

Синхронизация ACL и структуры организации

Часть 3

Вадим Андросов

В статье приводится окончательная реализация надстройки, позволяющей поддерживать в согласованном виде списки контроля доступа и структуру организации. Описаны механизмы ее установки и отключения. Подробно рассмотрены основные приемы создания постоянных обработчиков событий на основе сценариев в Windows 2003 Server.

Создание постоянного обработчика событий

Теперь необходимо обеспечить функционирование прикрепленных групп, так как штатных средств автоматического добавления пользователей в группы нет. Как уже говорилось, для обработки событий не будут использоваться временные подписчики. Рассмотрим основные шаги регистрации постоянного обработчика событий, который не требует постоянного функцио-

нирования сценария и продолжает работать после перезагрузки системы. Этот вариант функционирует на основе COM-объектов. Операционная система уже содержит необходимые классы для такой популярной операции, как запуск кода сценария в ответ на событие. Пользовательские классы здесь мы будем использовать только для удобства установки надстройки. Воспользуемся паттерном «Делегирование» [3], добавив в стандарт-

ные классы вызовы методов нашего. То есть все сценарии, выполняемые в ответ на то или иное событие, будут состоять из одинаковых шагов:

- Создание объекта класса, инкапсулирующего реакцию на событие.
- Инициализация объекта полным именем источника события.
- Вызов необходимого метода.

Первым делом определимся с общей настройкой постоянного обработ-

чика событий, а затем подробнее рассмотрим тонкости реализации классов-обработчиков. Для простоты было принято решение остановиться на одном классе-обработчике для всех событий, связанных с интересующими нас объектами операционной системы.

■ **Организационная единица.** При переносе объектов этого типа требуется подключить ее прикрепленную группу типа Tree к такой группе родительской организационной единицы. Поскольку предполагается, что прикрепленные группы находятся в своих организационных единицах, при удалении подразделения группы будут автоматически удалены ОС. Также в случае переноса организационной единицы группа отсоединяется от своего старого контейнера.

■ **Группы.** Требуется обрабатывать событие удаления группы. В случаях, если это прикрепленная к организационной единице группа, она должна быть корректно отключена (должен быть удален указатель на эту группу). При этом для случая группы типа Tree на место удаленной группы ставится указатель на такую группу родительского подразделения.

■ **Пользователи.** Операции над этими объектами нужно отслеживать для поддержки функциональности групп типа Single. При добавлении пользователя в организационную единицу он должен быть автоматически зарегистрирован в прикрепленной группе. Аналогично при удалении его требуется удалить из группы. Простое удаление работало бы корректно и без специальной обработки (при удалении пользователя он и так пропадает из всех содержащих его групп), однако здесь важно обрабатывать событие удаления, возникающее при перемещении объекта пользователя между отделами (перемещение – это на самом деле последовательные операции удаления и добавления).

Теперь сконцентрируем внимание на регистрации постоянного подписчика на события. Для подписки необходимо создать, необходимым образом проинициализировать и записать в хранилище объекты трех классов.

■ **__EventFilter.** Фильтр событий. С его помощью настраивается получение только нужных событий.

■ **__EventConsumer.** Обработчик (потребитель) события. В этом классе реализуется логика реакции на событие. Для непосредственной обработки воспользуемся специальным классом `ActiveScriptEventConsumer`, который будет делегировать [3] основную деятельность нашему классу.

■ **__FilterToConsumerBinding.** Связывает два предыдущих класса друг с другом, т.е. содержит информацию, какой объект заинтересован в получении определенных событий.

Обработчики событий каждого вида создаются практически одинаково, поэтому подробно рассмотрим лишь один – слежение за манипуляциями с объектами пользователей. Существует несколько способов создания необходимых объектов, остановимся на реализации с помощью сценариев. Сначала требуется подключиться к пространству имен. Поскольку мы работаем с объектами Active Directory,

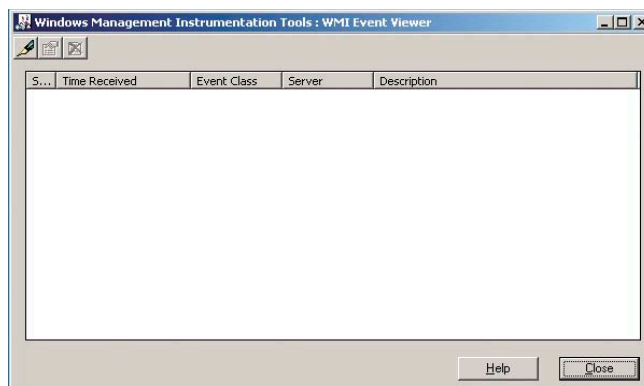


Рисунок 1. WMI Event Viewer

все интересующие объекты находятся в пространстве `root\directory\LDAP`.

Итак, подключаемся:

```
path = "\\.\root\directory\LDAP"
Set objSWbemServices = GetObject("winmgmts:" & _
    "{impersonationLevel=impersonate}!" & path)
```

Затем требуется подключиться к классам, объекты которых мы будем создавать:

```
Set eventFilterClass = _
    objSWbemServices.Get("__EventFilter")
Set consumerClass = _
    objSWbemServices.Get("ActiveScriptEventConsumer")
Set bindingClass = _
    objSWbemServices.Get("__FilterToConsumerBinding")
```

Классы, начинающиеся с двойного подчеркивания, – стандартные системные и размещены сразу во всех пространствах имен. Сложнее дело обстоит с классом `ActiveScriptEventConsumer`. В пространстве имен `root\directory\LDAP` его нет. Поэтому требуется скомпилировать специальный mof-файл, который поставляется вместе с операционной системой [9]:

```
mofcomp -N:root\directory\LDAP _
    %SYSTEMROOT%\system32\wbem\scrcons.mof
```

Ключ N как раз и необходим для регистрации скомпилированного класса в нужном пространстве имен. После этого можно создавать объекты необходимых классов. Классы поддерживают фабричный метод [3] `SpawnInstance_`, порождающий свои экземпляры. Итак, сначала создаем фильтр:

```
set userFilter = eventFilterClass.SpawnInstance_()
userFilter.Name = "UserOperationsFilter"
userFilter.QueryLanguage = "WQL"
userFilter.Query = _
    "SELECT * FROM __InstanceOperationEvent" & _
    "WITHIN 5 WHERE TargetInstance ISA 'ds_user'"
userFilter.EventNamespace = "root\directory\LDAP"
userFilter.Put_()
```

С помощью данной последовательности операторов системе сообщается, события какого типа будут обрабатываться надстройкой. Главная строка – WQL-запрос, аналогичный тому, который применялся при регистрации временного подписчика на события в [1]. В этом случае мы следим за операциями над объектами класса `ds_user`. `__InstanceOperationEvent` – родительский класс для несколь-

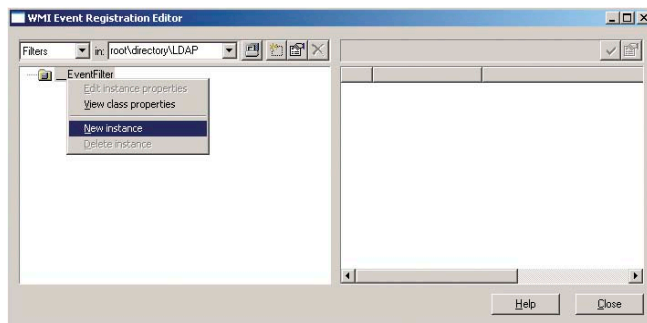


Рисунок 2. Создание фильтра

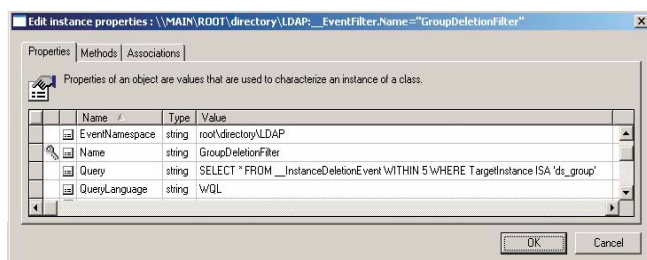


Рисунок 3. Настройка фильтра

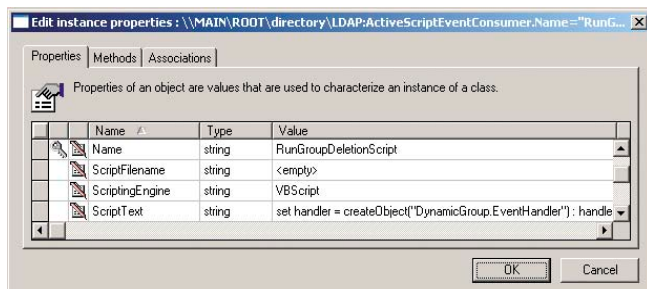


Рисунок 4. Настройка обработчика события

ких типов событий. Подобная регистрация позволит нам получать уведомления о событиях нескольких типов, из которых нужны два: создание экземпляра (__InstanceCreationEvent) и его удаление (__InstanceDeletionEvent).

Затем создается обработчик события (consumer). Отметим один важный аспект. Этому классу следует передать текст сценария в виде строки. Но строки в VBScript не могут содержать двойные кавычки и символы перехода на новую строку. Есть несколько общепринятых способов решения данной проблемы. Рассмотрим их на примере создания обработчика событий над объектами класса пользователь (user). Исходный код сценария требуется записать в свойство ScriptText класса ActiveScriptEventConsumer.

Во-первых, экземпляр этого класса можно создать посредством компиляции mof-файла. Именно такой способ использовался выше для того, чтобы поместить класс ActiveScriptEventConsumer в пространство имен root\directory\LDAP. Привожу содержание такого файла:

```
instance of ActiveScriptEventConsumer as $USERCONSUMER
{
    Name = " RunUserOperationsScript";
    ScriptingEngine = "VBScript";
    ScriptText =
        "set handler = createObject(\"DynamicGroup.EventHandler\")\n"
        "handler.dn = targetEvent.TargetInstance.ADSIPath\n"
        "handler.onUserEvent(targetEvent.path_.class) \n"
};
```

Итак, в mof-формате можно пользоваться с-подобным оформлением строк:

- переход на новую строку обозначается «\n»;
- двойная кавычка внутри строки экранируется с помощью обратного слеша.

Полученный файл компилируется и помещается в нужное пространство имен с помощью утилиты mofcomp, которая поставляется с операционной системой.

Часто применяется и другой вариант представления текста сценария в самой программе на VBScript. Рассмотрим пример создания аналогичного обработчика. Он основан на применении функции chr, которая возвращает символ по переданному ей коду. Именно таким образом в строку помещаются кавычки и переходы на новую строку. Двойная кавычка имеет код 34, переход на новую строку – 10. В примере указана только непосредственная инициализация свойства:

```
"set handler = createObject(" & chr(34) & _
    "DynamicGroup.EventHandler" & chr(34) & ") & chr(10) & _
    "handler.dn = targetEvent.TargetInstance.ADSIPath" & _
    chr(10) & "handler.onUserEvent(targetEvent.path_.class) "
```

Недостатки подходов очевидны – все они используют сценарий на языке VBScript вперемишу с посторонними символами. В результате даже программа из 3 строк становится практически нечитаемой. Еще сложнее ее модифицировать. Второй способ выглядит странно и даже несколько неправдоподобно, однако он используется, например, в [4].

Также оба метода имеют еще один, пусть и не такой существенный, недостаток. Часто свойство ScriptText обработчика нужно просмотреть или изменить в уже существующем объекте, например, с помощью утилит MS WMI Tools. Однако они отображают сценарий в виде одной строки, переходы же на новую строку заменены символом пустого прямоугольника (так обычно в программах отображаются символы, начертание которых неизвестно), что очень неудобно для прочтения. Кроме того, модификация в этом случае также сильно затруднена – новую строку добавить просто невозможно.

Однако язык VBScript содержит специальный элемент «:» (двоеточие), который служит для разделения операторов в одной строке. Таким образом, применяя этот разделитель, можно представить любой сценарий на VBScript в виде одной строки. Так и будет сделано в этой реализации. Сценарии непосредственной реакции на события разместим в отдельных файлах, чтобы было их проще менять. В результате нам нужно будет загрузить сценарий из файла и заменить переходы на новую строку символом двоеточия. Рассмотрим непосредственную реализацию:

```
set fso = createObject("Scripting.FileSystemObject")
Set srcFile = fso.OpenTextFile("ScriptText_user.vbs", _
    FOR_READING)
set userConsumer = consumerClass.SpawnInstance_(
    userConsumer.Name = "RunUserOperationsScript"
userConsumer.ScriptText = _
    multiLine2SingleString(srcFile.ReadAll)
userConsumer.ScriptingEngine = "VBScript"
userConsumer.Put_(
srcFile.close
```


- В качестве текста сценария добавляем команды, заменяя переходы на новые строки двоеточиями:

```
set handler = createObject("DynamicGroup.EventHandler")
handler.dn = targetEvent.TargetInstance.ADSIPath
handler.onGroupDelete()
```

Итак, последнее, что осталось сделать, — это создать экземпляр класса `__FilterToConsumerBinding`, который свяжет фильтр с обработчиком. После этого мы будем получать извещения об удалениях объектов групп. Эта операция выполняется утилитой неявно. Для этого требуется выделить обработчик события в левой части окна, нужный фильтр в правой и нажать кнопку «Register» (см. **рис. 5**).

После успешной регистрации слева от строки описания фильтра появится зеленая галочка. При желании можно подключить несколько фильтров к одному обработчику или использовать один фильтр для нескольких обработчиков. Но в текущей надстройке между этими классами используются только связи «один к одному».

Также рассмотрим способ настройки приложения WMI Event Viewer для отслеживания событий. Это очень удобный способ отладки приложений, использующих события операционной системы, поскольку утилита предоставляет возможность не только видеть факт возникновения события, но и просматривать его параметры.

Утилита использует для отслеживания событий собственный класс-обработчик — `EventViewerConsumer`. Его также нужно предварительно скомпилировать и поместить в наше пространство имен. В документации рекомендуется сначала отредактировать файл с классом (`evviewer.mof`), заменив в строчке:

```
#pragma namespace("\\root\\cimv2")
```

`root\\cimv2` на свое пространство имен (в нашем случае это `root\\directory\\LDAP`). После этого в утилите WMI Event Registration появляется обработчик `EventViewerConsumer`, который можно обычным образом связать с любым зарегистрированным фильтром. На **рис. 6** приведен пример отслеживания событий удаления группы и операций над объектами пользователей.

Теперь, скажем, при удалении группы будет не только вызван наш код обработки, но и появится информация о событии в утилите WMI Event Viewer (см. **рис. 7**).

Таким образом, использование этих утилит может существенно облегчить жизнь программистам, которым необходимо реализовать обработку событий операционной системы. Возможно, существуют и более совершенные коммерческие продукты, однако для большинства случаев и этого бесплатного набора утилит будет вполне достаточно.

Отключение надстройки

Ну и, наконец, рассмотрим способ отключения надстройки. Возможности неявного манипулирования составами групп при необходимости должны легко отключаться. Сначала рассмотрим процесс отмены регистрации обработчиков событий. Собственно для отключения извещения о событии достаточно удалить один класс из цепочки: «`__EventFilter`» —

«`__EventConsumer`» — «`__FilterToConsumerBinding`». Так и нужно делать при необходимости временного отключения реакции на события — в этом случае лучше удалять экземпляр класса `__FilterToConsumerBinding`, поскольку он не содержит никакой полезной информации, кроме связи фильтра с обработчиком. Тем не менее при полном удалении надстройки желательно удалить все классы, участвующие в обработке событий. Объекты классов `__EventFilter` и `__EventConsumer` имеют свойство «Имя (Name)». Такие объекты удаляются очень просто:

```
Set obj = GetObject("winmgmts:\\.\root\\directory \
LDAP:ActiveScriptEventConsumer= \
RunUserOperationsScript")
obj.Delete_
```

То есть нужно привязаться к объекту, указав имя компьютера (в данном примере точка используется для обозначения текущего сервера), пространство имен (`root\\directory\\LDAP`), тип объекта (`ActiveScriptEventConsumer`) и, наконец, его имя (`RunUserOperationsScript`). Затем объект удаляется из хранилища посредством вызова специального метода. Так можно удалять все именованные объекты.

По-другому ситуация обстоит с объектами типа `__FilterToConsumerBinding`. Они не имеют имени. Привязаться к ним можно с помощью специального запроса [5]. Запросы вида `references of {__EventFilter.Name='UserOperationsFilter'}` позволяют получить список всех объектов, ссылающихся на заданный (в этом случае — на `UserOperationsFilter`). Затем полученные объекты можно удалить стандартным образом (вызвав метод `delete_`):

```
Set objWIMService = GetObject("winmgmts:\\.\root \
directory\\LDAP")
Set objList = objWIMService.ExecQuery("references of \
{__EventFilter.Name='UserOperationsFilter'}")
For each objInst in objList
objInst.Delete_
Next
```

С помощью приведенной технологии должны быть удалены все объекты, помещенные в хранилище при регистрации обработчиков событий.

Далее рассмотрим функцию удаления дополнительных контекстных меню. Вся ее работа сводится к вызову функции `operateMenu`, реализованной в первой части статьи [10]:

```
const ADS_PROPERTY_DELETE = 4
function unInstallMenu(className, id, name, scriptPath)
operateMenu(ADS_PROPERTY_DELETE, className, id, name, \
scriptPath)
end function
```

Ей передаются те же самые параметры, что и функции добавления контекстных меню. Так что для удаления контекстных меню достаточно использовать код, аналогичный установке, заменив функцию `installMenu` на `unInstallMenu`.

Результаты

Перечислю основные сценарии, реализуемые надстройкой.

- **Добавление/удаление пользователя из организационной единицы.** Объект «пользователь» должен быть добавлен/удален из группы типа `Single`, прикрепленной

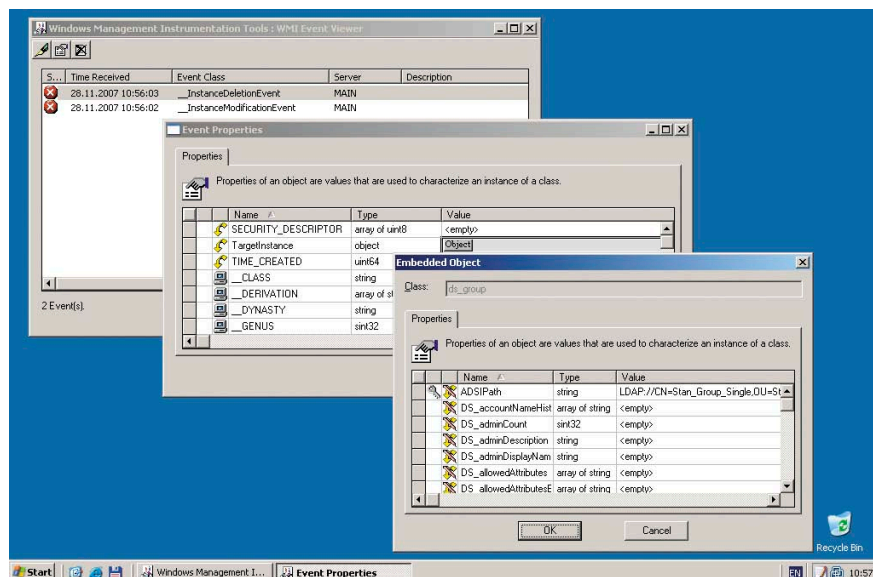


Рисунок 7. Просмотр параметров события удаления группы

к этому отделу. В случае отсутствия группы никаких действий не выполняется. Группы типа Tree на уровне отдельных пользователей не работают.

- **Добавление/удаление организационной единицы.** В прикрепленной группе типа Tree родительского подразделения регистрируется группа этого же типа добавляемой единицы. При удалении группа соответственно исключается.
- **Перенос объектов (пользователей и организационных единиц).** Перенос рассматривается как последовательные операции удаления и добавления объектов.
- **Прикрепление к организационной единице группы типа Tree.** В группу добавляются группы ти-

па двух ближайших дочерних организационных единиц и группа типа Single текущего подразделения.

- **Добавление/удаление подде-
рева.** Операции с поддеревом сводятся к действиям над его корневым элементом (следствие использования паттерна компоновщик [3] при организации иерархической структуры групп).

После развертывания надстрой-ки администратор получает возможность предоставлять доступ к ресурсам, опираясь исключительно на структурную модель организации. Так, чтобы открыть доступ к папке Documents поддереву с корнем в отделе бухгалтерии (Accounting) и руководству (Management), нужно сделать это для соответствующих прикрепленных групп. В дальнейшем при появлении новых сотрудников в указанных отделах, увольнениях, добавлении новых подразделений (в нашем примере – дочерних отделу «Бухгалтерия») достаточно будет изменить модель организации. В этих случаях корректировать списки доступа больше нет необходимости.

Итак, в данной статье описывается надстройка над стандартными механизмами разграничения доступа в сетях на основе операционных систем Windows 2003 Server на основе структуры организации. Данная серверная ОС позволяет создавать структурные модели организаций, однако не содержит законченных механизмов их пол-

ноценного использования, предоставляя вместо этого мощные механизмы расширения функциональности. В ходе проведенной работы были созданы инструменты, позволяющие согласованно использовать модель организации и списки контроля доступа при разграничении прав на корпоративные ресурсы. В результате основная часть повседневной работы администратора концентрируется на работе со структурной моделью организации (оснастка Active Directory Users and Computers). Отсутствие необходимости ручной поддержки дублирующих операций позволяет существенно повысить эффективность и качество администрирования сетей, в особенности крупных организаций со сложной структурой.

1. Андросов В. Реализуем нестандартные правила управления доступом на основе архитектуры организации в Windows Server 2003. //Системный администратор, №10, 2007 г. – С. 48-58.
2. Сайт Microsoft – www.microsoft.com.
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. «Приемы объектно-ориентированного проектирования. Паттерны проектирования». – СПб.: Питер, 2007. – 366 с.
4. Чарли Рассел, Шарон Кроуфорд, Джейсон Джеренд. «Windows server 2003 + SP1 и R2. Справочник администратора». – М.: Издательство «ЭКОМ», 2006. – 1424 с.
5. Alain Lissioir. «Understanding WMI Scripting». – Digital Press, 2003.
6. Microsoft Development Network – msdn.microsoft.com.
7. Чекмаев А.Н. «Windows 2000 Active Directory». – СПб.: БХВ-Петербург, 2001. – 400 с.
8. Don Jones, Jefferey Hicks «Advanced VBScript for Microsoft Windows Administrators». – Washington: Microsoft Press, 2006.
9. Microsoft® Windows® 2000 Scripting Guide «Enhanced WMI Monitoring Scripts».
10. Андросов В. Синхронизация ACL и структуры организации. Часть 1. // Системный администратор, №12, 2007 г. – С. 36-41.
11. Андросов В. Синхронизация ACL и структуры организации. Часть 2. //Системный администратор, №1, 2008 г. – С. 56-61.

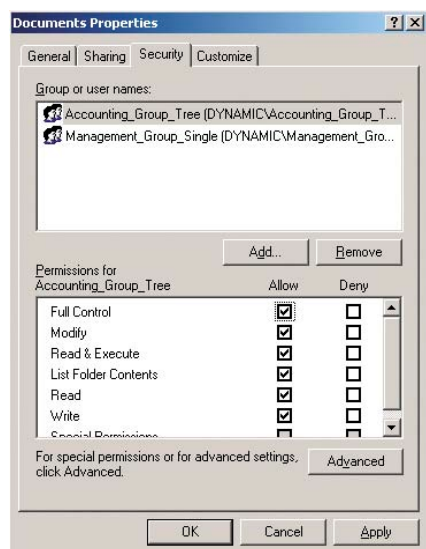


Рисунок 8. Использование надстройки