Руководство по написанию сценариев веб-API

Программное обеспечение McAfee ePolicy Orchestrator 5.1.0

АВТОРСКОЕ ПРАВО
Copyright © 2013 McAfee, Inc. Запрещается копирование материалов без разрешения.

ПРАВА НА ТОВАРНЫЕ ЗНАКИ
McAfee, логотип McAfee, McAfee Active Protection, McAfee CleanBoot, McAfee DeepSAFE, ePolicy Orchestrator, McAfee ePO, McAfee EMM, Foundscore, Foundstone, Policy Lab, McAfee QuickClean, Safe Eyes, McAfee SECURE, SecureOS, McAfee Shredder, SiteAdvisor, McAfee Stinger, McAfee Total Protection, TrustedSource, VirusScan, WaveSecure являются товарными знаками или зарегистрированными товарными знаками корпорации McAfee, Inc. или ее филиалов в США и других странах. Другие названия и фирменная символика являются собственностью соответствующих владельцев.

Названия и описания продукта и его функций могут быть изменены без предварительного уведомления. Наиболее актуальные продукты и функции можно найти на веб-сайте mcafee.com.

ИНФОРМАЦИЯ О ЛИЦЕНЗИИ

Лицензионное соглашение
ПРИМЕЧАНИЕ ДЛЯ ВСЕХ ПОЛЬЗОВАТЕЛЕЙ: ВНИМАТЕЛЬНО ОЗНАКОМЬТЕСЬ С СООТВЕТСТВУЮЩИМ ЮРИДИЧЕСКИМ СОГЛАШЕНИЕМ ПО ПРИОБРЕТЕННОЙ ЛИЦЕНЗИИ, В КОТОРОМ ОПРЕДЕЛЕНЫ ОСНОВНЫЕ УСЛОВИЯ И ПОЛОЖЕНИЯ ПО ИСПОЛЬЗОВАНИЮ ЛИЦЕНЗИРОВАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. ЕСЛИ ВЫ НЕ ЗНАЕТЕ, КАКОЙ ТИП ЛИЦЕНЗИИ ВЫ ПРИОБРЕЛИ, ОЗНАКОМЬТЕСЬ С ДРУГИМИ СОПРОВОДИТЕЛЬНЫМИ ДОКУМЕНТАМИ ПО ПРЕДОСТАВЛЕНИЮ ЛИЦЕНЗИИ ИЛИ ПОДАЧЕ ЗАКАЗА НА ПОКУПКУ, ПРИЛАГАЕМЫМИ К КОМПЛЕКТУ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ПОЛУЧЕННЫМИ ОТДЕЛЬНО КАК ЧАСТЬ ПОКУПКИ (В КАЧЕСТВЕ БУКЛЕТА, ФАЙЛА НА КОМПАКТ-ДИСКЕ (CD) ПРОДУКТА ИЛИ ФАЙЛА НА ВЕБ-САЙТЕ, С КОТОРОГО БЫЛ ЗАГРУЖЕН КОМПЛЕКТ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ). ЕСЛИ ВЫ НЕ СОГЛАСНЫ С УСЛОВИЯМИ НАСТОЯЩЕГО СОГЛАШЕНИЯ, НЕ УСТАНАВЛИВАЙТЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ. В СЛУЧАЕ НЕОБХОДИМОСТИ ВЫ МОЖЕТЕ ВЕРНУТЬ ПРОДУКТ В КОРПОРАЦИЮ MCAFEE ИЛИ НА МЕСТО ПОКУПКИ С ПОЛНЫМ ВОЗМЕЩЕНИЕМ СТОИМОСТИ.

Содержание

Содержание

Предисловие

Содержание

О данном руководстве

Здесь описывается целевая аудитория данного руководства, условные обозначения и пиктограммы, используемые в данном руководстве, а также его организация.

Аудитория

Документация McAfee тщательно анализируется и пишется для целевой аудитории.

Информация, содержащаяся в данном руководстве, в основном рассчитана на указанных ниже лиц:

- **Администраторы**. Люди, которые внедряют и применяют программы обеспечения безопасности компании.
- **Сотрудники службы безопасности**. Люди, которые определяют конфиденциальные данные, требующие защиты, и определяют корпоративную политику, направленную на защиту интеллектуальной собственности компании.

Принятые условные обозначения

В настоящем руководстве используются следующие условные обозначения и значки.

Название книги, *термин* и *выделение*	Название книги, главы или темы; новый термин; выделение.
Жирный шрифт	Данный текст имеет яркое выделение.
`Пользовательский ввод, код, сообщение`	Команды и другой текст, который вводится пользователем; пример кода; сообщение, отображаемое на экране.
Текст интерфейса	Слова, относящиеся к интерфейсу продукта: параметрам, меню, кнопкам, диалоговым окнам и т. п.
Гипертекст синего цвета	Ссылка на тему или внешний веб-сайт.
	**Примечание.** Дополнительная информация, например альтернативный способ получения доступа к параметру.
	**Совет.** Предложения и рекомендации.

Важно/внимание. Ценный совет по защите компьютера, программного обеспечения, сети, бизнеса или данных.

Предупреждение. Важный совет по предотвращению телесных повреждений при работе с аппаратным обеспечением.

Поиск документации по продукту

McAfee предоставляет необходимую информацию на каждом этапе внедрения продукта: от установки до ежедневного использования и устранения неисправностей. После выпуска продукта информация о нем включается в интернет-базу McAfee KnowledgeBase.

Процедура

1 Перейдите на портал технической поддержки McAfee Technical Support ServicePortal по адресу http://mysupport.mcafee.com.

2 В разделе **Self Service** (Самообслуживание) найдите необходимую информацию:

Для доступа...	Действия...
Документация пользователя	1 Щелкните **Product Documentation** (Документация по продукту). 2 Выберите продукт, а затем выберите версию. 3 Выберите документ по продукту.
KnowledgeBase	• Щелкните **Search the KnowledgeBase** (Поиск в базе знаний), чтобы найти ответы на вопросы по продукту. • Щелкните **Browse the KnowledgeBase** (Просмотр базы знаний) для просмотра списка статей, отсортированного по продукту и версии.

1 Обзор

McAfee® ePolicy Orchestrator® предоставляет прикладной программный веб-интерфейс (API), позволяющий использовать сценарии и автоматизировать стандартные действия по управлению. Например, можно автоматизировать обслуживание пользователей и дерева систем, а также операции импорта и экспорта данных.

В данном руководстве объясняется, что такое веб-API ePolicy Orchestrator, как его пользовать, и рассматриваются несколько примеров с использованием клиента Python. Кроме того, здесь более подробно рассматриваются несколько ключевых команд, включая расширенное описание системы запросов.

Содержание

Основы веб-API

Вместо пользовательского интерфейса для автоматизации настройки ePolicy Orchestrator с помощью сценариев можно использовать веб-API ePolicy Orchestrator с командной строкой.

В этом разделе рассматриваются примеры использования командной стройки cURL для переноса данных с синтаксисом URL. Все команды выполняются без вмешательства пользователя. Исполняемый файл cURL представляет собой свободно распространяемое программное обеспечение, работающее на многих операционных системах.

Каждый пример cURL содержит стандартные параметры, за которыми следует фактический URL-адрес веб-API ePolicy Orchestrator, который выполняет команду на сервере McAfee ePO. Эти параметры помогут вам понять, что делает команда, хотя на практике вам следует внедрить более строгий режим безопасности в отношении доверия к сертификату сайта.

Веб-API ePolicy Orchestrator поддерживает и другие средства командной строки, например `wget` (часть GNU Project, © Free Software Foundation, Inc., 2009 г.), используемые для получения данных с сервера McAfee ePO.

Следующая команда, например, отображает синтаксис cURL и URL-адрес в целях иллюстрации основных возможностей веб-API.

```
> curl -k -u ga:ga https://localhost:8443/remote/core.help
```

В данной таблице представлены параметры, используемые в примере команды `curl`.

Параметр	Описание
`-k`	Позволяет cURL осуществлять «небезопасное» SSL-подключение и обмен данными.
`-u`	Указывает имя пользователя и пароль для проверки подлинности на сервере. Если ввести только имя пользователя (без двоеточия), cURL запросит пароль.
`ga`	Имя пользователя «ga» (глобальный администратор), используемое в примерах в данном документе. В имени пользователя можно использовать специальные символы, однако необходимо убедиться, что соблюдаются правила употребления кавычек и написания управляющих последовательностей в соответствии с используемой оболочкой.
`ga`	Пароль «ga», используемый в примерах в данном документе. В пароле можно использовать специальные символы, однако необходимо убедиться, что соблюдаются правила употребления кавычек и написания управляющих последовательностей в соответствии с используемой оболочкой.
`localhost` :	В примерах в данном документе сервер ePolicy Orchestrator обозначается как localhost.
`8443`	Порт назначения, в данном примере используется значение по умолчанию 8443.

В примерах в данном документе сервер McAfee ePO и порт назначения определены как localhost и 8443 (по умолчанию). Вам необходимо будет заменить эти элементы на имя сервера и номер порта вашей собственной системы.

Команды веб-API следуют всем разрешениям на основе ролей, принудительно применяемым посредством графического интерфейса сервера McAfee ePO.

Веб-API применяется главным образом для двух целей:

- выполнение простых заданий без использования интерфейса пользователя
- составление сценариев для последовательностей заданий

Сценарии с использованием веб-API можно выполнять с любого компьютера, который может подключиться к серверу McAfee ePO. По соображениям безопасности эти команды не следует запускать на том же компьютере, что и собственно сервер McAfee ePO.

Общий синтаксис

Общий синтаксис команды, передаваемой по протоколу HTTPS, следующий:

```
https://<сервер>:<порт>/remote/<команда>?<арг1>=<знач1>&<арг2>=<знач2>
```

При необходимости можно указать дополнительные аргументы.

Некоторым командам требуется ввод в других форматах, например импорт файла. Например, импорт XML-файла, содержащего наборы разрешений, выглядит подобным образом:

```
> curl -k -u ga:ga "https://localhost:8443/remote/core.importPermissionSets" -F
file=@permissionSets.xml
```

Параметры вывода

По умолчанию команды выводят данные в читаемом для человека формате. В сценариях, однако, обычно требуется возврат командами данных в машиночитаемом формате. Этим управляет параметр `:output`.

```
https://localhost:8443/remote/core.help?:output=json
```

Этот пример возвращает данные в формате нотации объектов JavaScript (JSON). К другим параметрам относятся `verbose` (по умолчанию), `terse` и `xml`. Все символы этих аргументов должны иметь нижний регистр. Кроме того, доступны параметры, представленные в следующей таблице.

Таблица 1-1 Параметры формата вывода

Параметр	Описание	Значения
:output	Указывает формат вывода.	`verbose` (по умолчанию), `terse`, `xml`, `json`
:locale	Указывает локаль вывода.	По умолчанию используется локаль сервера. Примеры значений: `en`, `de`, `cn-zh`.
:validation	Указывает уровень проверки команды. Строгая проверка (strict) выдает ошибки при отсутствии аргумента, вольная (loose) игнорирует отсутствующие аргументы.	`strict` (по умолчанию), `loose`

Обнаружение доступных команд посредством URL-адресов

Вновь установленные расширения ePolicy Orchestrator предоставляют дополнительные команды в веб-API. Узнайте, какие команды вам доступны.

Чтобы узнать доступные команды и получить подробные сведения о конкретных командах, используйте команду `core.help`. Будучи вызванной без аргументов, команда `core.help` создает список доступных команд.

```
> curl -k -u ga:ga https://localhost:8443/remote/core.help
```

Содержание этого списка команд зависит от прав пользователя и установленных расширений.

Эта команда возвращает список примерно следующего содержания.

```
OK:
ComputerMgmt.createCustomInstallPackageCmd windowsPackage deployPath [ahId] [fallBackAhId]
[useCred] [domain] [username] [password] [rememberDomainCredentials] -
ComputerMgmt.create.Custom.Install.Package.Cmd.short-desc
agentmgmt.listAgentHandlers - отображает список всех обработчиков агентов
clienttask.export [productId] [fileName] - экспортирует задачи клиента
clienttask.find [searchText] - выполняет поиск задач клиента
clienttask.importClientTask importFileName - импортирует задачи клиента из XML-файла.
clienttask.run names productId taskId [retryAttempts] [retryIntervalInSeconds]
[abortAfterMinutes] [useAllAgentHandlers] [stopAfterMinutes] [randomizationInterval] -
выполняет задачу клиента в системах из предоставленного списка систем
```

```
clienttask.syncShared - предоставляет совместный доступ к задачам клиента на
зарегистрированных серверах-участниках
commonevent.purgeEvents queryId [unit] [purgeType] - удаляет события угроз на основании
возраста или идентификатора запроса. Запрос должен основываться на таблице.
commonevent.purgeProductEvents queryId [unit] [purgeType] - очистить события клиента на
основании идентификатора запроса или возраста.
console.cert.updatecrl console.updateCRL crlFile - cert.update.crl.help.oneline
core.addPermSetsForUser userName permSetName - добавляет набор(ы) разрешений конкретному
пользователю
core.addUser userName password [fullName=<>] [email=<>] [phoneNumber=<>] [notes=<>]
[disabled=<>] [admin=<>] - добавляет пользователя в систему.
core.executeQuery queryId [database=<>] - выполняет запрос SQUID и возвращает результаты
.
[information deleted]
.
system.report names - выполняет поиск систем в дереве систем
system.runTagCriteria tagID [resetTaggedSystems] - действие «Выполнить критерии добавления
тегов» оценивает по критериям добавления тегов все управляемые системы.
system.setUserProperties names [description] [customField1] [customField2] [customField3]
[customField4] - задает свойства пользователя в данной системе
system.transfer names epoServer - передает системы на другой сервер ePO
system.wakeupAgent names [fullProps] [superAgent] [randomMinutes] [forceFullPolicyUpdate]
[useAllHandlers] [retryIntervalSeconds] [attempts] [abortAfterMinutes] [includeSubgroups] -
вызывает агента в системах из предоставленного списка систем
tasklog.listMessages taskLogId - перечисляет сообщения для указанной записи журнала задач
tasklog.listSubtasks taskLogId - перечисляет подзадачи указанной записи журнала задач
tasklog.listTaskHistory [taskName] [taskSource] [maxRows] [age] [unit] - перечисляет записи
журнала задач, (необязательно) отфильтрованные по имени задачи, ее идентификатору или
источнику задачи
tasklog.listTaskSources - перечисляет источники задач
tasklog.purge [age] [unit] - очищает журнал задач сервера от записей старше указанного
возраста в единицах времени
```

В выводе справки отображается следующее:

- префикс, например core;
- имя команды, например help;
- обязательные и необязательные аргументы. Необязательные аргументы заключаются в квадратные скобки («[» и «]»).

  Аргументы со следующими за ними символами =<> требуют указания конкретного имени аргумента и значения. Например, если команда help выводит [email=<>], необходимо указать имя аргумента и значение: `email=joe_test@mcafee.com`.

- краткое описание действий, выполняемых командой.

Этот расширенный пример команды используется для запрашивания более подробной информации о конкретной команде.

```
> curl -k -u ga:ga  https://localhost:8443/remote/core.help?command=core.listQueries
```

Эта команда отобразит следующее:

```
OK:
core.listQueries
отображает все запросы, которые разрешено просматривать пользователю. Возвращает список
запросов или выдает сообщение об ошибке.
Требуется разрешение на использование запросов.
```

Пример задачи с использованием веб-API

С помощью веб-API ePolicy Orchestrator можно выполнять множество задач. Ниже приведен пример последовательности действий при выполнении задачи.

В данном примере веб-API будет использован для назначения политики группе.

Процедура

1 С помощью запроса Help узнайте аргументы, требуемые команде `policy.assignToGroup`.

```
> curl -k -u ga:ga https://localhost:8443/remote/core.help?command=policy.assignToGroup
```

Эта команда отобразит следующее:

```
OK:
policy.assignToGroup groupId productId objectId [resetInheritance]
Назначает политику для указанной группы или сбрасывает наследование в группе для
указанной политики.
Требует разрешения не менее чем для одной группы в дереве систем и изменения разрешения
не менее чем для одного продукта.
Параметры:
groupId (param 1) - идентификатор группы, возвращаемый командой system.findGroups
productId (param 2) - идентификатор продукта, возвращаемый командой policy.find
objectId (param 3) - идентификатор объекта, возвращаемый командой policy.find
resetInheritance (param 4) - если имеет значение true, сбрасывает наследование указанной
политики в заданной группе. Значение по умолчанию: false
```

Запрос Help показывает, что у команды есть три обязательных аргумента:

- идентификатор группы;
- идентификатор продукта, для которого выполняется назначение политики;
- идентификатор объекта назначаемой политики.

Также можно сбросить наследование, но этот аргумент необязателен, и мы не будем использовать его в данном примере.

2 С помощью команды `system.findGroups` определите идентификатор группы.

```
> curl -k -u ga:ga https://localhost:8443/remote/system.findGroups?searchText=Моя группа
```

Эта команда возвращает результат, подобный следующему:

```
OK:
groupId: 2
groupPath: Моя организация

groupId: 4
groupPath: Моя организация\Моя Группа
```

3 С помощью команды `policy.find` найдите идентификатор продукта и идентификатор объекта политики:

```
> curl -k -u ga:ga https://localhost:8443/remote/policy.find?searchText=карантин
```

Эта команда возвращает два результата:

```
OK:
featureId: VIRUSCAN8800
featureName: VirusScan Enterprise
objectId: 131
objectName: McAfee Default
objectNotes:
productId: VIRUSCAN8800
productName: VirusScan Enterprise 8.8.0
typeId: 67
typeName: Политики диспетчера карантина

featureId: VIRUSCAN8800
featureName: VirusScan Enterprise
objectId: 142
objectName: My Default
objectNotes:
productId: VIRUSCAN8800
productName: VirusScan Enterprise 8.8.0
typeId: 67
typeName: Политики диспетчера карантина
```

4 Выберите один из результатов команды `policy.find`. В данном примере используется политика My Default, и у вас есть все необходимые данные для назначения этой политики группе:

- Идентификатор группы — 4
- Идентификатор продукта — VIRUSCAN8800
- Идентификатор объекта политики — 142

5 Используйте эти данные и команду `policy.assignToGroup` для присвоения политики группе:

```
> curl -k -u ga:ga https://localhost:8443/remote/policy.assignToGroup?
groupId=4&productId=VIRUSCAN8800&objectId=142
```

Эта команда возвращает

```
OK:
Истина
```

Политика присвоена.

В целом, веб-API может возвращать три вида результатов своих команд:

1 результаты `True` и `False` указывают на успешное или неуспешное выполнение команды;

2 данные, возвращаемые командой;

3 различные элементы данных с указанием успешности их обработки.

Основы использования клиента Python

McAfee предоставляет возможность загрузки программного обеспечения, включающего Python версии 2.7 и клиентское программное обеспечение Python Remote Client, также известное как

pyclient. Предоставляемое программное обеспечение Python Remote Client упрощает взаимодействие с веб-API ePolicy Orchestrator и его освоение.

Использование команд веб-API ePolicy Orchestrator с языком написания сценариев, таким как Python, обеспечивают большую гибкость работы. Язык сценариев позволяет использовать вывод одной команды в качестве входных данных для другой или выполнять условные действия в зависимости от входных данных сценария или вывода команды.

Если вы подключаетесь к ePolicy Orchestrator 5.x с помощью клиента Python ePolicy Orchestrator 4.6.x, могут возникнуть проблемы соединения. Пользователям ePolicy Orchestrator 5.x необходимо загрузить клиент Python ePolicy Orchestrator 5.x, доступный в диспетчере программ.

Требования к программному обеспечению клиента Python

Для клиента Python требуется программное обеспечение Python версии 2.6 или 2.7. Исходный код включен в файл с именем `mcafee.py`. Чтобы разработанные сценарии клиента Python работали надлежащим образом, файл `mcafee.py` должен размещаться либо в той же папке, что и сценарии, либо в папке, на которую указывает путь поиска модуля Python.

Применение клиента

Для облегчения обработки клиент Python преобразует данные, возвращаемые всеми командами, в объекты Python. Клиент Python может применяться двумя способами: импортироваться в сценарий `.py`, запускаемый из командной строки, либо с помощью интерактивного режима Python. В любом случае соблюдайте требования к использованию сценариев. Подробные сведения см. в разделе *Требования к сценариям Python для веб-API*.

Предоставляемый исходный код клиента может применяться в образовательных целях: его можно переписывать на других языках, а также разъяснять на его примере возможности клиента Python.

Примечания о параметрах

Некоторые команды, например `system.clearTag`, могут принимать в качестве параметра список значений, разделенных запятыми. Если необходимо внедрить в такой список пробелы, весь список нужно заключить в кавычки.

```
mc.system.clearTag("система1, система2, система3","старый_тег")
```

В любом параметре, которому требуется имя файла, используется формат `file:///c:/путь_к_файлу`, применяемый в URL-адресах.

См. также

Требования сценария Python для веб-API на стр. 14

Требования сценария Python для веб-API

Прежде чем сценарий Python сможет выполнить какую-либо команду Python Remote Client, он должен импортировать библиотеку клиента McAfee Python и пройти проверку подлинности на сервере McAfee ePO.

Требования к сценарию клиента Python

Любой создаваемый вами сценарий Python должен содержать в начале следующие две строки кода:

```
# McAfee Python script requirements import mcafee mc =
mcafee.client("localhost","8443","username","password")
```

В данном примере:

- `import mcafee` — импортирует в сценарий библиотеку Python клиента McAfee. См. раздел *Импорт библиотеки клиента McAfee Python*.
- `mc = mcafee.client("localhost","8443","username","password")` — выполняет проверку подлинности на сервере. См. раздел *Проверка подлинности сценария на сервере McAfee ePO*.

По завершении этих двух задач оставшаяся часть сценария будет функционировать надлежащим образом.

См. также

Импорт библиотеки клиента McAfee Python на стр. 14
Проверка подлинности сценария на сервере McAfee ePO на стр. 14

Импорт библиотеки клиента McAfee Python

Импортируйте библиотеку клиента McAfee Python в любой сценарий, созданный для обмена данными с сервером McAfee ePO.

Процедура

1. Убедитесь в том, что файл `mcafee.py` хранится либо в той же папке, что и сценарий, либо в папке, на которую указывает путь поиска модуля Python.
2. Импортируйте клиентский код в свой сценарий с помощью следующей команды:

   ```
   import mcafee
   ```

С помощью данной команды выполняется импорт клиентского кода сервера McAfee ePO, включающего метод, который принимает информацию о соединении, создает сеанс с указанным сервером и возвращает объект сеанса.

Проверка подлинности сценария на сервере McAfee ePO

Прежде чем отправлять на сервер McAfee ePO любые команды, выполните проверку подлинности на сервере и сохраните возвращенный сервером объект сеанса.

Чтобы обеспечить взаимодействие сценария с сервером, воспользуйтесь командой `mcafee.client`, принимающей от четырех до шести строковых параметров.

Таблица 1-2 Параметры mcafee.client

Параметр	Описание
`server`	Содержит имя сервера. Не включайте префикс `https://`. Если имя сервера — `https://мой_сервер`, поместите в качестве параметра `мой_сервер`.
`port`	Содержит порт, используемый сервером. Для HTTPS-соединений, как правило, используется значение `8443`, но может быть и иное.
`username`	Содержит имя пользователя, используемое при проверке подлинности учетных данных.
`password`	Содержит пароль, используемый при проверке подлинности учетных данных.  Пароль, используемый в сценарии, хранится в виде неформатированного текста. Вам следует предпринять необходимые меры по защите сценария. В качестве альтернативы можно либо запрашивать пароль у пользователя, либо хранить пароль в зашифрованном виде. Веб-API не поддерживает проверку подлинности на основании сертификатов.
`protocol`	[необязательно] Содержит протокол. Значение этого параметра по умолчанию — `HTTPS`.
`outputtype`	[необязательно] Содержит тип вывода, возвращенный командами. По умолчанию используется значение `json`, но допустимы также значения `verbose`, `terse` и `xml`.

Примеры проверки подлинности

Если для входа на `сервер ePO` используется порт 8443, имя пользователя `adminfred` и пароль `имяМоейс0б@ки37`, команда Python для входа будет иметь следующий вид:

```
mc = mcafee.client("север_ePO","8443","adminfred","имяМоейс0б@ки37")
```

В переменной `mc` хранится информация о сеансе, используемая для всех команд, выполняемых в дальнейшем в данном сценарии. Например, если следующим действием, предпринимаемым в рамках сценария, является вывод списка задач сервера, выполняемых в данный момент, команда будет выглядеть следующим образом:

```
mc.scheduler.listRunningServerTasks()
```

Обнаружение доступных команд в Python

Расширения McAfee могут использоваться для добавления команд в библиотеку клиента McAfee Python. Клиент Python предоставляет способ определения, какие именно команды доступны на сервере.

Предварительные операции

Для работы этих команд сценарий должен отвечать условиям, описанным в разделе *Требования сценария Python для веб-API*.

Процедура

1 Используйте команду Python `dir()`, чтобы найти список доступных моделей.

```
>>> dir(mc)
```

Эта команда возвращает список Python примерно следующего содержания:

```
['__class__', '__delattr__', '__dict__', '__doc__', '__format__', '__getattr__',
'__getattribute__',
'__hash__', '__init__', '__module__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_invoker', 'core', 'help',
'scheduler', 'tasklog']
```

В нем содержится ряд атрибутов, но вам следует обратить внимание на список имен, не начинающихся с подчеркивания. Они приведены в конце. В данном примере — это `core`, `help`, `scheduler` и `tasklog`. Эти объекты содержат команды, которые можно выполнить.

Другие имена атрибутов, которые начинаются с подчеркивания, — это внутрисистемные имена клиента или собственно Python.

2 Чтобы узнать список команд в модуле, используйте команду `dir()`, указав модуль в качестве параметра.

```
>>> dir(mc.scheduler)
```

Эта команда возвращает список атрибутов и команд модуля планировщика примерно следующего содержания:

```
['__doc__', '__getattr__', '__init__', '__module__', '_invoker', '_module',
'cancelServerTask', 'getServerTask', 'listAllServerTasks', 'listRunningServerTasks',
'runServerTask', 'setServerTaskStatus']
```

Как и на предыдущем шаге, следует обращать внимание на имена, не начинающиеся с подчеркивания. В данном случае этих команд шесть: `cancelServerTask`, `getServerTask`, `listAllServerTasks`, `listRunningServerTasks`, `runServerTask` и `setServerTaskStatus`.

3 Когда требуемая команда найдена, воспользуйтесь функцией `help()` и передайте в качестве параметра эту команду.

```
>>> mc.help('scheduler.listRunningServerTasks')
```

Данная команда возвращает описание в следующем виде:

```
scheduler.listRunningServerTasks
Получение списка всех выполняемых задач сервера. Возвращает список задач или выдает
сообщение об ошибке. Требуется разрешение на просмотр задач сервера.
```

В описании перечислены имена команд и параметров (в данном случае они отсутствуют), описание выполняемых действий и разрешения, необходимые для их выполнения.

Выполнение таких же действий для найденных вами команд поможет выявить все возможности, которые могут использоваться для сценариев.

Дополнительная документация, прилагаемая к веб-API

В пакет веб-API включены несколько HTML-файлов, содержащих дополнительные сведения.

Все файлы включены в папку `pyclient/Python26/mcafee-docs`, содержащуюся в пакете.

Таблица 1-3 Файлы документации в пакете веб-API

Файл	Тема
`FAQ.html`	Содержит ответы на распространенные вопросы по использованию клиента Python.
`getting_started.html`	Содержит краткое учебное пособие по написанию сценариев на Python.
`implementation.html`	Содержит подробности реализации клиента Python, включая форматы ответов.
`setup.html`	Содержит инструкции по установке дистрибутива Python или по настройке существующей установки.

Основные команды

Некоторые команды используются более часто, чем другие. Чтобы быстро создавать сценарии, рекомендуем ознакомиться с синтаксисом этих распространенных команд.

В приведенных ниже таблицах перечислены часто используемые команды с их синтаксисом и описанием. Каждая таблица отражает определенную функциональную область.

Укажите аргументы по именам, а затем добавьте `=<>`. Например, в команду должен быть включен аргумент `fullName=`: `core.addUser("ga", "ga", fullName="Joe Tester")`

Таблица 1-4 Команды для поиска, составления запросов и списков

Команда	Синтаксис	Описание
`core.executeQuery`	`core.executeQuery queryId [database=<>]` `core.executeQuery target=<> [select=<>] [where=<>] [order=<>] [group=<>] [database=<>] [depth=<>] [joinTables=<>]`	Выполняет запрос и возвращает результаты в виде списка объектов.
`core.help`	`core.help [command] [prefix=<>]`	Перечисляются все зарегистрированные команды и отображаются строки справки.
`core.listDatabases`	`core.listDatabases`	Возвращает все базы данных, разрешенные для просмотра пользователю, в виде списка объектов.
`core.listQueries`	`core.listQueries`	Возвращает все запросы, разрешенные для просмотра пользователю, в виде списка объектов.
`core.listTables`	`core.listTables [table=<>]`	Возвращает все таблицы баз данных, разрешенные для просмотра пользователю, в виде списка объектов.
`policy.find`	`policy.find searchText`	Выполняет поиск всех политик, соответствующих искомому тексту, на которые у пользователя есть разрешения на просмотр.
`repository.find`	`repository.find searchText`	Выполняет поиск всех репозиториев, соответствующих искомому тексту, на которые у пользователя есть разрешения на просмотр.
`system.find`	`system.find searchText`	Выполняет поиск систем в дереве систем.

Таблица 1-5 Команды для создания, импортирования и экспортирования

Команда	Синтаксис	Описание
core.addUser	core.addUser userName password [fullName=<>] [email=<>] [phoneNumber=<>] [notes=<>] [disabled=<>] [admin=<>] core.addUser userName=<> windowsUserName=<> windowsDomain=<> [fullName=<>] [email=<>] [phoneNumber=<>] [notes=<>] [disabled=<>] [admin=<>] core.addUser userName=<> subjectDN=<> [fullName=<>] [email=<>] [phoneNumber=<>] [notes=<>] [disabled=<>] [admin=<>]	Добавляет пользователя в систему. Параметры проверки подлинности являются взаимоисключающими: можно задать либо password, либо windowsUserName/ windowsDomain, либо subjectDN.
core.importPermissionSets	core.importPermissionSets file [overwrite]	Импортирует наборы разрешений из файла.
core.exportPermissionSets	core.exportPermissionSets	Экспортирует все наборы разрешений в XML-строки.
system.importSystem	system.importSystem fileName branchNodeID [allowDuplicates] [uninstallRemoved] [pushAgent] [pushAgentForceInstall] [pushAgentSkipIfInstalled] [pushAgentSuppressUI] [pushAgentInstallPath] [pushAgentPackagePath] [pushAgentDomainName] [pushAgentUserName] [pushAgentPassword] [deleteIfRemoved] [createNewInLostAndFound] [flattenTreeStructure]	Импортирует системы из текстового файла или из предоставленного списка с разделителями в виде запятых.
repository.checkInPackage	repository.checkInPackage packageLocation branch [option] [moveToPrevious] [allowUnsignedPackages]	Регистрирует пакет в главном репозитории.

Таблица 1-6 Команды для изменения, назначения и перемещения

Команда	Синтаксис	Описание
core.updateUser	core.updateUser userName [password=<>] [windowsUserName=<>] [windowsDomain=<>] [subjectDN=<>] [newUserName=<>] [fullName=<>] [email=<>] [phoneNumber=<>] [notes=<>] [disabled=<>] [admin=<>]	Обновляет существующего пользователя. Параметры проверки подлинности являются взаимно исключающими: можно задать либо пароль, имя пользователя Windows/ доменное имя Windows, либо различающееся имя темы.
core.addPermSetsForUser	core.addPermSetsForUser userName permSetName	Добавляет данный набор разрешений определенному пользователю.
system.applyTag	system.applyTag names tagName	Назначает данный тег предоставленному списку систем.

Таблица 1-6 Команды для изменения, назначения и перемещения *(продолжение)*

Команда	Синтаксис	Описание
system.setUserProperties	system.setUserProperties name [description] [customField1] [customField2] [customField3] [customField4]	Задает свойства пользователя в данной системе.
system.deployAgent	system.deployAgent names username [password] [agentPackage] [skipIfInstalled] [suppressUI] [forceInstall] [installPath] [domain] [useAllHandlers] [primaryAgentHandler] [retryIntervalSeconds] [attempts] [abortAfterMinutes] [includeSubgroups] [useSsh] [inputSource]	Развертывает агент в системах из данного списка.
policy.assignToSystem	policy.assignToSystem names productId typeId objectId [resetInheritance]	Назначает политику предоставленному списку систем.

Таблица 1-7 Команды для выполнения и аварийного прерывания

Команда	Синтаксис	Описание
clienttask.run	clienttask.run names productId taskID [retryAttempts] [retryIntervalInSeconds] [abortAfterMinutes] [useAllAgentHandlers] [stopAfterMinutes] [randomizationInterval]	Выполняет задачу клиента в системах из предоставленного списка систем.
scheduler.cancelServerTask	scheduler.cancelServerTask taskLogId	Прекращает выполнение текущей задачи.
scheduler.runServerTask	scheduler.runServerTask taskName	Выполняет указанную задачу сервера.
system.wakeupAgent	system.wakeupAgent names [fullProps] [superAgent] [randomMinutes] [forceFullPolicyUpdate] [useAllHandlers] [retryIntervalSeconds] [attempts] [abortAfterMinutes] [includeSubgroups]	Вызывает агента в системах из предоставленного списка систем.
repository.pull	repository.pull sourceRepository targetBranch [moveToPrevious] [productList]	Извлекает пакеты из исходного репозитория.

Таблица 1-8 Команды для удаления и очистки

Команда	Синтаксис	Описание
`commonevent.purgeEvents`	`commonevent.purgeEvents queryId [unit]`	Удаляет события угроз на основании возраста или идентификатора запроса. Этот запрос должен основываться на таблице.
`core.purgeAuditLog`	`core.purgeAuditLog [age] [unit]`	Очищает журнал аудита по возрасту.
`system.delete`	`system.delete names [uninstall]`	Удаляет системы с сервера McAfee ePO по имени или идентификатору.
`tasklog.purge`	`tasklog.purge [age] [unit]`	Очищает журнал задач по возрасту. По умолчанию выполняется очистка всех записей.

2 Примеры сценариев Python

Создание нескольких пробных сценариев Python даст вам представление о множестве способов обслуживания и обновления серверов McAfee ePO.

Просмотрите все эти сценарии, отражающие различные категории задач, в представленном порядке.

1. Назначение тегов системам из списка.
2. Получение тега на входе и отправка вызова агента McAfee Agent всем системам с данным тегом.
3. Применение автоматизации для очистки отключенных пользователей ePolicy Orchestrator.
4. Импорт компьютеров из внешнего файла и добавление их в дерево систем.
5. Экспорт политик и задач клиента с одного сервера McAfee ePO и импорт их на другой.
6. Организация дерева систем.

В последующих сценариях используется ранее рассмотренный код предыдущих, и логика их построения постепенно поясняется.

Однако в более поздних сценариях не повторяются идеи, раскрытые в более ранних сценариях.

Содержание

Пример 1. Назначение тегов системам из списка

Этот пример обучает основам написания сценариев Python с помощью веб-API для ePolicy Orchestrator. После демонстрации всего сценария мы подробно изучим отдельные его части.

Этот сценарий основан на следующих предположениях.

- Имеется текстовый файл `мой_файл.txt` со списком управляемых систем (каждая система в отдельной строке), при этом системы перечислены по именам или IP-адресам.
- Все системы в списке необходимо снабдить тегом (в данном примере — `мой_тег`).
- Тег `мой_тег` уже создан.

Эти предположения можно исключить, используя более надежный сценарий, но это усложнит пример.

```
#Example 1
import mcafee
mc = mcafee.client('localhost','8443','ga','ga')

file = open('C:/путь_к_моему_файлу.txt', 'r')
for line in file:
mc.system.applyTag(line.rstrip('\n'), 'мой_тег')
```

Рассмотрим разделы сценария подробнее

Эта строка импортирует предоставленный модуль McAfee Python (`mcafee.py`), хранящийся в том же каталоге, что и сценарий.

```
import mcafee
```

В следующей строке создается подключение к серверу McAfee ePO путем указания имени сервера, порта подключения, имени пользователя и пароля, в приведенном порядке.

Эта функция инициализации может принимать до двух дополнительных параметров, указывающих протокол и представление вывода.

```
mc = mcafee.client('localhost','8443','ga','ga')
```

Полный список параметров:

```
mc = mcafee.client('yourservername','port','username','password', 'protocol',
'outputtype')
```

- По умолчанию параметр `protocol` принимает значение `https` на сервере McAfee ePO.
- Параметр `outputtype` определяет формат вывода команд, как описано в разделе *Основы веб-API*.

С помощью этих двух строк мы установили подключение к серверу.

Эта строка создает дескриптор файла в режиме «только для чтения».

```
file = open('C:/путь_к_моему_файлу.txt', 'r')
```

Эти строки повторяются далее в файле для выполнения команды `system.applyTag` с каждой из систем, указанных в файле, при этом каждый раз удаляется символ перевода строки (`'\n'`):

```
 for line in file:
mc.system.applyTag(line.rstrip('\n'), 'мой_тег'')
```

После завершения цикла каждая система из файла теперь должна иметь тег `мой_тег`.

Пример 2. Автоматизация повторяющихся задач в управляемых системах

Этот сценарий позволяет получить имя тега на входе и отправить вызов агента McAfee Agent всем системам, снабженным этим тегом.

```
#Example 2
import mcafee
mc = mcafee.client('localhost','8443','ga','ga')

input = 'myTag'
systems = mc.system.find(input)
for system in systems:
id = system['EPOComputerProperties.ParentID']
result = mc.system.wakeupAgent(id)
```

Данный сценарий принимает на вход один аргумент (в данном примере — `мой_тег`). Затем с помощью команды `system.find` он выполняет поиск всех компьютеров с соответствующим тегом.

В качестве входного параметра можно использовать что-либо отличное от тега, например описание команды `system.find` — при этом отображается сообщение «Поиск систем в дереве ePolicy Orchestrator по имени, IP-адресу, MAC-адресу, имени пользователя, AgentGUID или тегу».

Вышеприведенный сценарий использует свойство `EPOComputerProperties.ParentID` для отправки команде `system.wakeupAgent`, но так как эта команда также принимает имя, строку можно написать следующим образом:

```
id = system['EPOComputerProperties.ComputerName']
```

Можно создать строку с разделением запятыми и отправить список непосредственно команде, поскольку команда `system.wakeupAgent` также принимает в качестве ввода список имен или идентификаторов.

Пример 3. Автоматизация управления пользователями

Данный сценарий просматривает всех пользователей ePolicy Orchestrator и удаляет тех из них, которые помечены как «отключенные», если они не являются администраторами.

Кроме того, данный сценарий обеспечивает более подробную обработку действий, а также создает исключение в случае невозможности удаления пользователя.

```
#Example 3
import mcafee
mc = mcafee.client('localhost','8443','ga','ga')

users = mc.core.listUsers();
for user in users:
if user['disabled'] == True and user['admin'] == False:
name = user['UserName']
try:
mc.core.removeUser(name)
except Exception, e:
print 'Error ' + str(e)
```

Чтобы узнать доступные свойства, изучите вывод команды `core.listUsers`. Для поиска и удаления конкретных пользователей этот сценарий использует свойства `disabled`, `admin` и `UserName`.

Для выполнения команды `core.removeUser` необходимы права администратора. Необходимые права для каждой команды перечислены в подробной справке по ней.

Команда `core.listUsers` возвращает различные значения `authType`. В таблице перечислены используемые в пользовательском интерфейсе человекочитаемые форматы и машиночитаемые форматы.

Человекочитаемый	Машиночитаемый
Проверка подлинности MFS	pwd
Проверка подлинности на основании использования сертификатов	cert
Проверка подлинности Windows	ntlm

Пример 4. Импорт компьютеров из внешних источников

Этот сценарий можно использовать для получения файла с разделением запятыми, содержащего сведения о системах, и импорта этих систем в указанную группу дерева систем на сервере McAfee ePO.

Данный сценарий предполагает наличие двух исходных файлов.

- `мой_файл.txt` — содержит группу дерева систем, в которую добавляются системы.
- `добавляемые_системы.txt` — содержит по одной системе на строку со списком свойств, разделенным запятыми, в следующем порядке: MAC-адрес, IP-адрес, имя системы и имя домена.

```
#Example 4
import mcafee, sys
mc = mcafee.client('localhost','8443','ga','ga','https','json')

file = open('C:/путь_к_моему_файлу.txt', 'r')
for line in file:
#determine the ID of the group to add it to
groups = mc.system.findGroups(line.rstrip('\n'))
groupId = -1
for group in groups:
if group['groupPath'] == 'My Organization\\' + line.rstrip('\n'):
groupId = group['groupId']

    if groupId == -1:
    error = 'Error finding the specified group.'
    sys.exit(error)

#now that we have the group id, pull in the systems from file
sourceId = "12"
sourceType = "CLI"
file = open('C:/добавляемые_системы.txt', 'r')
for line in file:
sysProps = line.rstrip('\n').split(',')
# Contains line break at "\"
systemId = mc.detectedsystem.add(sourceId,sourceType,sysProps[0],sysProps[1],dnsName= \
sysProps[2],do main=sysProps[3])
mc.detectedsystem.addToTree(str(systemId),str(groupId))
```

В этом примере с помощью команды `system.findGroups()` определяется место добавления систем в группу по имени, а затем с помощью этого имени извлекается идентификатор группы.

Системы добавляются в группу с помощью команды `detectedsystem.add`. Эта команда имеет эти дополнительные параметры:

```
detectedsystem.add sourceID sourceType MAC IPAddress [IPSubnet][IPSubnetMask][dnsName]
[OSPlatform] [OSFamily] [OSVersion]
[domain] [netbiosName][netbiosComment] [users] [agentGUID] [detectedTime] [externalID]
```

Можно добавить дополнительные параметры по порядку (используя произвольные значения для параметров, которые не указываются). Такими дополнительными параметрами могут быть, например, следующие:

```
mc.detectedsystem.add(sourceId, sourceType,sysProps[0], sysProps[1], "0.0.0.0", "0.0.0.0",
sysProps[2], '', '', '', sysProps[3])
```

Также параметры можно назначать по имени, как сделано для параметров `dnsName` и `domainName` в вышеприведенном сценарии.

Параметры `sourceID` и `sourceType` являются произвольными значениями, определяемыми при добавлении систем. Они хранятся в базе данных, так что имеется возможность регистрировать, какой источник обнаружил или добавил ту или иную заданную систему.

Возвращаемое этой командой значение представляет собой идентификатор вновь добавленной обнаруженной системы. Этот идентификатор используется в качестве ввода для эты команды, detectedsystem.addToTree, которая добавляет обнаруженные системы в дерево систем.

```
detectedsystem.addToTree UIDs branchNodeID [allowDuplicates] [dirSort]
```

По умолчанию система не будет добавлена, если она является дубликатом, равно как и не будет автоматически отсортирована; но при желании можно переопределить такое поведение. Данный сценарий принимает значения по умолчанию и, используя только что полученный идентификатор обнаруженной системы и идентификатор группы, перемещает эту систему в дерево систем.

Пример 5. Импорт и экспорт данных

Можно экспортировать данные с сервера McAfee ePO и импортировать их на другой сервер McAfee ePO с помощью сценария.

Импорт и экспорт общих настроек разрешений, политик или других объектов ePolicy Orchestrator используется при выполнении миграции серверов, настройке дополнительного сервера McAfee ePO или просто при создании тестовой среды. Данный сценарий экспортирует политики и задачи клиента для заданного продукта, а затем импортирует их на другой сервер.

```
#Example 5
import mcafee
mc = mcafee.client('localhost','8443','ga','ga','https','json')

# Find the product id
productId = None
policies = mc.policy.find('McAfee Agent')
for policy in policies:
productId = policy['productId']

if productId == None:
error = 'Error finding the product id.'
sys.exit(error)

tasks = mc.clienttask.export(productId=productId)
file = open('tasks.xml', 'w')
print >>file, tasks
file.close()
```

```
policies = mc.policy.export(productId=productId)
file = open('policies.xml', 'w')
print >>file, policies
file.close()

# Import these into another server:
mc2 = mcafee.client('anotherEpoServer','8443','ga','ga','https','json')
mc2.clienttask.importClientTask(uploadFile='file:///tasks.xml')
mc2.policy.importPolicy(file='file:///policies.xml')
```

Этот сценарий извлекает идентификатор продукта путем поиска политики, содержащей строку `'McAfee Agent'`. Используя этот идентификатор продукта, можно экспортировать все задачи клиента и политики для этого продукта.

Чтобы импортировать задачи и политики, создайте подключение ко второму серверу ePolicy Orchestrator (`mc2`) и выполните соответствующие команды импорта.

Пример 6. Автоматизированное обслуживание дерева систем

Данный сценарий повторно применяет правила сортировки дерева систем к любым системам, найденным в группе **Потерянные и найденные**.

```
#Example 6
import mcafee
mc = mcafee.client('localhost','8443','ga','ga','https','json')

#first, as before, get the id of the Lost&Found group
groups = mc.system.findGroups('Lost&Found')
groupId = -1
for group in groups:
if group['groupPath'] == 'My Organization\\Lost&Found':
groupId = group['groupId']

  if groupId == -1:
  error = 'Error finding the specified group.'
  sys.exit(error)

  #find all systems for this group
  systems = mc.epogroup.findSystems(str(groupId),'true')
  for system in systems:
  id = system['EPOComputerProperties.ParentID']
  mc.system.resort(str(id))
```

Этот пример должен быть в целом понятен с учетом предыдущих сценариев. Здесь мы вводим новую команду `epogroup.findSystems`, которая находит все системы в данной группе. Последний параметр (`'true'`) определяет, выполнять ли при этом поиск во всех подгруппах. В данном случае мы задаем этому параметру значение `true`, проходя по всем системам, найденным в группе `Потерянные и найденные` и во всех ее подгруппах, а также повторно применяя к ним правила сортировки.

3 Удаленные запросы

Удаленные команды ePolicy Orchestrator позволяют выполнять запросы к базе данных удаленно с помощью веб-API. Эти команды позволяют выполнять постоянные запросы, существующие в базе данных ePolicy Orchestrator, а также динамические, определяемые пользователем нерегламентированные запросы.

Содержание

Постоянные запросы

Постоянный запрос — это запрос, доступный со страницы **Запросы и отчеты** в пользовательском интерфейсе ePolicy Orchestrator.

К постоянным запросам относятся как предварительно установленные запросы, так и запросы, создаваемые пользователями. Чтобы удаленно выполнить постоянный запрос, необходимо знать его идентификатор.

Примеры запросов

Во всех примерах запросов первые две строки содержат URL-адрес и Python-формы команды. Например, иллюстрация команды `core.listQueries` имеет следующий вид:

```
URL: https://имя_сервера:порт/remote/core.listQueries
Python: mc.core.listQueries();
```

Пример URL-адреса можно ввести непосредственно в адресной строке обозревателя:

```
https://localhost:8443/remote/core.listQueries
```

Также возможно использование примера URL-адреса с командой cURL:

```
> curl -k -u ga:ga https://localhost:8443/remote/core.listQueries
```

Этот пример Python можно использовать следующим образом:

```
import mcafee
mc = mcafee.client('localhost','8443','ga','ga')
mc.core.listQueries();
```

Поиск доступных запросов

Для выполнения любого удаленного запроса необходимо выяснить доступные запросы и идентификаторы запросов.

Команда `core.listQueries` используется для получения идентификатора любого постоянного запроса, к которому пользователь может получить доступ.

```
URL: https://имя_сервера:порт/remote/core.listQueries
Python: mc.core.listQueries();

OK:
ИД: 1
Имя: Действующие разрешения пользователей
Описание: Отображение всех разрешений для всех пользователей
Критерии: ( where ( ne EntitlementView.RoleUri "%%NOEPOROLES%%" ) )
Имя группы: Запросы разрешений
Владелец: ga
Тип базы данных:
Цель: EntitlementView
Создано: ga
Дата создания: 10/25/10 8:40:33 AM PDT
Изменено: ga
Дата изменения: 10/25/10 8:40:33 AM PDT

ИД: 2
Имя: Сведения о наборах разрешений
Описание: Отображение разрешений, связанных со всеми наборами разрешений
```

Удаленное выполнение запроса

Выполните запрос, используя команду `core.executeQuery` с параметром `queryId`, когда узнаете идентификатор запроса. В данном примере в качестве идентификатора используется значение 5.

```
URL: https://имя_сервера:порт/remote/core.executeQuery?queryId=5
Python: mc.core.executeQuery('5');

OK:
Имя пользователя: ga
Действие: Создать ответ
Успешно: true
Время начала: 10/26/10 9:00:24 AM PDT

Имя пользователя: ga
Действие: Создать ответ
Успешно: true
Время начала: 10/26/10 9:00:24 AM PDT
```

Нерегламентированные запросы

Нерегламентированные запросы выполняются полностью удаленно и не используют запрос, хранящийся в базе данных ePolicy Orchestrator.

В нерегламентированном запросе нужно указать цель запроса и до четырех параметров из данной таблицы.

Таблица 3-1 Параметры нерегламентированного запроса

Оператор	Описание	Поведение в случае пропуска
`select`	Позволяет выбрать столбцы из целевой таблицы (и любых присоединенных таблиц), возвращаемые запросом.	Возвращаются все столбцы целевой таблицы.
`condition`	Параметр `condition` фильтрует результаты. Возвращаются только записи из базы данных, удовлетворяющие предложению фильтрации.	Возвращаются все строки.
`group`	Управляет группированием возвращаемых данных.	Возвращаемые данные не группируются.
`order`	Контролирует порядок сортировки возвращаемых данных (либо возрастающий, либо убывающий по столбцам).	Возвращаемые строки упорядочиваются в соответствии с естественным порядком базы данных.

С помощью команд `core.listTables`, `core.listDatabases` и `core.listDatatypes` можно определить имена и типы столбцов целевой таблицы. Эта информация должна облегчить определение столбцов, которые следует выбрать, и операций, разрешенных в данном условии.

Примеры нерегламентированных запросов

Это простой нерегламентированный запрос к таблице `OrionAuditLog`.

```
URL: https://имя_сервера:порт/remote/core.executeQuery?target=OrionAuditLog&select=(select
OrionAuditLog.UserName OrionAuditLog.CmdName)

Python: mc.core.executeQuery(target="OrionAuditLog", select="(select OrionAuditLog.UserName
OrionAuditLog.CmdName)");

OK:
Имя пользователя:
Действие: Перезапуск сервера

Имя пользователя: ga
Действие: Попытка входа

Имя пользователя: ga
Действие: Выгрузить расширение
```

Данный запрос возвращает `CmdName` (имя команды) и `EndTime` (время окончания) для всех записей журнала аудита. Результаты группируются в алфавитном порядке по параметру `CmdName` (имя команды), затем по параметру `EndDate` (время окончания).

```
URL: https://имя_сервера:порт/remote/core.executeQuery?target=OrionAuditLog& select=(select
OrionAuditLog.CmdName.OrionAuditLog.EndTime)&
group=(group.OrionAuditLog.CmdName OrionAuditLog.EndTime)

Python: mc.executeQuery(target="OrionAuditLog", select="(select OrionAuditLog.CmdName
OrionAuditLog.EndTime)",
group="(group OrionAuditLog.CmdName OrionAuditLog.EndTime)");

OK:
Имя пользователя: ga
Приоритет: 1
```

```
Действие: Попытка входа
Сведения: Не удалось выполнить вход пользователя "ga" с IP-адреса: 172.1.6.1
Успешно: false
Время начала: 10/11/12 4:41:18 PM PDT
Время завершения: 10/11/12 4:41:18 PM PDT

Имя пользователя: system
Приоритет: 1
Действие: Перезапуск сервера
Сведения: Сервер запущен.
Успешно: true
Время начала: 10/11/12 4:41:42 PM PDT
Время завершения: 10/11/12 4:41:42 PM PDT
```

Данный запрос возвращает все записи OrionTaskLog, которые были созданы пользователем ga. Результаты перечисляются в порядке возрастания по задаче `StartDate` (дата начала).

```
https://имя_сервера:порт/remote/core.executeQuery?target=OrionTaskLogTask& where=(where (eq
( OrionTaskLogTask.UserName "ga" ))) & order=(order (asc OrionTaskLogTask.StartDate) )

Python: mc.core.executeQuery(target="OrionTaskLogTask", where="(where ( eq
( OrionTaskLogTask .UserName "ga" )))", order="(order (asc OrionTaskLogTask.StartDate) )");

OK:
Имя: Развернуть McAfee Agent
Дата начала: 10/11/12 5:00:01 PM
Дата окончания: 10/11/12 5:00:38 PM PDT
Имя пользователя: ga
Статус: 0
Источник: планировщик
Продолжительность: 36846

Имя: Развернуть McAfee Agent
Дата начала: 10/11/12 5:04:19 PM PDT
Дата окончания: null
Имя пользователя: ga
Статус: 10
Источник: планировщик
Продолжительность: 1889951790
```

ePolicy Orchestrator использует символические выражения (S-выражения) для внутренних целей в качестве портативной и независимой от баз данных схемы для определения запросов и операций. Значения, передаваемые каждому параметру, должны быть допустимыми S-выражениями.

Создание нерегламентированного запроса из определения запроса

Запросы, хранящиеся в ePolicy Orchestrator, можно экспортировать и дублировать в сценарии.

Если у вас есть постоянный запрос, и он определен как нерегламентированный запрос с помощью `core.executeQuery`, используйте для получения внутреннего представления запроса действие **Экспортировать определения** в ePolicy Orchestrator. Почти во всех случаях экспортированное определение можно использовать для построения вызова метода `core.executeQuery`. Например, начав с существующего запроса в качестве модели, измените параметры, отфильтровав или отсортировав их при выполнении запроса из сценария.

Ниже приведен пример использования экспортированного постоянного запроса для создания нерегламентированного запроса.

Пример

Типичное экспортированное определение запроса:

```
<query>
<name language="en">My AuditLogQuery</name>
<description language="en"></description>
<property name="target">OrionAuditLog</property>
<property name="tableURI">query:table?orion.table.columns=OrionAuditLog.UserName
%3AOrionAuditLog.CmdName%3A
OrionAuditLog.Success
%3AOrionAuditLog.StartTime&amp;orion.table.order.by=OrionAuditLog.CmdName
&amp;orion.table.order=asc</property>
<property name="conditionURI">query:condition?orion.condition.sexp=%28+where+%28+olderThan+
OrionAuditLog.EndTime+3600000++%29+%29</property>
<property name="summaryURI">query:summary?
orion.sum.query=false&amp;orion.query.type=table.table</property>
</query>
```

Это определение состоит из следующих компонентов:

- Атрибут `target` используется непосредственно как параметр `target` нерегламентированного запроса.
- Атрибут `conditionURI` содержит S-выражение, которое можно использовать как параметр `where`.

В S-выражении предложение SELECT отражает ограничения предложения SELECT SQL. Операции предложения SELECT включают столбцы и унарные операции со столбцами таблицы. Например, Count, Max, Top и прочие.

Унарные операторы работают только с одним выражением одного из типов данных числовой категории. Например, невозможно использовать SUM или другую операцию объединения с предложением SELECT.

Чтобы понять, какие аргументы предложения SELECT поддерживаются и каковы их ограничения в S-выражении нерегламентированного запроса, достаточно экспортировать запросы и изучить их структуру.

Помните, что экспортированная форма запроса содержит строки, подвергнутые URL-кодированию. Чтобы сформировать допустимую строку запроса, потребуется декодировать URL-кодированные символы. Например:

- «+» используется для обозначения одного пробела « »;
- `%28` — открывающая круглая скобка «(»;
- `%29` — закрывающая круглая скобка «)»;
- `%3A` — двоеточие «:».

Ниже представлен эквивалентный нерегламентированный запрос URL-адреса с использованием экспортированного определения запроса:

```
https://имя_сервера:порт/remote/core.executeQuery?target=OrionAuditLog&select=(select
OrionAuditLog.UserName OrionAuditLog.CmdName OrionAuditLog.Success
OrionAuditLog.StartTime)&where=(where(olderThan OrionAuditLog.EndTime
36000000))&order=(order(asc OrionAuditLog.CmdName))
```

Ниже представлен эквивалентный нерегламентированный запрос Python с использованием экспортированного определения запроса:

```
mc.core.executeQuery(target="OrionAuditLog",
select="(select OrionAuditLog.UserName OrionAuditLog.CmdName OrionAuditLog.Success
OrionAuditLog.StartTime)",
where="(where(olderThan OrionAuditLog.EndTime 36000000))",
order="(order(asc OrionAuditLog.CmdName))");
```

Этот эквивалентный нерегламентированный запрос возвращает следующий вывод:

```
OK:
Имя пользователя: ga
Приоритет: 1
Действие: Попытка входа
Сведения: Не удалось выполнить вход пользователя "ga" с IP-адреса: 172.1.5.1
Успешно: false
Время начала: 10/11/12 4:41:18 PM PDT
Дата завершения: 10/11/12 4:41:18 PM PDT

Имя пользователя: system
Приоритет: 1
Действие: Перезапуск сервера
Сведения: Сервер запущен.
Успешно: true
Время начала: 10/11/12 4:41:42 PM PDT
Дата завершения: 10/11/12 4:41:42 PM PDT
.
.
.
```

Получение информации о зарегистрированных базах данных и таблицах

С помощью команд удаленных запросов можно получить информацию о зарегистрированных базах данных и таблицах. Эта информация необходима для создания нерегламентированных запросов.

С помощью команды `core.listTables` можно получить подробные сведения о каждой таблице в системе, включая имена и типы столбцов.

Пример вывода

Следующий вывод является результатом выполнения команды `core.listTables` над таблицей `OrionAuditLog`.

```
Имя: Записи в журнале аудита
Цель: OrionAuditLog
Тип: target
Тип базы данных:
Описание: Получает сведения об изменениях и действиях, совершаемых пользователями сервера.
Столбцы:
Имя       Тип           Выбор? Условие? Группировка? Порядок? Число?
--------- ------------- ------- ---------- -------- ------ -------
AutoId    int           false   false      false    true   true
UserId    int           false   false      false    true   true
UserName  string_lookup true    true       true     true   false
Priority  enum          true    true       true     true   false
CmdName   string_lookup true    true       true     true   false
Message   string        true    true       false    true   false
Success   boolean       true    true       true     true   false
StartTime timestamp     true    true       true     true   false
EndTime   timestamp     true    true       true     true   false
Related Tables:
```

```
Name
----
Внешние ключи: Отсутствует
```

Команда `core.listTables` выводит список столбцов, их типы и возможность использования столбца в параметрах `select`, `condition`, `group` и `order`, а также указание, является ли столбец числом. Также команда перечисляет любые зарегистрированные связанные таблицы, которые могут быть присоединены с помощью параметра `joinTables`.

Получение информации о зарегистрированных базах данных

С помощью команды `core.listDatabases` можно получить подробные сведения о каждой базе данных системы. Эту информацию в дальнейшем можно использовать для создания нерегламентированных запросов.

В общем случае при выпуске запросов к целям, не являющимся частью схемы по умолчанию, следует добавить имя базы данных в начале имени цели. Например, для обращения к цели `"Политика_применяемая_к_стороннему_пользователю"`, являющейся частью базы данных `"Сторонний_пользователь"`, следует использовать идентификатор `"target=Сторонний_пользователь.Политика_применяемая_к_стороннему_пользователю"`.

Запросы с объединениями

Для отображения данных из двух и более таблиц можно использовать объединения. Объединения обрабатываются автоматически.

Чтобы использовать функцию объединения, необходимо указать таблицы (`targets`) и требуемые столбцы из этих таблиц в параметре `select` команды `core.executeQuery`. Базовая система запросов прозрачно вычисляет необходимые критерии объединения и отображает правильные результаты.

При отсутствии зарегистрированной информации об объединении для указанных таблиц запрос возвращает ошибку.

С помощью команды `core.listTables` можно определить, какие таблицы взаимосвязаны и могут быть использованы в объединениях. Эта информация содержится в свойстве таблицы `relatedTables`.

Пример запроса с объединением

Данный пример выполняет простое объединение между таблицами `OrionTaskLogTask` и `OrionTaskLogMessage`.

```
URL: https://имя_сервера:порт/remote/core.executeQuery?
target=OrionTaskLogTaskMessage&select=(select OrionTaskLogTask.Name
OrionTaskLogTaskMessage.Message )&joinTables=OrionTaskLogTask

Python: mc.core.executeQuery(target="OrionTaskLogTaskMessage", select="(select
OrionTaskLogTask.Name OrionTaskLogTaskMessage.Message )", joinTables="OrionTaskLogTask")

OK:
Имя: Создать задачу
Сообщение: Очистить журнал аудита

Имя: Создать задачу
Сообщение: Очистить журнал аудита (Очистить записи в журнале старше: 1 дней)
```

Извлечение иерархических результатов запроса

Запросы с задействованием объединенных таблиц могут возвращать результаты в иерархическом виде.

Команда `core.executeQuery` может также использоваться в режиме `joinTables`. В этом режиме указываются только таблицы, которые необходимо объединить. Результаты запроса используются в качестве ключей для выполнения субзапроса всех связанных результатов из объединенной таблицы. В объединенной таблице создается иерархический набор результатов, который может содержать вложенные результаты. Результаты вкладываются до уровня, который зависит от числа объединенных таблиц.

Для объединения таблиц с помощью команды `core.executeQuery` следует указать разделенный запятыми список объединяемых таблиц в качестве параметра `joinTables`. При объединении таблиц не указывается параметр select. Результаты возвращаются в виде иерархии результатов, где каждая запись родительской таблицы становится родительским узлом для связанных записей в каждой дочерней таблице. Эта иерархия продолжается до тех пор, пока не будут отображены все объединенные таблицы.

Пример

Данный пример выполняет простое объединение таблиц `OrionTaskLog` и `OrionTaskLogMessage`.

```
OK:
<?xml version="1.0" encoding="UTF-8"?>
<result>
<list>
<row>
<OrionTaskLogTask.Name>Создать задачу</OrionTaskLogTask.Name>
<OrionTaskLogTask.StartDate>2010-11-23T13:01:37-08:00</OrionTaskLogTask.StartDate>
<OrionTaskLogTask.EndDate>2010-11-23T13:01:37-08:00</OrionTaskLogTask.EndDate>
<OrionTaskLogTask.UserName>ga</OrionTaskLogTask.UserName>
<OrionTaskLogTask.Status>0</OrionTaskLogTask.Status>
<OrionTaskLogTask.TaskSource>scheduler</OrionTaskLogTask.TaskSource>
<OrionTaskLogTask.Duration>493</OrionTaskLogTask.Duration>
<OrionTaskLogTaskMessage>
<list>
<row>
<OrionTaskLogTaskMessage.Message>Очистить журнал аудита</OrionTaskLogTaskMessage.Message>
</row>
<row>
<OrionTaskLogTaskMessage.Message>Очистить журнал аудита (Очистить записи в журнале старше: 1
дней)</OrionTaskLogTaskMessage.Message>
</row>
</list>
</OrionTaskLogTaskMessage>
</row>
</list>
</result>
```

По выводу можно увидеть, что команда `core.executeQuery` вернула каждый объект верхнего уровня (запись `OrionTaskLogTask`) и две связанные с ним записи сообщений. Вывод показан в виде XML-кода, чтобы продемонстрировать иерархическую организацию результатов.

Ограничение глубины результатов запроса

Запросы, объединяющие таблицы, могут возвращать результаты с глубокими иерархиями. Глубиной иерархии можно управлять с помощью параметров.

Запрос, в котором не используются объединения, возвращает табличные результаты. Табличные результаты определяются как имеющие глубину, равную единице. Запросы, в которых используется параметр `joinTable`, возвращают иерархические результаты. Указание более одной таблицы в параметре `joinTable` может привести к многоуровневому набору результатов. Каждый уровень объектов увеличивает глубину результатов на единицу. Для предотвращения

чрезмерной глубины наборов результатов команда `core.executeQuery` по умолчанию предусматривает максимум 5 уровней глубины. Если лимит, заданный по умолчанию, чересчур ограничивает выполняемый запрос, можно изменить его, указав новое значение параметра `depth`.

Нерегламентированные запросы, возвращающие глубокие наборы результатов, могут выполняться длительное время, а для их создания может требоваться значительный объем системных ресурсов. Помните об этом, объединяя более двух таблиц или увеличивая лимит глубины результатов запроса.

Команды удаленных запросов

Существует небольшое число команд, которые можно использовать при выполнении удаленных запросов.

Укажите аргументы по именам, а затем добавьте =<>. Например, в команду core должен быть включен аргумент «target=». Например, `executeQuery?target=EntitlementView`

Таблица 3-2 Команды удаленных запросов

Команда	Синтаксис	Описание
`core.executeQuery`	`core.executeQuery queryId [database=<>]` `core.executeQuery target=<> [select=<>] [where=<>] [order =<>] [group=<>] [database=<>] [depth=<>] [joinTables=<>]`	Выполняет запрос и возвращает результаты в виде списка объектов.
`core.listDatabases`	`core.listDatabases`	Возвращает все базы данных, разрешенные для просмотра пользователю, в виде списка объектов.
`core.listTables`	`core.listTables [table]`	Возвращает все таблицы баз данных, разрешенные для просмотра пользователю, в виде списка объектов.
`core.listQueries`	`core.listQueries`	Возвращает все запросы, разрешенные для просмотра пользователю, в виде списка объектов.
`core.listDatatypes`	`core.listDatatypes`	Возвращает список всех типов и поддерживаемые ими операции.

Справка по нерегламентированным запросам

Нерегламентированным запросам требуются операторы, типы данных и другая информация, которая может быть получена из сценариев.

В общем, тип данных в столбце определяет операторы, поддерживаемые в этом столбце. Например:

- Тип данных `string` поддерживает только операции `startsWith` и `endsWith`.
- Столбцы с цифрами поддерживают операции `gt` и `lt`.

Типы данных и типы столбцов в целевой таблице могут быть определены с помощью команды `core.listTables`.

В этих таблицах рассматриваются базовые элементы и приводятся примеры их использования.

Типы данных общих запросов

Данные в базах данных ePolicy Orchestrator следует сохранять с конкретными типами в удаленных запросах.

Таблица 3-3 Типы данных общих запросов

Тип	Описание
Int	Целое число
string_lookup	Строковое поле подстановки
Enum	Перечисляемое значение, хранимое в базе данных в виде целого числа
Mac	MAC-адрес
Long	Значение длинного типа
Float	Значение с плавающей запятой
Timespan	Интервал времени SQL
boolean	Логическое значение
string_enum	Перечисляемое значение, хранящееся в виде строки, а не целого числа
ipv4	IPv4-адрес
ipv6	IPv6-адрес

Общие операции с S-выражениями

ePolicy Orchestrator использует S-выражения для внутренних целей для определения запросов и операций. Эти S-выражения следует использовать корректно.

В следующих таблицах указаны операторы, которые можно использовать в S-выражениях для определения запросов.

Таблица 3-4 Общие операции с S-выражениями

Оператор	Описание	Пример
and	Логическое И из двух или нескольких S-выражений.	`(where (and (eq OrionAuditLog .UserName "ga") (eq OrionAuditLog .Priority 1)))`
or	Логическое ИЛИ из двух или нескольких S-выражений.	`(where (or (eq OrionAuditLog .UserName "ga") (eq OrionAuditLog .UserName "admin")))`
not	Логическое отрицание S-выражения.	`(where (not (eq OrionAuditLog .UserName "ga")))`
eq	Логическое сравнение на равенство.	`(where (eq OrionAuditLog.UserName "ga"))`
ne	Логическое сравнение на неравенство.	`(where (ne OrionAuditLog.UserName "ga"))`

Таблица 3-4 Общие операции с S-выражениями *(продолжение)*

Оператор	Описание	Пример
gt	Логическое сравнение «больше».	`(where (gt OrionAuditLog.Priority 1))`
lt	Логическое сравнение «меньше».	`(where (lt OrionAuditLog.Priority 3))`
ge	Логическое сравнение «больше или равно».	`(where (ge OrionAuditLog.Priority 2))`
le	Логическое сравнение «меньше или равно».	`(where (le OrionAuditLog.Priority 2))`
is_Blank	Возвращает строку, если значение столбца равно null или усеченное значение является пустым.	`(where (isBlank OrionAuditLog .UserName))`
not_isBlank	Возвращает строку, если значение столбца не равно null или усеченное значение не является пустым.	`(where (not_isBlank OrionAuditLog .UserName)))`
in	Возвращает строку, если следующий элемент (обычно свойство или значение) содержится в следующем списке. Это подобно директиве SQL `IN`.	`(where (in OrionAuditLog.UserName "ga" "bob"))`
contains	Возвращает строку, если значение столбца содержит значение аргумента подстроки.	`(where (contains OrionAuditLog .UserName "ga"))`
notContains	Возвращает строку, если значение столбца не содержит значение аргумента подстроки.	`(where (notContains OrionAuditLog .UserName "ga"))`
startsWith	Возвращает строку, если строковое значение начинается с указанного строкового значения. Подобно столбцу `in s%` в SQL.	`(where (startsWith OrionAuditLog .UserName "g"))`
endsWith	Возвращает строку, если столбец заканчивается значением аргумента.	`(where (endsWith OrionAuditLog .UserName "a"))`
like	Возвращает строку, если столбец содержит значение, соответствующее шаблону. Подобно `like` в SQL.	`(where (like OrionAuditLog.UserName "ga"))`
newerThan	Возвращает строку, если первый параметр метки времени новее второго параметра метки времени.	`(where (newerThan OrionAuditLog .EndTime 3600000))`
olderThan	Возвращает строку, если первый параметр метки времени старше второго параметра метки времени.	`(where (olderThan OrionAuditLog .EndTime 36000000))`
between	Возвращает строку, если аргумент метки времени или IP-адреса (столбец или значение) лежит между двумя значениями.	`(where (between OrionAuditLog .EndTime (timestamp 1288888320000) (timestamp 1288888360000)))`
beforeNow	Возвращает строку, если значение метки времени (столбец или значение) предшествует текущему времени.	`(where (beforeNow OrionAuditLog .EndTime))`
match_any	Возвращает строку, если IP-адрес совпадает с одним из адресов из следующего списка.	`(where(match_any EPOComputerProperties.IPV4x (ipv4 "192.168.1.1")))`

Таблица 3-4 Общие операции с S-выражениями *(продолжение)*

Оператор	Описание	Пример
notBetween	Возвращает строку, если IP-адрес не находится между двумя адресами из аргументов.	`(where (notBetween EPOComputerProperties.IPV4x (ipv4 "192.168.1.1")(ipv4 "192.168.255.255")))`
not_in_subnet	Возвращает строку, если адрес не находится в заданной подсети.	`(where (not_in_subnet EPOComputerProperties.IPV4x (ipv4 "255.255.255.0" ) 25 ))`
in_subnet	Возвращает строку, если адрес находится в заданной подсети.	`(where (in_subnet EPOComputerProperties.IPV4x (ipv4 "255.255.255.0" ) 25 ))`
in_ipv6_subnet	Возвращает строку, если адрес находится в подсети IPv6.	`(where (not_in_ipv6_subnet EPOComputerProperties.IPV6 (ipv6 "0.0.0.0") 1))`
not_match_any	Возвращает строку, если IPv6-адрес из аргумента не соответствует ни одному из адресов в аргументах.	`(where(not_match_any EPOComputerProperties.IPV6 (ipv6 "fc00::/7")))`

Таблица 3-5 S-выражения только для выборки

Оператор	Описание	Пример
Distinct	Выбирает записи, содержащие отличительное выходное значение в заданном столбце. Подобно директиве SQL `distinct`.	`(select (distinct) OrionAuditLog.UserName)`
	При использовании предложения `order` с `select distinct` необходимо включить в выборку столбцы `order`.	
Top N	Выбирает первые N записей для отображения. N должно быть целым числом. Подобно предложению Microsoft SQL `Top` или предложению MySQL `Limit`.	`(select (top 5) OrionTaskLogTask.Name OrionTaskLogTask.StartDate)`

Комбинации операторов и типов данных S-выражений

С каждым оператором S-выражения можно использовать только определенные типы данных. В следующей таблице можно проверить правильность использования типов данных и операторов.

Operator	int	long	float	string	enum	mac	timestamp	timespan	string_enum	boolean	string_lookup	issue_type	ipv4	ipv6
eq	x	x		x	x				x	x	x	x	x	x
ne	x	x		x	x				x	x	x	x	x	x
gt	x	x	x											
lt	x	x	x											
ge	x	x												
le	x	x												
startsWith				x							x			
endsWith				x							x			
like				x							x			
in														
isBlank				x			x				x		x	x
not_isBlank				x			x				x		x	x
contains				x							x			
not_contains				x							x			
olderThan							x							
newerThan							x							
olderThanAbsolute							x							
newerThanAbsolute							x							
between							x						x	x
beforeNow							x							
mac_match_any						x								
mac_not_match_any						x								
match_any													x	x
notBetween													x	x
in_ipv6_subnet														x
not_match_any														x
not_in_ipv6_subnet														x

Специальные типы данных ePolicy Orchestrator

ePolicy Orchestrator создает ряд специальных типов данных, используемых для различных объектов.

Таблица 3-6 Специальные типы данных ePolicy Orchestrator

Тип	Описание
applied_tags	Используется для столбца `EPOLeafNode.AppliedTags`, чтобы определить, какие теги применены к системе.
byte	Используется для данных об объеме памяти, например `EPOComputerProperties.TotalPhysicalMemory` и `EPOComputerProperties.FreeMemory`.
eventID	Используется для идентификации событий, например событий угроз.
group	Используется для групп в дереве систем. Например, `Groups .L1ParentID`.
multiselect_group	Используется для выбора нескольких групп дерева систем.
managedState	Логическое значение, используемое для управляемого статуса устройства в дереве систем.

Таблица 3-6 Специальные типы данных ePolicy Orchestrator *(продолжение)*

Тип	Описание
Megabytes	Используется для данных об объеме дисковой памяти, например `EPOComputerProperties.FreeDiskSpace` и `EPOComputerProperties.TotalDiskSpace`.
optionGroup_enum	Перечисление группированных списков, например категорий политик.
Percentagefromstring	Используется для процентных значений и столбцов соответствия.
Policycolumn	Используется для политик.
productVersion	Используется для номеров версий. Поведение при использовании функций равенства стандартное, но поддерживается обработка числовых строк с более чем одной десятичной точкой.
Rsdmac	Используется для MAC-адресов. Чаще всего связан с обнаруживаемыми системами.
Rsdoui	Используется для OUI, связанных с обнаруженными системами.
string_lookupWithResolver	Используется для разрешения пользователю выбора из списка известных строковых значений.
Threatcategory	Группированное перечисление, используемое для вывода списка доступных категорий угроз.
DATversion	Используется для информации о версии DAT-файла.
engineVersion	Используется для информации о версии ядра.
datVersion	Используется для информации о версии DAT-файла. Идентичен типу DATversion.

Специальные операторы ePolicy Orchestrator

ePolicy Orchestrator определяет ряд операторов, предназначенных для работы с собственными специальными типами данных этой программы.

Таблица 3-7 Специальные операторы ePolicy Orchestrator

Оператор	Описание	Пример
hasTag	Вычисляется как true, если указанный тег применен к системе.	`(hasTag EPOLeafNode.AppliedTags 'полное_имя_тега')`
containsTag	Вычисляется как true, если какие-либо теги, примененные к системе, содержат указанное частичное имя тега.	`(containsTag EPOLeafNode .AppliedTags 'частичное_имя _тега')`
hasTagExcluded	Вычисляется как true, если указанный тег исключен в системе.	`(hasTagExcluded EPOLeafNode .AppliedTags 'исключенное_имя _тега')`
doesNotHaveTag	Вычисляется как true, если указанный тег не применен к системе.	`(doesNotHaveTag EPOLeafNode .AppliedTags 'полное_имя_тега')`
doesNotHaveAnyTag	Вычисляется как true, если к системе не применены никакие теги.	`(doesNotHaveAnyTag EPOLeafNode .AppliedTags)`
childOf	Возвращает строки, являющиеся прямыми потомками заданных узлов.	`(childOf EPOLeafNode.parentId <ИД_узла1> <ИД_узла2> ...)`

Таблица 3-7 Специальные операторы ePolicy Orchestrator *(продолжение)*

Оператор	Описание	Пример
descendsFrom	Возвращает строки, являющиеся потомками заданных узлов.	`(descendsFrom EPOLeafNode .parentId <ИД_узла1> <ИД_узла2> ...)`
stateEq	Вычисляется как true, если управляемое состояние системы соответствует одному из значений "managed" или "unmanaged".	`(stateEq EPOLeafNode .ManagedState ["managed"|"unmanaged"])`
version_eq	Вычисляется как true, если версии равны при использовании нотации версий с точками.	`(version_eq EPOMasterCatalog .ProductVersion '3.1.4.1')`
version_neq	Вычисляется как true, если версии не равны при использовании нотации версий с точками.	`(version_neq EPOMasterCatalog .ProductVersion '3.1.4.1')`
version_ge	Вычисляется как true, если первая версия больше или равна второй при использовании нотации версий с точками.	`(version_ge EPOMasterCatalog .ProductVersion '3.1.4.1')`
version_lt	Вычисляется как true, если первая версия меньше второй при использовании нотации версий с точками.	`(version_lt EPOMasterCatalog .ProductVersion '3.1.4.1')`
threatcategory_belongs	Возвращает строку, если категория угрозы принадлежит к заданной группе.	`(threatcategory_belongs EPOEvents.ThreatCategory 'av')`
threatcategory_not_belongs	Возвращает строку, если категория угрозы не принадлежит к заданной группе.	`(threatcategory_not_belongs EPOEvents.ThreatCategory 'av')`
withinRepositoryDatVersion	Вычисляется как true, если версия DAT-файла находится в пределах указанного числа версий до той, что содержится в репозитории.	`(withinRepositoryDatVersion EPOProdPropsView_VIRUSCAN .datver 3)`
not_withinRepositoryDatVersion	Вычисляется как true, если версия DAT-файла не находится в пределах указанного числа версий до той, что содержится в репозитории.	`(not_withinRepositoryDatVersion EPOProdPropsView_VIRUSCAN .datver 3)`

Комбинации специальных операторов и типов данных ePolicy Orchestrator

Можно использовать только сочетания типов данных и операторов, определенные в ePolicy Orchestrator и указанные в данной таблице.

	applied_tags	byte	eventId	group	multiselect_group	managedState	megabytes	optionGroup_enum	Percentagefromstring	Policycolumn	productVersion	Rsdmac	Rsdoui	String_lookupWithResolver	Threatcategory	DATversion	engineVersion	datVersion
eq		x	x				x	x	x	x		x	x	x		x		
ne		x	x				x	x	x	x		x	x	x		x		
gt		x	x				x		x									
lt		x	x				x		x							x		
ge		x	x				x		x							x		
le		x	x				x		x									
in			x						x									
hasTag	x																	
containsTag	x																	
hasTagExcluded	x																	
doesNotHaveTag	x																	
doesNotHaveAnyTag	x																	
childOf				x	x													
descendsFrom				x	x													
stateEq						x												
version_eq											x						x	x
version_neq											x						x	x
version_ge											x						x	x
version_lt											x						x	x
isBlank												x		x				
not_isBlank												x		x				
startsWith												x	x	x				
notStartsWith												x	x	x				
contains														x				
notContains														x				
threatcategory_belongs															x			
threatcategory_not_belongs															x			
withinRepositoryDatVersion																x		
not_withinRepositoryDatVersion																x		

Указатель

О

П

Т

У