**Active Directory for Application Mode**

Сергей Зернюков

**Введение**

**Что такое ADAM и для чего он нужен?**

Active Directory for Application Mode (ADAM) представляет собой локальную версию Active Directory (далее по тексту – AD), устанавливаемую в виде службы NT. ADAM предназначен для хранения данных, имеющих иерархическую структуру. Типичными примерами этого могут служить: глобальные адресные книги и системы работы с клиентами или персоналом. Конечно, иерархически организованные данные можно хранить и в XML-файле. Но это сомнительно с точки зрения безопасности, тем более, если хранимая информация является конфиденциальной. ADAM же использует встроенные механизмы безопасности Windows, что в большинстве случаев обеспечивает надежную защиту и прозрачный контроль доступа к данным. Самым удачным вариантом применения ADAM может быть хранение в Active Directory дополнительной информации об объектах.

**Установка и начальная настройка**

ADAM может быть установлен на операционной системе Microsoft Windows 2003 Server или Microsoft XP. Подробно процесс установки описан в руководстве, поставляемом вместе с установочным пакетом. Здесь хотелось бы остановиться только на двух моментах. Первый касается разделов (partitions). Раздел в ADAM-е является контейнером верхнего уровня, в котором может храниться иерархия объектов. В процессе установки создаются два системных раздела: один для хранения схемы, другой для хранения конфигурации. Кроме того, пользователю предлагается создать раздел, который будет использоваться приложением. Если вы точно знаете, как будет называться раздел данных приложения, введите имя раздела, и он будет создан во время установки. Полное имя, или, в терминах AD – distinguished name, имеет следующий вид: ‘OU=MyApp,O=MyCompany,C=RU’. Подробнее разделы будут рассмотрены далее в статье.

Так же пользователю предлагается импортировать LDF-скрипты, расширяющие схему такими объектами как User и User-Proxy. Если вы собираетесь использовать объекты этих типов, вы можете создать эти классы в процессе установки. Если же изначально использование таких объектов не планируется, то импортировать ничего не нужно. В любом случае, это можно будет сделать и после установки.

**Средства администрирования**

Средства администрирования являются, на мой взгляд, одним из самых слабых мест ADAM – они представляют собой обширный набор различных, порой не очень удобных в использовании утилит (см. таблицу 1).

|  |  |
| --- | --- |
| Утилита | Назначение |
| ADAM ADSI Edit | MMC-приложение, позволяющее редактировать иерархию объектов и производить общую настройку работы ADAM. |
| ADAM Schema | MMC-приложение для работы со схемой. |
| Csvde.exe | Утилита командной строки для импорта и экспорта данных и обновления схемы с помощью текстовых comma-separated файлов. |
| Dsmgmt.exe | Утилита командной строки для общего администрирования. |
| Dsacls.exe | Утилита командной строки для просмотра и назначения прав доступа к объектам иерархии. |
| Dsdbutil.exe | Утилита командной строки для управления файлами базы данных (.dit) ADAM. |
| Dssiag.exe | Утилита командной строки для мониторинга ADAM. |
| Ldifde.exe | Утилита командной строки для импорта и экспорта данных и обновления схемы с помощью файлов в формате LDAP Data Interchange Format (LDIF). |
| Ldp.exe | Утилита для администрирования ADAM. |
| Repadmin.exe | Утилита командной строки для управления репликацией. |

Таблица 1. Утилиты администрирования ADAM.

Из этого списка видно, что для администрирования в основном применяются утилиты командной строки, что никак нельзя назвать удобным. Кроме того, поскольку, в большинстве случаев, сопровождением приложений занимаются люди, далекие от администрирования операционной системы – было бы не плохо иметь более удобные способы работы с ADAM.

Почти все представленные утилиты находятся в каталоге, в который был установлен ADAM, чаще всего это C:\WINDOWS\ADAM. Исключение составляет ADAM Schema. Для подключения этой утилиты необходимо запустить Microsoft Management Console (MMC), в меню File добавить snap-in “ADAM Schema”, а затем сохранить файл .MMC.

**Работа с ADAM**

**Понятие раздела**

Важным понятием в ADAM является раздел (partition). Раздел является контейнером верхнего уровня, в котором может храниться иерархия объектов. Конечно, приложение может использовать и несколько разделов для хранения данных, однако осуществлять поиск и устанавливать связи между объектами можно только внутри одного раздела.

В процессе установки создаются два системных раздела, используемых для хранения конфигурации и схемы. Создать новый раздел можно с помощью утилиты LDP.EXE. Для этого надо запустить программу, подключиться к требуемой службе ADAM, установить соединение (bind) и выполнить несколько манипуляций, а именно:

В меню Browse выбрать пункт Add child.

В поле Dn ввести полное имя (distinguished name) нового раздела.

В поле Attribute ввести ObjectClass, в поле Values - container и нажать Enter.

Затем в поле Attribute ввести InstanceType, в поле Values – 5 и нажать Enter.

Нажать Run.

Рисунок 1. Утилита LDP.EXE - создание нового раздела.

Новый раздел будет создан, а вы получите сообщение об успешно выполненной операции.

**Работа с данными в ADAM**

После того как раздел создан, можно приступать к заполнению его данными, необходимыми для работы приложения. Единицей данных в ADAM является объект. Каждый объект имеет набор атрибутов, определяемых классом объекта. Некоторые из атрибутов являются общими для всех классов – атрибуты, унаследованные от класса top. Наиболее важные из них:

name – короткое имя объекта (MyObject1);

distinguishedName – полное имя объекта, уникальное в пределах сервера значение, идентифицирующее объект (CN= MyObject1,OU=MyApp,O=MyCompany,C=RU, см. ниже);

instanceType – системное значение, определяющее поведение объекта. В частности, этот атрибут указывает, может ли объект быть изменен вручную, или он является репликой объекта на сервере, с которым данный сервер синхронизируется.

objectClass – класс объекта, определяет набор атрибутов и поведение объекта.

distinguishedName содержит имя объекта, а так же имена всех “контейнерных” объектов – каталогов, которые его содержат. Частью distinguishedName является полное имя раздела. Имя каждого объекта в полном имени предваряется префиксом, указывающим на тип объекта. Вот расшифровка некоторых из этих аббревиатур:

CN – common name,

OU – organization unit,

O – organization,

DC – domain component,

C – country/region.

Основным средством создания, модификации и удаления объектов в ADAM является “ADAM ADSI Edit”.

Рисунок 2. Внешний вид утилиты "ADAM ADSI Edit".

После подключения к нужному разделу нужного сервера можно добавлять собственные объекты в иерархию. В диалоговом окне создания нового объекта вам будет предложено выбрать класс, от которого будет унаследован объект. Объект определенного класса может содержать вложенные объекты классов, для которых он (класс) прописан в схеме как возможный “родитель”.

Расширение схемы

Описанное выше было бы совершенно бесполезно, если бы нельзя было расширять схему ADAM, то есть добавлять собственные классы и атрибуты. К счастью, Microsoft позаботился об этом, и в нашем распоряжении есть утилита “ADAM Schema”.

Рисунок 3. Внешний вид утилиты "ADAM Schema".

Схема – формальное описание содержимого и структуры службы каталогов – представляет собой набор связанных между собой классов, которые в свою очередь состоят из набора атрибутов.

Ниже приведены некоторые термины, необходимые для понимания сути схемы и принципов ее обновления.

1. Атрибут – поле данных, характеризующее какое-либо свойство объекта. Объект может содержать только атрибуты, прописанные в схеме для его класса.

2. Класс – формальное описание типа объекта в службе каталогов.

3. Объект – единица данных в службе каталогов.

4. Идентификатор объекта (OID) – уникальное численное значение, однозначно идентифицирующее элемент данных, синтаксиса и прочие части распределенных приложений. Подобные идентификаторы можно обнаружить в различных приложениях и системах, для которых важна глобальная уникальность элемента системы. Примеры таких приложений: приложения OSI, служба каталогов X.500 и SNMP. Идентификаторы имеют древовидную структуру, в которой главный источник, такой как ISO, выделяет интервал/поддерево для использования младшими источниками идентификаторов, которые, в свою очередь, выделяют поддеревья для дочерних источников. Рассмотрим пример из MSDN. OID = "1.2.840.113556.1.5.4". Здесь:

1 – ISO, корневой источник, выделил “1.2” для ANSI;

2 – ANSI, выделил “1.2.840” для США;

840 – США, выделил “1.2.840.113556” для Microsoft;

113556 – Microsoft;

1 – Microsoft DS;

5 – классы NTDS;

4 – класс Builtin-Domain.

5. X.500 – система стандартов, разработанная совместно ISO и ITU (известна как CCITT), которая определяет способы именования, представления данных и протоколы связи для служб каталогов.

6. Подкласс – класс, который наследуется от другого класса.

7. Суперкласс – класс, от которого унаследован один или более подклассов.

Схема описывает все атрибуты и классы. Для каждого класса должны быть определены следующие атрибуты:

Poss-superiors – описывает структуру дерева каталогов, определяя возможных “родителей” данного класса, то есть классы, в объекты которых может быть вложен объект данного класса.

Must-Contain – список атрибутов, обязательных для заполнения.

May-Contain – список атрибутов, доступных для заполнения.

Все классы непосредственно или опосредованно через свои суперклассы наследуются от абстрактного класса top.

|  |
| --- |
| ПРЕДУПРЕЖДЕНИЕЗаметьте, что единожды созданный класс или атрибут невозможно удалить из схемы в дальнейшем! Microsoft объясняет такое поведение тем, что не может гарантировать отсутствия коллизий, если удаленный вроде бы класс или атрибут все еще будет использоваться в других частях распределенной системы. По той же причине невозможно удалить из класса обязательный атрибут. Набор же необязательных атрибутов можно изменять в любое время. Помните об этом во время проектирования схемы.  |

Добавление нового класса в схему сводится к выбору суперкласса, вводу имени нового класса, его OID-а X.500 и заполнению всех описанных выше обязательных атрибутов. Создать класс можно с помощью мастера создания нового класса утилиты “ADAM Schema”.

При добавлении атрибута необходимо ввести название нового атрибута, OID X.500, выбрать синтаксис или тип атрибута, задать интервал возможных значений и указать – является ли атрибут набором (списком) значений. Создать атрибут, как и класс, можно с помощью предусмотренного мастера создания нового атрибута в утилите “ADAM Schema”.

Рисунок 4. Диалоговое окно создания нового класса.

Управление пользователями и группами пользователей

Чтобы иметь возможность создавать учетные записи собственных пользователей ADAM, необходимо либо на этапе установки, либо позже с помощью утилиты Ldifde.exe, импортировать класс User, который определен в LDF-скрипте MS-User.LDF.

Для создания пользователя достаточно в “ADAM ADSI Edit” добавить новый объект класса User. После того, как новый пользователь добавлен, необходимо задать для него пароль. Утилита “ADAM ADSI Edit” позволяет сделать это из контекстного меню пользователя (пункт “Reset Password”). Поведение объекта класса User настраивается с помощью набора атрибутов. Некоторые из них:

msDS-UserAccountDisabled – если TRUE, учетная запись пользователя не активирована. Сразу после создания значение этого атрибута устанавливается в TRUE. Соответственно, для активации учетной записи нового пользователя необходимо вручную изменить значение на FALSE.

msDS-UserDontExpirePassword – если TRUE, указывает системе, что пароль данного пользователя имеет неограниченный срок действия.

msDS-UserPasswordExpired – значение TRUE данного атрибута говорит о том, что время действия текущего пароля закончилось, и требуется его смена.

Для управления правами доступа удобнее всего назначать права не напрямую пользователям, а группам пользователей. Такой алгоритм обеспечивает наиболее гибкое администрирование доступа к объектам службы каталогов. Здесь группы выступают в качестве ролей, которые определяют доступ к тем или иным частям дерева каталогов. Соответственно пользователь наследует права доступа у группы или нескольких групп, членом которых он является.

Чтобы создать группу пользователей, создается новый объект класса group. При создании надо не забыть установить значение атрибута groupType равным 2147483650 (0x80000002 в шестнадцатеричном представлении). Это значение указывает, что созданная группа является группой пользователей. За связывание пользователей с группой отвечает атрибут member класса group. “ADAM ADSI Edit” позволяет добавлять в группу как доменных пользователей, так и собственных пользователей ADAM.

Авторизация и управление правами доступа. Утилита DSACLS.EXE

Авторизация – это процесс определения, прав пользователя на доступ к объекту службы каталогов. Определение прав осуществляется на основе так называемых списков контроля доступа – Access Control List (ACLs). Утилита командной строки DSACLS.EXE позволяет просматривать и изменять права доступа пользователей и групп пользователей.

Полный список команд и ключей этой утилиты можно получить, запустив ее с ключом /?. Приведу два наиболее типичных случая применения.

Просмотр списка контроля доступа к объекту с полным именем ‘CN=MyObject1,OU=MyApp,O=MyCompany,C=RU’:

|  |
| --- |
| dsacls.exe "\\localhost:389\ CN= MyObject1,OU=MyApp,O=MyCompany,C=RU" |

Установка прав на чтение объекта ‘CN= MyObject1,OU=MyApp,O=MyCompany,C=RU’ и объектов его поддерева группе пользователей ‘CN=USER\_GROUP,OU=MyApp,O=MyCompany,C=RU’:

|  |
| --- |
| dsacls.exe "\\localhost:389\ CN= MyObject1,OU=MyApp,O=MyCompany,C=RU" /I:T /G "CN=USER\_GROUP,OU=MyApp,O=MyCompany,C=RU":GR |

Способы идентификации пользователей.

Подсоединиться к ADAM можно тремя разными способами: как пользователь Windows, как пользователь ADAM (класс User) или с помощью объекта класса UserProxy.

Во время установки ADAM запрашивает имя пользователя, который в дальнейшем будет администратором ADAM. По умолчанию это пользователь, устанавливающий ADAM. Все администраторы ADAM входят в группу пользователей Administrators, расположенную в разделе конфигурации; полное имя этой группы – CN=Administrators, CN=Roles, CN=Configuration, CN={идентификатор раздела конфигурации}. Таким образом, вы можете назначить свою учетную запись Windows администратором ADAM и подключаться в дальнейшем к ADAM как пользователь Windows. Точно так же можно создать свою группу в разделе приложения и добавить пользователей Windows в эту группу.

Если вы создали пользователя ADAM и назначили ему права доступа к объектам раздела приложения, вы можете подключаться к ADAM-у от имени этого пользователя. Именем пользователя в этом случае будет полное имя (Distinguished Name) пользователя.

Кроме описанных выше двух способов, ADAM позволяет подключаться, используя так называемое перенаправление подключения, которое реализуется классом UserProxy. Объект класса UserProxy хранит в себе ссылку на пользователя Windows в AD. Такой режим может быть удобен, когда приложению необходима дополнительная информация (та, которой нет в Active Directory), связанная с пользователем Windows.

Пакетная обработка. Утилита LDIFDE.EXE

Все вышеописанные инструменты обеспечивают достаточно средств для администрирования ADAM. Однако все изменения, такие как добавление новых классов, объектов и пользователей, должны выполняться вручную. Это не очень удобно, особенно если есть необходимость повторять точно те же действия на другом сервере. Типичный пример – разработка ведется на компьютере разработчика, тестирование на специальном сервере тестирования, а работает приложение на рабочем сервере. В этом случае было бы удобно иметь возможность выполнять все изменения в пакетном режиме. Такую возможность предоставляет утилита LDIFDE.EXE, которая позволяет экспортировать данные из ADAM в LDF-скрипты и импортировать LDF-скрипты в ADAM. Рассмотрим типичные случаи применения этой утилиты, возникающие при разработке программного обеспечения.

Экспорт всего поддерева CN=Personnel,OU=MyApp,O=MyCompany,C=RU в виде, удобном для дальнейшего импорта на другой сервер:

|  |
| --- |
| C:\WINDOWS\ADAM\ldifde.exe -f data.ldf -s localhost:389 -d " CN=Personnel,OU=MyApp,O=MyCompany,C=RU " -j . -p subtree -o "cn,distinguishedName,instanceType,whenCreated,whenChanged,uSNCreated,uSNChanged,objectGUID,objectCategory,dSCorePropagationData" |

Импорт данных в ADAM:

|  |
| --- |
| ldifde -i -f data.ldf -s localhost:389 -j . –k |

Примеры команд LDF-скриптов.

Добавление группы пользователей:

|  |
| --- |
| dn: CN=USER\_GROUP, OU=MyApp,O=MyCompany,C=RU changetype: addobjectClass: topobjectClass: groupcn: AdministratorsinstanceType: 4groupType: 2147483650 |

Удаление объекта:

|  |
| --- |
| dn: CN=USER\_GROUP, OU=MyApp,O=MyCompany,C=RU changetype: remove |

Изменение объекта:

|  |
| --- |
| dn: CN=Object1, OU=MyApp,O=MyCompany,C=RU changetype: modifyreplace:ColorColor: RED- |

Переименование объекта:

|  |
| --- |
| dn: CN=Object1, OU=MyApp,O=MyCompany,C=RU changetype: modrdnnewrdn: CN=RenamedObject1deleteoldrdn: 1 |

Для расширения схемы ADAM-а нужно использовать командную строку с указанием пространства имен схемы:

|  |
| --- |
| C:\WINDOWS\ADAM\LDIFDE.EXE -i -f SchemaExt.ldf -s localhost:389 -j . -c "CN=Schema,CN=Configuration,DC=X" #schemaNamingContext -k |
| ПРЕДУПРЕЖДЕНИЕЗаметьте, что если для обновления данных раздела, можно использовать утилиту LDIFDE.EXE, идущую в поставке WINDOWS, то при изменении схемы ADAM необходимо использовать утилиту, установленную вместе с ADAM, иначе команда вернет ошибку. |

Содержимое файла SchemaExt.ldf, используемого в примере:

|  |
| --- |
| #Добавление в схему нового атрибута:dn: CN=MyAttribute1,CN=Schema,CN=Configuration,DC=Xchangetype: ntdsschemaaddobjectClass: topobjectClass: attributeSchemaattributeID: 1.3.444.656565.1.1attributeSyntax: 2.5.5.12isSingleValued: TRUEshowInAdvancedViewOnly: TRUEadminDisplayName: MyAttribute1oMSyntax: 64lDAPDisplayName: MyAttribute1 #Добавление в схему нового класса:dn: CN=MyClass1,CN=Schema,CN=Configuration,DC=Xchangetype: ntdsschemaaddobjectClass: topobjectClass: classSchemapossSuperiors: containersubClassOf: topgovernsID: 1.3.444.656565.2.1mayContain: MyAttribute1showInAdvancedViewOnly: TRUEadminDisplayName: MyClass1adminDescription: New classlDAPDisplayName: MyClass1 dn:changetype: modifyadd: schemaUpdateNowschemaUpdateNow: 1- |

В этом примере первая команда создает новый атрибут MyAttribute1, а вторая – новый класс MyClass1, использующий этот атрибут. Последняя команда фиксирует изменения в схеме. Такая команда должна вызываться каждый раз после создания нового класса.

Способы взаимодействия приложения с ADAM

Для работы с ADAM используются те же средства, что и для работы с AD. Это – Active Directory Services Interfaces (ADSI), Lightweight Directory Access Protocol (LDAP), пространство имен System.DirectoryServices Microsoft .NET Framework, а также провайдер доступа к данным OLEDB ADsDSOObject.

ADSI предоставляет набор COM-интерфейсов для работы с ADAM и обеспечивает наиболее широкий набор инструментов.

LDAP – “облегченный” протокол доступа, позволяющий изменять содержимое службы каталогов и выполнять поиск по дереву объектов. LDAP может быть использован в приложениях, для которых работа с ADAM-ом не является основной функциональностью.

Пространство имен System.DirectoryServices (SDS) разработано на базе ADSI и предоставляет возможности, схожие с теми, что предоставляет протокол LDAP. SDS можно использовать в приложениях на базе технологии Microsoft.NET.

Провайдер доступа к данным ADsDSOObject предоставляет SQL-подобный интерфейс чтения, изменения и поиска данных в AD. Из всех описанных средств это обладает самой низкой производительностью, но оно может быть использовано в приложениях, выполняющих поиск в ADAM, если скорость работы не является критичной.

Работа с объектами: чтение, изменение и поиск данных

Рассмотрим работу с ADAM на примере, использующем пространство имен System.DirectoryServices.

В примере производится поиск, и в найденных объектах изменяется значение свойства MyAttribute1. Проверка существования значения атрибута не выполняется – предполагается, что оно существует. Также не выполняется обработка исключений. Сделано это для того, чтобы не загромождать код примера.

|  |
| --- |
| using System;using System.DirectoryServices;namespace SDSExample{ class SDSExample { [STAThread] static void Main(string[] args) { //Подключение к ADAM-у DirectoryEntry deRoot =  new DirectoryEntry( "LDAP://LOCALHOST:389/CN=Personnel,OU=MyApp,O=MyCompany,C=RU");  //Создание объекта класса DirectorySearcher, выполняющего поиск DirectorySearcher dSeacher =  new DirectorySearcher(deRoot, "(objectClass=MyClass1)"); //Указание искать по всему поддереву корневого объекта dSeacher.SearchScope = SearchScope.Subtree; //Поиск всех объектов, удовлетворяющих условию SearchResultCollection srEntries = dSeacher.FindAll(); for(int index=0; index<srEntries.Count; index++)  { //Вывод значений атирбутов объекта SearchResult srEnrty = srEntries[index]; Console.WriteLine(index); Console.WriteLine("distinguishedName ="  + srEnrty.Properties["distinguishedName"]); Console.WriteLine("MyAttribute1 ="  + srEnrty.Properties["MyAttribute1"][0]); Console.WriteLine(); //Изменение значения атрибута MyAttribute1 DirectoryEntry deEntry = srEnrty.GetDirectoryEntry(); deEntry.Properties["MyAttribute1"].Value =  "new\_value\_of\_Object" + index.ToString(); //Фиксирование изменений deEntry.CommitChanges(); deEntry.Close(); } deRoot.Close(); }//Main() }//class SDSExample}//namespace SDSExample |

Работа с пользователями

Для некоторых приложений может потребоваться возможность управления пользователями ADAM (не Windows). Работу с пользователями из приложения демонстрирует следующий пример. В нем создается новый пользователь с именем ‘CN=Vasiliy\_Pupkin,CN=Users,OU=MyApp,O=MyCompany,C=RU’ и добавляется в группу ‘CN=USER\_GROUP,OU=MyApp,O=MyCompany,C=RU’.

|  |
| --- |
| using System;using System.DirectoryServices;namespace ADAMUserExample{ class ADAMUser { [STAThread] static void Main(string[] args) { DirectoryEntry deUserRoot =  new DirectoryEntry( "LDAP://LOCALHOST:389/CN=Users,OU=MyApp,O=MyCompany,C=RU"); //Создание пользователя DirectoryEntry deNewUser = deUserRoot.Children.Add( "CN=Vasiliy\_Pupkin", "User"); deNewUser.Properties["displayName"].Value = "Vasiliy Pupkin"; deNewUser.CommitChanges(); //назначение пароля deNewUser.Properties["userpassword"].Value = "Vasiliy\_Password\_1"; deNewUser.CommitChanges(); //активирование пользователя deNewUser.Properties["msds-useraccountdisabled"].Value = false; //отмена срока действия пароля deNewUser.Properties["msds-userdontexpirepassword"].Value = true; deNewUser.CommitChanges(); DirectoryEntry deUserGroup =  new DirectoryEntry( "LDAP:// LOCALHOST:389/CN=USER\_GROUP,OU=MyApp,O=MyCompany,C=RU"); deUserGroup.Properties["member"].Add( deNewUser.Properties["distinguishedName"].Value); deUserGroup.CommitChanges(); deUserGroup.Close(); deNewUser.Close(); deUserRoot.Close(); }//Main }//class ADAMUser} |

Однако если попытаться выполнить данный пример, используя конфигурацию ADAM по умолчанию, возникнет ошибка в момент применения пароля пользователя. Это произойдет потому, что конфигурация по умолчанию запрещает смену пароля с использованием незащищенного канала (в примере используется порт 389 – порт незащищенного канала, предлагаемый при установке ADAM). Возможны два варианта решения проблемы. Первый – настроить защищенный канал (порт защищенного канала, предлагаемый при установке, 636, использует SSL для шифрования канала) и подключаться к ADAM через него. Второй – настроить ADAM так, чтобы изменение пароля через незащищенное соединение было разрешено. Для этого необходимо установить тринадцатый символ атрибута dSHeuristics объекта конфигурации с полным именем ‘CN=Directory Service,CN=Windows NT,CN=Services,CN=Configuration,CN={идентификатор раздела конфигурации ADAM-а}’ в ‘1’. По умолчанию значение этого атрибута в ADAM-е не установлено.

LDF-скрипт изменяющий соответствующим образом конфигурацию ADAM:

|  |
| --- |
| dn: CN=Directory Service,CN=Windows NT,CN=Services,CN=Configuration,DC=Xchangetype: modifyreplace: dSHeuristicsdSHeuristics : 0000000001001- |

Выполнить этот скрипт можно с помощью утилиты LDIFDE.EXE, как показано ниже.

|  |
| --- |
| c:\windows\adam\ldifde.exe -i -f config.ldf -s localhost.ldf -j . -c "CN=Configuration,DC=X" #configurationNamingContext -k |

Особенности использования ADAM в многопользовательских распределенных приложениях

Работа с ADAM из многопользовательских приложений имеет один нюанс, который необходимо учитывать на этапе проектирования системы. Нюанс этот касается возможности идентификации (authentication) пользователя ADAM. По какой-то причине подключение к ADAM, как и к AD, возможно только при наличии первичного контекста безопасности (primary security token) в процессе, обращающемся к ADAM. Рассмотрим два случая: первый – когда приложение работает от имени пользователя, информация о котором хранится в ADAM, и второй – когда приложение работает от имени текущего пользователя Windows. В первом случае идентификатор пользователя и пароль вводятся при запуске приложения. Обладая этой информацией, приложение может подключиться к ADAM под первичным контекстом безопасности в любой свой части (очень важно для распределенных приложений). Сложности возникают во втором случае. Заключаются они в том, что получить первичный контекст безопасности можно только на том компьютере, где запущено клиентское приложение, с которым работает пользователь. Этим приложением может быть как Windows-клиент, работающий с сервером приложений, так и Internet Explorer в случае Web-приложений. В обоих случаях приложению неизвестен пароль пользователя. Из-за этого возникает необходимость в передаче контекста безопасности с клиента на сервер. Сделать это можно с помощью функций WinAPI InitializeSecurityContext и AcceptSecurityContext, которые позволяют зашифровать данные о контексте безопасности пользователя, передать их в процесс сервера (возможно, на другом компьютере) и восстановить контекст безопасности в процессе сервера. В этом случае для передачи контекста безопасности необходимо использовать механизм идентификации Kerberos, что не всегда возможно из-за сложности настройки и прочих причин, таких, как соединение клиента и сервера через Proxy-сервер. Использовать именно Kerberos нужно потому, что этот механизм, в отличие от NTLM и Digest, позволяет передавать по сети первичный идентификатор безопасности.

Рассмотрим для примера ASP.NET-приложение. Здесь существует возможность использовать интегрированную систему безопасности Windows для идентификации пользователя на сайте. После идентификации серверный код можно запустить под опознанным пользователем – так называемое имитирование (impersonation) контекста безопасности пользователя. Однако добиться таким образом желаемого результата – подключения к ADAM под пользователем приложения – не удастся. Подключение к ADAM из-под сымитированного контекста безопасности пользователя не возможно – необходим первичный контекст безопасности. В англоязычной литературе данная проблема носит название double-hop issue, что можно перевести как проблема двойного скачка. Она возникает всякий раз, когда необходимо передать контекст безопасности от одного процесса другому через процесс-посредник. В случае с ASP.NET-приложением процессом-источником является Internet Explorer, процессом-приемником – ADAM-а, а посредником является процесс ASP.NET.

Таким образом, при проектировании архитектуры приложения, в котором предполагается использовать ADAM, необходимо на раннем этапе позаботиться о способах подключения к ADAM и методах идентификации пользователей в случае многопользовательского приложения.

**Заключение**

Из всего выше сказанного можно сделать вывод, что ADAM является мощным инструментом, позволяющим использовать в приложении возможности Active Directory. Отчасти это так – ADAM позволяет расширять схему, создавать объекты этих классов, организовывать их в иерархии, а также выполнять поиск этих объектов. Гибкая система управления пользователями и контролем доступа предоставляет большие возможности для использования ADAM в приложениях, оперирующих конфиденциальной информацией. Механизмы архивирования и репликации позволяют создавать надежные отказоустойчивые системы.

Слабая эргономичность средств администрирования и некоторые трудности применения в многопользовательских приложениях несколько портят картину. Однако при грамотном проектировании использование ADAM может стать отличным выбором для широкого спектра программных систем.