM* – две десятичные цифры, соответствующие месяцу, *DD* – две десятичные цифру соответствующие дню месяца
Time	Время суток с точностью до секунды	Строка формата *hh:mm:ss*, где *hh* – две десятичные цифры, соответствующие часам, *mm* – две десятичные цифры, соответствующие минутам, *ss* – две десятичные цифры, соответствующие секундам, *:* - символ (‘:’)
DateTime	Дата и время суток	Строка формата *YYYMMDDThh:mm:ss*, где *YYYY* – четыре десятичных цифры, соответствующие году, *MM* – две десятичные цифры, соответствующие месяцу, *DD* – две десятичные цифру соответствующие дню месяца, *hh* – две десятичные цифры, соответствующие часам, *mm* – две десятичные цифры, соответствующие минутам, *ss* – две десятичные цифры, соответствующие секундам, *T* – символ ‘T’ (заглавная латинская буква *T*), : - символ (‘:’)

Окончание табл. 3.1

Наименование типа данных	Описание типа данных	Представление с использованием XML
TimeTicks	Неотрицательное целое, показывающее время в сотых долях секунды от некоторой опорной точки. Имеет значение в диапазоне $0 \div 2^{32}$-1.	Последовательность десятичных цифр
Enumerated	Множество (перечисление) возможных целых значений (до 256 элементов перечисления), каждому из которых поставлена в соответствие строка (до 255 символов)	Последовательность десятичных цифр, представляющая целое, соответствующее одному из допустимых значений
Counter	Счетчик. Монотонно увеличивается. При достижении максимального значения переходит в 0 и продолжает увеличиваться. Имеет неотрицательное целое значение в диапазоне $0 \div 2^{32}$-1.	Последовательность десятичных цифр
Counter64	Счетчик. Монотонно увеличивается. При достижении максимального значения переходит в 0 и продолжает увеличиваться. Имеет неотрицательное целое значение в диапазоне $0 \div 2^{64}$-1.	Последовательность десятичных цифр
Gauge	Может увеличиваться и уменьшатся. При достижении максимального значения перестает изменяться – «защелкивается». Имеет неотрицательное целое значение в диапазоне $0 \div 2^{32}$-1.	Последовательность десятичных цифр

3.3. Неопределенные значения

В качестве результата на запрос значения для скаляров и ячеек таблиц, не являющихся индексными, агент может вернуть неопределенное значение. Это означает, что значение соответствующего параметра либо не определено, либо не имеет смысла в комбинации с текущими значениями других параметров. Неопределенное значение передается с использованием тега **nil**, который не имеет атрибутов и вложенных тегов. Синтаксис тега **nil**:

```
<!ELEMENT nil EMPTY>
```

3.4. Ограничения значений

При описании типов данных, определяемых пользователем, а также типов данных значений скаляров и столбцов таблицы, могут быть указаны дополнительные ограничения значений.

Ограничения значений могут ужесточать требования к значениям, определяемые родительским типом данных.

В зависимости от используемого типа данных протокола или родительского типа используемого пользовательского типа данных могут применяться ограничения на диапазон значений и ограничения на перечисляемые значения.

Для значений типов данных *Integer*, *Unsigned*, *Counter*, *Counter64*, *Gauge*, *Float*, *Date*, *Time*, *DateTime*, *TimeTicks*, *String*, *Bytes* и производных от них применяются ограничения на диапазон значений. При этом для типов *String* и *Bytes* эти ограничения интерпретируются как ограничения на длину соответствующих величин, а для всех остальных типов – как собственно ограничения значений.

Ограничения на диапазон значений описываются последовательностью тегов range. Тег range не имеет атрибутов и содержит пару вложенных тегов – **min** и **max**, определяющих минимальное и максимальное значение диапазона соответственно. Значения **min** и max включаются в диапазон и являются допустимыми значениями. Если в качестве ограничения значений указывается последовательность диапазонов, то диапазоны не должны пересекаться, т.е. максимальное значение предшествующего диапазона не может быть больше или равным минимального значения последующего диапазона. Значения диапазонов не могут выходить за пределы диапазонов родительского типа данных.

Синтаксис тегов **range**, **min** и **max**:

```
<!ELEMENT range (min, max)>
<!ELEMENT min (val)>
<!ELEMENT max (val)>
```

Для значений типов данных, производных от *Enumerated* и *Set*, применяются ограничения на перечисляемые значения. Собственно типы *Enumerated* и *Set* не могут использоваться для описания скаляров и столбцов таблиц, т.к. для них не заданы изначальные ограничения на перечисляемые значения. Набор перечисляемых значений должен являться подмножеством перечисляемых значений родительского типа данных.

Ограничения на перечисляемые значения описываются последовательностью тегов **enumItem**. Тег **enumItem** не содержит вложенных тегов.

Тег **enumItem** имеет три атрибута:

- ***name*** – наименование элемента перечисления;
- ***value*** – числовое значение элемента перечисления;
- ***hint*** – неформальное описание элемента перечисления, которое может быть использовано для отображения в качестве подсказки (не является обязательным).

Синтаксис тега **enumItem**:

```
<!ELEMENT enumItem EMPTY>
<!ATTLIST enumItem %name;
                   value CDATA #REQUIRED
                   %hint;>
```

4. Коды ошибок

В процессе обработки агентом запросов на чтение и установку значений параметров могут возникать ошибки. Агент сообщает менеджеру об ошибке, возвращая соответствующий числовой код.

Коды ошибок и их описания перечислены в таблице 4.1.

Сведения об ошибке передаются между агентом и менеджером с использованием тега **error**. Информация о коде ошибки предается значением атрибута ***code***.

Синтаксис тега **error**:

```
<!ELEMENT error EMPTY>
<!ATTLIST error code CDATA #REQUIRED>
```

Таблица 4.1

Ошибки, определяемые агентом XMLNMP

Код ошибки	Символическое имя	Описание ошибки
1	NO_SUCH_NAME	Отсутствует параметр или столбец таблицы с указанным в запросе именем
2	READ_ONLY	Для соответствующего скаляра или столбца таблицы не предусмотрена установка новых значений
3	WRONG_TYPE	Недопустимый тип значения
4	WRONG_VALUE	Недопустимое значение (значение не удовлетворяет установленным ограничениям)
5	WRONG_ROW_INDEX	Ошибка в формате индекса строки таблицы (не указаны все значения индексных столбцов, указаны значения столбцов, не являющихся индексными и т.п.)
6	NO_CREATION	Таблица не допускает создания строк
7	NO_DELETION	Таблица не допускает удаления строк
8	OPERATION_ERROR	Ошибка агента в процессе выполнения запроса, внешняя по отношению к агенту (ошибка базы данных, сервера приложений и т.п.)
9	AGENT_ERROR	Внутренняя ошибка агента

5. Запросы, предусмотренные протоколом

5.1. Запрос информации о параметрах, поддерживаемых агентом

Запрос представляется тегом **getDefinitions**, который не имеет атрибутов и не содержит вложенных тегов.

Синтаксис тега **getDefinitions**:

```
<!ELEMENT getDefinitions EMPTY>
```

Ответ представляется тегом **definitions**. Тег не имеет атрибутов. Тег содержит необязательный тег **userTypeDefinitions**, тег **variableDefinitions** и необязательный тег **notificationDefinitions**.

Синтаксис тега **definitions**:

```
<!ELEMENT definitions (userTypeDefinitions?,
variableDefinitions, notificationDefinitions?)>
```

Тег **userTypeDefinitions** служит для описания типов данных, определяемых пользователем. Тег содержит непустую последовательность тегов **userTypeDef**, каждый из которых определяет один тип, определяемый пользователем.

Синтаксис тега **userTypeDefinitions**:

```
<!ELEMENT userTypeDefinitions (userTypeDef)+>
```

Тег **userTypeDef** описывает тип, определяемый пользователем. Тег может содержать последовательность тегов **enumItem** или последовательность тегов **range**, накладывающих дополнительные ограничения на родительский тип данных.

Тег **userTypeDef** имеет три атрибута:

- **name** – наименование типа, определяемого пользователем;
- **parent** – наименование родительского типа. Родительским типом может быть основной тип данных или тип данных, определяемый пользователем;
- **hint** – неформальное описание типа данных, которое может быть использовано для отображения в качестве подсказки (не является обязательным).

Синтаксис тега **userTypeDef**:

```
<!ELEMENT userTypeDef (enumItem*|range*)>
<!ATTLIST userTypeDef %name;иparent CDATA #REQUIREDм%hint;>
```

Тег **variableDefinitions** служит для описания параметров, поддерживаемых агентом. Тег содержит непустую последовательность тегов **scalarDef** или **tableDef**, каждый из которых описывает скаляр или таблицу соответственно.

Синтаксис тега **variableDefinitions**:

```
<!ELEMENT variableDefinitions (scalarDef|tableDef)+>
```

Тег **scalarDef** описывает параметр-скаляр. Тег может содержать последовательность тегов **enumItem** или последовательность тегов **range**, накладывающих дополнительные ограничения на значение параметра.

Тег **scalarDef** имеет четыре атрибута:

- **name** – наименование скаляра;
- **type** – тип данных скаляра;
- **access** – тип доступа к данным (доступен скаляр только на чтение или на чтение и запись);

- **hint** – неформальное описание скаляра, которое может быть использовано для отображения в качестве подсказки (не является обязательным).

Синтаксис тега **scalarDef**:

```
<!ELEMENT scalarDef (enumItem*|range*)>
<!ATTLIST scalarDef %name; type CDATA #REQUIRED
                    %access; %hint;>
```

Тег **tableDef** описывает таблицу и содержит непустую последовательность тегов **columnDef**, а также непустую последовательность тегов **indexItem**.

Тег **tableDef** имеет четыре атрибута:

- **name** – наименование таблицы;
- **createRow** – допускается ли создание строк таблицы;
- **deleteRow** – допускается ли удаление строк таблицы;
- **hint** – неформальное описание таблицы, которое может быть использовано для отображения в качестве подсказки (не является обязательным).

Синтаксис тега **tableDef**:

```
<!ELEMENT tableDef (columnDef+, indexItem+)>
<!ATTLIST tableDef %name;
                   createRow (yes|no) "no"
                   deleteRow (yes|no) "no"
                   %hint;>
```

Тег **columnDef** описывает столбец таблицы. Тег может содержать последовательность тегов **enumItem** или последовательность тегов **range**, накладывающих дополнительные ограничения на значения ячеек данного столбца таблицы.

Тег **columnDef** имеет четыре атрибута:

- **name** – наименование столбца;
- **type** – тип данных столбца;
- **access** – тип доступа к данным (доступен столбец только на чтение или на чтение и запись);
- **hint** – неформальное описание столбца, которое может быть использовано для отображения в качестве подсказки (не является обязательным).

Синтаксис тега **columnDef**:

```
<!ELEMENT columnDef (enumItem*|range*)>
<!ATTLIST columnDef %name; type CDATA #REQUIRED
                    %access; %hint;>
```

Последовательность тегов **indexItem** описывает индекс параметра-таблицы. Тег **indexItem** имеет единственный атрибут – **name**, указывающий имя столбца таблицы.

Синтаксис тега **indexItem**:

```
<!ELEMENT indexItem EMPTY>
<!ATTLIST indexItem %name;>
```

Тег **notificationDefinitions** служит для описания извещений, с помощью которых агент может информировать менеджера об изменении состояния сетевого элемента. Тег содержит непустую последовательность тегов **notifDef**, каждый из которых определяет один вид извещений.

Синтаксис тега **notificationDefinitions**:

```
<!ELEMENT notificationDefinitions (notifDef)+>
```

Тег **notifDef** описывает извещение. Тег может содержать последовательность тегов **varName**, указывающих имена параметров, ассоциированных с извещением.

Тег **notifDef** имеет два атрибута:

- **name** – наименование извещения;

- **hint** – неформальное описание извещения, которое может быть использовано для отображения в качестве подсказки (не является обязательным).

Синтаксис тега **notifDef**:

```
<!ELEMENT notifDef (varName)*>
<!ATTLIST notifDef %name; %hint;>
```

Тег **varName** указывает имя параметра, ассоциированного с извещением. Тег **varName** имеет единственный атрибут – **name**, указывающий имя параметра.

Синтаксис тега **varName**:

```
<!ELEMENT varName EMPTY>
<!ATTLIST varName %name;>
```

Пример сведений о параметрах, поддерживаемых агентом.

Пусть агент управления поддерживает:

- два определяемых пользователем типа данных (*Boolean* и *NonZeroInteger*);

- скаляры *boolFlag* и *count*, а также таблицу *portActivity*, содержащую три столбца *interface*, *portNo* и *active*, два из которых (*interface*, *portNo*) определяют индекс.

Пусть этот же агент управления может посылать менеджеру извещения двух типов: *variableChanges* и *anotherNotification*.

Ответ такого АУ на запрос информации о данных, поддерживаемых агентом, в предположении о наличии соответствующих подсказок и возможных значений по своему содержанию может быть следующим:

```
<?xml version='1.0' encoding='windows-1251'?>
<definitions>
  <userTypeDefinitions>
    <userTypeDef name='Boolean' parent='Enumerated'
  hint='Логическое'>
      <enumItem name='false' value='0' hint='ложь'/>
      <enumItem name='true' value='1' hint='истина'/>
    </userTypeDef>
    <userTypeDef name='NonZeroInteger' parent='Integer'
  hint='Ненулевое целое'>
      <range>
        <min> <val type='Integer'>-2147483648</val> </min>
        <max> <val type='Integer'>-1</val> </max>
      </range>
      <range>
        <min> <val type='Integer'>1</val> </min>
        <max> <val type='Integer'>2147483647</val> </max>
      </range>
    </userTypeDef>
  </userTypeDefinitions>
  <variableDefinitions>
    <scalarDef name='boolFlag' type='Boolean'
  access='readonly'/>
```

```
      <scalarDef name='count' type='Unsigned' access='readwrite'
    hint='какой-то счетчик'>
        <range>
          <min> <val type='Unsigned'>0</val> </min>
          <max> <val type='Unsigned'>32767</val> </max>
        </range>
        </scalarDef>
      <tableDef name='portActivity' createRow='yes'
    deleteRow='no' hint='активность портов'>
        <columnDef name='interface' type='String'/>
        <columnDef name='portNo' type='Unsigned'>
          <range>
            <min> <val type='Unsigned'>0</val> </min>
            <max> <val type='Unsigned'>32767</val> </max>
          </range>
        </columnDef>
        <columnDef name='active' type='Boolean'
      access='readwrite' hint='признак активности'/>
          <indexItem name='interface'/>
          <indexItem name='portNo'/>
      </tableDef>
    </variableDefinitions>
    <notificationDefinitions>
      <notifDef name='variableChanges'>
        <varName name='count'/>
        <varName name='portActivity'/>
      </notifDef>
      <notifDef name='anotherNotification' hint='другое уведомле-
    ние'/>
    </notificationDefinitions>
  </definitions>
```

5.2. Запрос на получение значений параметров

Запрос представляется тегом **getVariables**. Тег содержит непустую последовательность тегов **scalarReq** и/или **tableReq** и не имеет атрибутов.

Синтаксис тега **getVariables**:

```
<!ELEMENT getVariables (scalarReq|tableReq)+>
```

Тег **scalarReq** описывает запрашиваемый параметр-скаляр. Тег имеет единственный атрибут – **name**, указывающий имя параметра.

Синтаксис тега **scalarReq**:

```
<!ELEMENT scalarReq EMPTY>
<!ATTLIST scalarReq %name;>
```

Тег **tableReq** описывает запрашиваемый параметр-таблицу. Тег содержит возможно пустую последовательность тегов **colName**, а также возможно пустую последовательность тегов **colFilter**. Тег имеет единственный атрибут **name**, указывающий имя параметра-таблицы.

Синтаксис тега **tableReq**:

```
<!ELEMENT tableReq (colName*, colFilter*)>
<!ATTLIST tableReq %name;>
```

Последовательность тегов **colName** указывает запрашиваемые столбцы таблицы. Если последовательность пуста, то это означает, что запрашиваются все столбцы таблицы.

Тег **colName** указывает имена столбцов таблицы. Тег имеет единственный атрибут – **name**, указывающий имя параметра.

Синтаксис тега **colName**:

```
<!ELEMENT colName EMPTY>
<!ATTLIST colName %name;>
```

Последовательность тегов **colFilter** указывает условия фильтрации строк таблицы. Если последовательность пуста, то это означает, что запрашиваются все строки таблицы. Тег **colFilter** указывает возможные значения одного столбца таблицы. Тег имеет единственный атрибут – ***name***, указывающий имя столбца таблицы. В зависимости от типа данных столбца тег содержит непустую последовательность тегов **val**, либо непустую последовательность тегов **range**.

Синтаксис тега **colFilter**:

```
<!ELEMENT colFilter (val+|range+)>
<!ATTLIST colFilter %name;>
```

Пример 1. Менеджер запрашивает у агента значение скаляра *count* и таблицы *portActivity*:

```
<?xml version='1.0' encoding='windows-1251'?>
<getVariables>
<scalarReq name='count'/>
<tableReq name='portActivity'/>
</getVariables>
```

Пример 2. Менеджер запрашивает у агента значения скаляров *sysUpTime*, *boolFlag* и *count*, а также столбцов *interface* и *active* таблицы *portActivity*:

```
<getVariables>
<scalarReq name='sysUpTime'/>
<scalarReq name='boolFlag'/>
<scalarReq name='count'/>
<tableReq name='portActivity'>
<colName name='interface'/>
<colName name='active'/>
</tableReq>
</getVariables>
```

Пример 3. Менеджер запрашивает у агента значения скаляров *sysUpTime*, *boolFlag* и *count*, а также столбцов *interface*, *portNo* и *active* таблицы *portActivity*. Из таблицы отбираются только строки, у которых столбец *interface* принимает значения *127.0.0.1* или *192.168.0.101*, а значение столбца *portNo* находится в одном из диапазонов *80-82* или *8080-8082*. Содержание такого запроса должно быть следующим:

```
<getVariables>
  <scalarReq name='sysUpTime'/>
  <scalarReq name='boolFlag'/>
  <scalarReq name='count'/>
  <tableReq name='portActivity'>
       <colName name='interface'/>
       <colName name='portNo'/>
       <colName name='active'/>
```

```
        <colFilter name='interface'>
          <val type='String'>127.0.0.1</val>
          <val type='String'>192.168.0.101</val>
        </colFilter>
        <colFilter name='portNo'>
          <range>
            <min> <val type='Unsigned'>80</val> </min>
            <max> <val type='Unsigned'>82</val> </max>
          </range>
          <range>
            <min> <val type='Unsigned'>8080</val> </min>
            <max> <val type='Unsigned'>8082</val> </max>
          </range>
        </colFilter>
    </tableReq>
  </getVariables>
```

Пример 4. Менеджер запрашивает у агента значения скаляров *sysUpTime*, *boolFlag* и *count*, а также всех столбцов таблицы *portActivity*. Из таблицы отбираются только строки, у которых столбец *interface* принимает значения *127.0.0.1*, а значение столбца *portNo* находится в диапазоне *80-82*. Содержание такого запроса должно быть следующим:

```
<getVariables>
  <scalarReq name='sysUpTime'/>
  <scalarReq name='boolFlag'/>
  <scalarReq name='count'/>
  <tableReq name='portActivity'>
    <colFilter name='interface'>
      <val type='String'>127.0.0.1</val>
    </colFilter>
    <colFilter name='portNo'>
      <range>
      <min> <val type='Unsigned'>80</val> </min>
      <max> <val type='Unsigned'>82</val> </max>
      </range>
    </colFilter>
  </tableReq>
</getVariables>
```

Ответ представляется тегом **variables**, который не имеет атрибутов. Тег содержит непустую последовательность тегов **var**.

Синтаксис тега **variables**:

```
<!ELEMENT variables (var)+>
```

Тег **var** описывает значение возвращаемого параметра и содержит один из следующих тегов: **val**, **nil**, **error**, **rows**.

Синтаксис тега **variables**:

```
<!ELEMENT var (val|nil|error|rows)>
```

Тег **nil** возвращается в случае, если значение параметра не определено, не установлено или не имеет смысла при определенной комбинации других значений переменных.

Синтаксис тега **nil**:

```
<!ELEMENT nil EMPTY>
```

Тег **error** возвращается в случае ошибки при обработке агентом запроса на чтение параметра. Тег не имеет подчиненных тегов и имеет единственный атрибут ***code***, описывающий код ошибки, возникшей при выполнении запроса на чтение.

Синтаксис тега **error**:

```
<!ELEMENT error EMPTY>
<!ATTLIST error code CDATA #REQUIRED>
```

Тег **val** возвращается в ответ на успешный запрос параметра-скаляра. Содержимое тега **val** соответствует значению параметра.

Тег **rows** возвращается в ответ на успешный запрос параметра-таблицы. Тег содержит возможно пустую последовательность тегов **row**.

Синтаксис тега **rows**:

```
<!ELEMENT rows (row*)>
```

Тег **row** описывает строку запрашиваемой параметра-таблицы. Тег содержит непустую последовательность из тегов **val**, **nil** и **error**.

Синтаксис тега **row**:

```
<!ELEMENT row (val|nil|error)+>
```

Тег **nil** возвращается, если значение ячейки таблицы не определено, не установлено или не имеет смысла при определенной комбинации других значений переменных.

Тег **error** возвращается в случае возникновения ошибки при обработке агентом запроса на чтение соответствующей ячейки таблицы.

Тег **val** возвращается в ответ на успешный запрос ячейки таблицы. Содержимое тега **val** соответствует значению ячейки таблицы.

Пример 1. Результатом выполнения запроса на чтение параметров являются три скаляра, а также ячейки из двух таблиц. Для первой таблицы агент вернул значения трех столбцов для четырех строк, а для второй таблицы – значения одного столбца для четырех строк.

```
<?xml version='1.0' encoding='windows-1251'?>
<variables>
  <var> <val type='TimeTicks'>10529</val> </var>
  <var> <val type='Enumerated'>1</val> </var>
  <var> <val type='Unsigned'>0</val> </var>
  <var>
    <rows>
      <row>
        <val type='String'>127.0.0.1</val>
        <val type='Unsigned'>21</val>
        <val type='Enumerated'>1</val>
      </row>
      <row>
        <val type='String'>127.0.0.1</val>
        <val type='Unsigned'>80</val>
        <val type='Enumerated'>0</val>
      </row>
      <row>
        <val type='String'>192.168.0.101</val>
        <val type='Unsigned'>21</val>
        <val type='Enumerated'>1</val>
```

```
        </row>
        <row>
          <val type='String'>192.168.0.101</val>
          <val type='Unsigned'>80</val>
          <val type='Enumerated'>1</val>
        </row>
      </rows>
    </var>
    <var>
      <rows>
        <row> <val type='Enumerated'>1</val> </row>
        <row> <val type='Enumerated'>0</val> </row>
        <row> <val type='Enumerated'>1</val> </row>
        <row> <val type='Enumerated'>1</val> </row>
      </rows>
    </var>
  </variables>
```

Пример 2. Агент сообщил, что не поддерживает параметр с именем, указанным в первой позиции списка запроса, вернул значение параметра, указанного во второй позиции списка запроса, сообщил, что при получении параметра, указанного в третьей позиции списка запроса, произошла внутренняя ошибка. Для таблицы, указанной в четвертой позиции списка запроса, агент вернул значения первого и второго столбцов, и сообщил, что при получении третьего столбца (во всех строках) произошла внутренняя ошибка. В данном случае ответ может иметь следующий вид:

```
  <?xml version='1.0' encoding='windows-1251'?>
  <variables>
    <var> <error code='1'/> </var>
    <var>
      <val type='Enumerated'>1</val>
    </var>
    <var> <error code='9'/> </var>
    <var>
      <rows>
        <row>
          <val type='String'>127.0.0.1</val>
          <val type='Unsigned'>21</val>
          <error code='1'/>
        </row>
        <row>
          <val type='String'>127.0.0.1</val>
          <val type='Unsigned'>80</val>
          <error code='1'/>
        </row>
        <row>
          <val type='String'>192.168.0.101</val>
          <val type='Unsigned'>21</val>
          <error code='1'/>
        </row>
        <row>
          <val type='String'>192.168.0.101</val>
          <val type='Unsigned'>80</val>
```

```
          <error code='1'/>
        </row>
      </rows>
    </var>
  </variables>
```

Пример 3. Агент сообщил, что для параметра, указанного в первой позиции списка запроса, значение не установлено, вернул значение для переменной, указанной во второй позиции списка запроса, и сообщил, что при получении переменной, указанной в третьей позиции списка запроса, произошла внутренняя ошибка. Для таблицы, указанной в четвертой позиции списка запроса, агент вернул значения первого и второго столбцов, и сообщил, что для третьего столбца (во всех строках) значение не установлено. В данном случае ответ может иметь следующий вид:

```
<?xml version='1.0' encoding='windows-1251'?>
<variables>
  <var>
    <nil/>
  </var>
  <var>
    <val type='Enumerated'>1</val>
  </var>
  <var>
    <error code='9'/>
  </var>
  <var>
    <rows>
      <row>
        <val type='String'>127.0.0.1</val>
        <val type='Unsigned'>21</val>
      <nil/>
      </row>
      <row>
        <val type='String'>127.0.0.1</val>
        <val type='Unsigned'>80</val>
        <error code='1'/>
      </row>
    <row>
      <val type='String'>192.168.0.101</val>
      <val type='Unsigned'>21</val>
      <nil/>
    </row>
    <row>
      <val type='String'>192.168.0.101</val>
      <val type='Unsigned'>80</val>
      <nil/>
    </row>
    </rows>
  </var>
</variables>
```

5.3. Запрос на установку значений параметров

Запрос представляется тегом **setVariables**, который содержит непустую последовательность тегов **scalar** и **table** и не имеет атрибутов.

Синтаксис тега **setVariables**:

```
<!ELEMENT setVariables (scalar|table)+>
```

Тег **scalar** описывает устанавливаемое значение параметра-скаляра. Тег содержит тег **val**, содержащий новое значение параметра, и имеет единственный атрибут ***name***, определяющей имя параметра.

Синтаксис тега **scalar**:

```
<!ELEMENT scalar (val)>
<!ATTLIST scalar %name;>
```

Тег **table** содержит непустую последовательность тегов **rowUpd**. Тег имеет единственный атрибут ***name***, определяющей имя параметра-таблицы.

Синтаксис тега **table**:

```
<!ELEMENT table (rowUpd)+>
<!ATTLIST table %name;>
```

Тег **rowUpd** содержит данные, необходимые для модификации одной строки таблицы. Тег содержит тег **index**, либо тег **values**, либо пару тегов **index** и **values**. Если указан единственный тег **index**, то строка таблицы с указанным индексом будет удалена. Если указан единственный тег **values**, то в таблицу будет добавлена строка с перечисленными значениями. Если указана пара тегов **index** и **values**, то в строке, определяемой указанным индексом, будут модифицированы значения, указанные в теге **values**.

Синтаксис тега **rowUpd**:

```
<!ELEMENT rowUpd (index|values|(index, values))>
```

Тег **index** содержит непустую последовательность тегов **cell**.

Синтаксис тега **index**:

```
<!ELEMENT index (cell)+>
```

Тег **values** содержит непустую последовательность тегов **cell**.

Синтаксис тега **values**:

```
<!ELEMENT values (cell)+>
```

Тег **cell** содержит тег v**al**, определяющий значение ячейки таблицы, и имеет единственный атрибут **name**, определяющий имя столбца таблицы.

Синтаксис тега **cell**:

```
<!ELEMENT cell (val)>
<!ATTLIST cell %name;>
```

Пример. Менеджер предписывает агенту установку двух скаляров *boolFlag* и *count*, а также модификацию трех строк таблицы *portActivity*:

- в строке с индексом *interface* равным *127.0.0.1*, и *portNo* равным *80*, значение столбца *active* устанавливается в *1*;
- удаляется строка с индексом *interface*, равным *192.168.0.101*, и *portNo*, равным *80*;
- добавляется строка с индексом *interface*, равным *192.168.0.111*, *portNo* равным *99*.

В данном случае запрос должен иметь следующий вид:

```
<?xml version='1.0' encoding='windows-1251'?>
<setVariables>
  <scalar name='boolFlag'>
    <val type='Enumerated'>1</val>
  </scalar>
  <scalar name='count'>
    <val type='Unsigned'>111</val>
  </scalar>
  <table name='portActivity'>
    <rowUpd>
      <index>
        <cell name='interface'>
          <val type='String'>127.0.0.1</val>
        </cell>
        <cell name='portNo'>
          <val type='Unsigned'>80</val>
        </cell>
      </index>
      <values>
        <cell name='active'>
          <val type='Enumerated'>1</val>
        </cell>
      </values>
    </rowUpd>
    <rowUpd>
      <index>
        <cell name='interface'>
        <val type='String'>192.168.0.101</val>
        </cell>
        <cell name='portNo'>
          <val type='Unsigned'>80</val>
        </cell>
      </index>
    </rowUpd>
    <rowUpd>
      <values>
        <cell name='interface'>
          <val type='String'>192.168.0.111</val>
        </cell>
        <cell name='portNo'>
          <val type='Unsigned'>99</val>
        </cell>
        <cell name='active'>
          <val type='Enumerated'>1</val>
        </cell>
      </values>
    </rowUpd>
  </table>
</setVariables>
```

Ответ представляется тегом **setResult**. Тег не имеет атрибутов и содержит тег **success** или тег **fault**.

Синтаксис тега **setResult**:

```
<!ELEMENT setResult (success|fault)>
```

В случае успешного выполнения запроса на модификацию переменных тег **setResult** содержит тег **success**.

Тег **success** не имеет атрибутов и не содержит вложенных тегов.

Синтаксис тега **success**:

```
<!ELEMENT success EMPTY>
```

Модификация значений переменных осуществляется в порядке, соответствующем запросу на модификацию. В случае возникновения ошибки при модификации переменной дальнейшие модификации не производятся, и тег **setResult** содержит тег **fault**.

Тег **fault** имеет три атрибута:

- **name** – наименование параметра, при модификации которого произошла ошибка;
- **code** – код ошибки;
- **rowUpdNumber** – позиция строки в запросе на модификацию таблицы, при модификации которой произошла ошибка (указывается, если ошибка произошла при модификации переменной-таблицы и нумеруется от 0).

Синтаксис тега **fault**:

```
<!ELEMENT fault EMPTY>
<!ATTLIST fault %name;
                code CDATA #REQUIRED
                rowUpdNumber CDATA #IMPLIED>
```

Пример 1. Для всех переменных новые значения были установлены успешно.

```
<?xml version='1.0' encoding='windows-1251'?>
<setResult>
  <success/>
</setResult>
```

Пример 2. При установке значения переменной *portActivity* успешно были изменены первые две строки из запроса на модификацию, модификация третьей строки не произошла в связи с ошибками в указании индекса в запросе на модификацию. Параметры (скаляры или таблицы), запрос на модификацию которых следовал до запроса на модификацию *portActivity* в списке переменных запроса, были модифицированы успешно.

```
<?xml version='1.0' encoding='windows-1251'?>
<setResult>
  <fault name='portActivity' code='5' rowUpdNumber='2'/>
</setResult>
```

5.4. Извещения

Извещение описывается тегом **notification**. Тег имеет единственный атрибут **name**, указывающий имя извещения. Тег может содержать последовательность тегов **scalar** или **table**, описывающих изменения значений переменных агента.

Синтаксис тега **notification**:

```
<!ELEMENT notification (scalar|table)*>
<!ELEMENT notification %name;>
```

Пример 1. Извещение *testNotification*, не содержащее сведений об изменении значений переменных:

```
<?xml version='1.0' encoding='windows-1251'?>
<notification name='testNotification'/>
```

Пример 2. Извещение *variableChanges* содержит новые значения для скаляров *boolFlag и count*, а также сведения об изменении трех строк таблицы *portActivity*:

- в строке с индексом *interface* равным *127.0.0.1*, *portNo* равным *80*, значение столбца *active* устанавливается в *1*;
- удаляется строка с индексом *interface* равным *192.168.0.101*, *portNo* равным *80*;
- добавляется строка с индексом *interface* равным *192.168.0.111*, *portNo* равным *99*.

Содержание извещения:

```
<?xml version='1.0' encoding='windows-1251'?>
<notification name='variableChanges'>
  <scalar name='boolFlag'><val type='Enumerated'>1</val>
  </scalar>
  <scalar name='count'><val type='Unsigned'>111</val>
  </scalar>
  <table name='portActivity'>
    <rowUpd>
      <index>
        <cell name='interface'>
          <val type='String'>127.0.0.1</val>
        </cell>
        <cell name='portNo'>
          <val type='Unsigned'>80</val>
        </cell>
      </index>
      <values>
        <cell name='active'>
          <val type='Enumerated'>1</val>
        </cell>
      </values>
    </rowUpd>
    <rowUpd>
      <index>
        <cell name='interface'>
          <val type='String'>192.168.0.101</val>
        </cell>
        <cell name='portNo'>
          <val type='Unsigned'>80</val>
        </cell>
      </index>
    </rowUpd>
    <rowUpd>
      <values>
        <cell name='interface'>
          <val type='String'>192.168.0.111</val>
        </cell>
```

```
          <cell name='portNo'>
            <val type='Unsigned'>99</val>
          </cell>
          <cell name='active'>
            <val type='Enumerated'>1</val>
          </cell>
        </values>
      </rowUpd>
    </table>
  </notification>
```

6. Требования к реализации протокола

При реализации протокола следует придерживаться следующих требований:

1) Имена скаляров, таблиц и их столбцов, а также извещений, поддерживаемых агентом, должны быть уникальны в пространстве имен агента. Для их записи должны использоваться буквы латинского алфавита, цифры, знак подчеркивания (первый символ – всегда буква). Все имена являются регистрозависимыми (т.е., например, *updFlag* и *updflag* являются разными именами);

2) Каждый агент, поддерживающий протокол XMLNMP, должен поддерживать скалярную переменную с именем **sysUpTime**. Тип данных переменной – *TimeTicks*, режим доступа – *readonly*. Данный скаляр определяет время с момента запуска агента управления (в сотых долях секунды);

3) Если агент управления осуществляет отправку извещений, то должна быть предусмотрена возможность указания для него конфигурационных параметров, определяющие адрес и имя порта получателя извещения, а также аутентификационной информации, необходимой для установления соединения с менеджером;

4) Если агенту управления не удается отправить какое-либо извещение одному из адресатов, то в дальнейшем агент должен приостановить попытки отправки последующих извещений в адрес этого адресата. С этого момента агент должен начать осуществлять периодические попытки отправки этому адресату специального извещения **notificationsLost**, не имеющего параметров. После успешной отправки такого извещения, агент осуществляет отправку последующих извещений в штатном порядке.

Факт получения менеджером извещения *notificationsLost* свидетельствует о необходимости актуализации менеджером данных о состоянии агента, находящихся в кэше менеджера.

Приложение. DTD для протокола XMLNMP

```
  <--!  DTD for XMLNMP
    XMLNMP - XML based Network Management Protocol version: 1.0
    Copyright (C) 2006, Mike Rozhnov. All Rights Reserved. -->
  <!ENTITY % name 'name CDATA #REQUIRED'>
  <!ENTITY % hint 'hint CDATA #IMPLIED'>
  <!ENTITY % access 'access (readonly|readwrite) "readonly"'>
  <--! Common tags -->
  <--! val - tag contain string representation of value and type
of value -->
  <!ELEMENT val (#PCDATA)>
  <!ATTLIST val type CDATA #REQUIRED>
  <--! This tags are used to describe changes of variables:
  - scalar tag contains name and new value for scalar variable,
  - table tag contains sequence of rowUpd tags.
  Each rowUpd tag contains manipulation with one row of table:
  - if index is absent, row will be added to the table,
  - if values is absent, row will be deleted from the table,
  - else row will be updated.
  Index must contain tags for all columns in table, which marked
as 'index'. -->
  <!ELEMENT scalar (val)>
  <!ATTLIST scalar %name;>
  <!ELEMENT table (rowUpd)+>
  <!ATTLIST table %name;>
  <!ELEMENT rowUpd (index|values|(index, values))>
  <!ELEMENT index (cell)+>
  <!ELEMENT values (cell)+>
  <!ELEMENT cell (val)>
  <!ATTLIST cell %name;>
  <--! This tags are used to describe value constrains.
  item tag describe items for Enumerated and Set types.
  range tag describe range for rest of types.
  For String and Bytes types range describe length of variable.
  For all other types range describe minimal amd maximal values of
variable.Ranges can't intersect. -->
  <!ELEMENT enumItem EMPTY>
  <!ATTLIST enumItem %name; value CDATA #REQUIRED %hint;>
  <!ELEMENT range (min, max)>
  <!ELEMENT min (val)>
  <!ELEMENT max (val)>
  <--! Protocol requests and responses -->
  <--! *************** getDefinitions *************** -->
  <--! getDefinitions request -->
  <!ELEMENT getDefinitions EMPTY>
  <--! getDefinitions response -->
  <!ELEMENT definitions (userTypeDefinitions?,
            variableDefinitions, notificationDefinitions?)>
  <!ELEMENT userTypeDefinitions (userTypeDef)+>
  <!ELEMENT userTypeDef (enumItem*|range*)>
  <!ATTLIST userTypeDef %name; parent CDATA #REQUIRED %hint;>
```

```
<!ELEMENT variableDefinitions (scalarDef|tableDef)+>
<!ELEMENT scalarDef (enumItem*|range*)>
<!ATTLIST scalarDef %name; type CDATA #REQUIRED
                    %access; %hint;>
<!ELEMENT tableDef (columnDef+, indexItem+)>
<!ATTLIST tableDef %name;
                   createRow (yes|no) "no"
                   deleteRow (yes|no) "no"
                   %hint;>
<!ELEMENT columnDef (enumItem*|range*)>
<!ATTLIST columnDef %name; type CDATA #REQUIRED
                    %access; %hint;>
<!ELEMENT indexItem EMPTY>
<!ATTLIST indexItem %name;>
<!ELEMENT notificationDefinitions (notifDef)+>
<!ELEMENT notifDef (varName)*>
<!ATTLIST notifDef %name;
                   %hint;>
<!ELEMENT varName EMPTY>
<!ATTLIST varName %name;>
<--! *************** getVariables *************** -->
<--! getVariables request -->
<!ELEMENT getVariables (scalarReq|tableReq)+>
<!ELEMENT scalarReq EMPTY>
<!ATTLIST scalarReq %name;>
<!ELEMENT tableReq (colName*, colFilter*)>
<!ATTLIST tableReq %name;>
<!ELEMENT colName EMPTY>
<!ATTLIST colName %name;>
<!ELEMENT colFilter (val+|range+)>
<!ATTLIST colFilter %name;>
<--! getVariables response -->
<!ELEMENT variables (var)+>
<!ELEMENT var (val|nil|error|rows)>
<!ELEMENT rows (row*)>
<!ELEMENT row (val|nil|error)+>
<!ELEMENT error EMPTY>
<!ATTLIST error code CDATA #REQUIRED>
<!ELEMENT nil EMPTY>
<--! *************** setVariables *************** -->
<--! setVariables request -->
<!ELEMENT setVariables (scalar|table)+>
<--! setVariables response -->
<!ELEMENT setResult (success|fault)>
<!ELEMENT success EMPTY>
<!ELEMENT fault EMPTY>
<!ATTLIST fault %name;
                code CDATA #REQUIRED
                rowUpdNumber CDATA #IMPLIED>
<--! *************** notification *************** -->
<!ELEMENT notification (scalar|table)*>
<!ELEMENT notification %name;>
```