Microsoft

Zestaw narzędzi do administracji Windows:

Efektywne rozwiązania dla specjalistów IT

Resource Kit

Dan Holme

Zestaw narzędzi do administracji Windows: efektywne rozwiązania dla specjalistów IT
Resource Kit
Edycja polska Microsoft Press

Tytuł oryginału: Windows® Administration Resource Kit: Productivity Solutions for
IT Professionals

Polish edition by APN PROMISE Sp. z o.o. Warszawa 2008

APN PROMISE Sp. z o.o., biuro: 00-108 Warszawa, ul. Zielna 39
tel. (022) 355-16-00; fax (022) 355-16-99
e-mail: mspress@promise.pl

ISBN: 978-83-7541-026-6

Przekład: Małgorzata Dąbkowska-Kowalik, Witold Sikorski, Marzenna Latawiec
Redakcja: Marek Włodarz
Korekta: Ewa Swędrowska
Skład i łamanie: MAWart Marek Włodarz

Dan Holme

Ta książka jest dedykowana moim niezwykłym klientom,
dzięki którym zyskałem ponad dziesięcioletnią naukę,
okres doświadczeń i mnóstwo wiedzy.
Ta książka jest dla nich, a także od nich – za moim pośrednictwem –
dla społeczności administratorów Windows.

O autorze

Dan Holme jest absolwentem Uniwersytetu w Yale i Thunderbird School of Global Management. Spędził ponad dziesięć lat będąc konsultantem i szkoleniowcem, dostarczając rozwiązania dziesiątkom tysięcy specjalistów IT z najbardziej renomowanych organizacji i korporacji na całym świecie. Firma Dana, Intelliem, specjalizuje się w zwiększaniu wydajności specjalistów IT oraz użytkowników poprzez tworzenie zaawansowanych, niestandardowych rozwiązań oraz włączenie określonych projektów i konfiguracji klienta do służących poprawie wydajności narzędzi, szkoleń i usług zarządzania wiedzą.

Dan jest również redaktorem współpracującym magazynu *Windows IT Pro*, Microsoft MVP (Microsoft Office SharePoint Server) i prowadzącym w społeczności OfficeSharePointPro.com. Ze swojej bazy na pięknej wyspie Maui[1] Dan podróżuje po całym świecie, wspierając klientów i szkoląc ich z technologii Windows. Zaraz po wydaniu tego zestawu narzędzi będzie się przygotowywać do roli konsultanta Windows Technologies dla telewizji NBC na Igrzyskach Olimpijskich w Pekinie; taką samą funkcję pełnił również w Turynie w 2006 r.

[1] Wyspa na Hawajach (przyp. tłum.)

Podziękowania

Historia książki, którą czytelnicy mają w rękach, jest długa i kształtowało ją wielu wspaniałych ludzi, którzy pomagali mi w trakcie jej tworzenia.

Po pierwsze są wśród nich moi klienci – harujący jak niewolnicy, szaleńcy i budzący respekt oraz podziw przyjaciele, którzy powierzyli mi rolę przewodnika dla czytelników oraz ich przedsiębiorstw, i dzielili się ze mną swoją wiedzą i doświadczeniem. Bez nich nie byłoby wiedzy, rozwiązań i doświadczenia, z których powstał ten zestaw narzędzi. Dziękuję, że dzięki Wam moja kariera stała się nieustanną nauką. Dziękuję za wasze działania i zaufanie dla mnie. Dziękuję za danie mi wielu okazji w życiu.

Dalej są moi koledzy – to wy chłopcy i dziewczęta, super zwariowani, super guru, porwaliście mnie swoimi mózgami i krzepą. Jeremy, Don, Darren, Mark, Rhonda, Derek, Alan, Gil, Sean, Guido, Jim, Brian, Steve, Richard, Joel, Tom. Przepraszam, jeśli kogoś pominąłem... Dziękuję za ustawienie poprzeczki tak wysoko i zachęcenie mnie do jej dosięgnięcia!

Następnie są niesamowici ludzie z Microsoft Press. Zaczynając od Martina Del Re. Zobaczył mnie w roku 2003, gdy przedstawiałem informacje oparte na rozwiązaniach i powiedział: „Pewnego dnia będziesz musiał to spisać" i stał cały czas u mego boku. Dokonaliśmy tego! Karen Szall, w tym projekcie wykorzystałem cały swój kredyt i jestem Ci winien następny raz! Melissa von Tschudi-Sutton, weszłaś do tego dużego pociągu, nie pracując ze mną wcześniej, a z wdziękiem, w ciągu tylko 10 tygodni wyciągnęłaś ponad 650 stron treści i dziesiątki skryptów. . Nigdy dość podziękowań dla Ciebie, Melisso! I, oczywiście, Curtis Philips, Rozanne Whalen, Roger LeBlanc i Teresa Barensfeld – wzięliście się do tego nowego typu zestawu narzędzi z niezwykłymi umiejętnościami. Ten projekt był gigantyczny i mógłby nie powstać bez każdego z Was. Jestem szczęśliwy, że z Wami pracowałem!

Na koniec najważniejsze podziękowania, dla moich przyjaciół i rodziny: Lyman, Maddie, Mom i Dad, Bob i Joni, Stan i Marylyn, Julie, Joe i całej paczki w Maui i Phoenix. Wasza cierpliwość, wparcie oraz miłość dodawały mi sił. Dziękuję za dodawanie mi otuchy, wspieranie mnie i czekanie na mnie na linii mety. Jestem Wam winien dużo czasu po zakończeniu projektu. Nauczyliście mnie znaczenia słowa ohana! Mahalo![2]

[2] W kulturze hawajskiej *ohana* oznacza rodzinę w znaczeniu szerszym niż tylko więzy krwi, a *mahalo* znaczy „Dziękuję!" (przyp. tłum.)

Spis Treści

Wstęp

Witamy w *Windows Administration Resource Kit: Productivity Solutions for IT Professionals*!

Windows Administration Resource Kit jest skarbnicą wiedzy zaprojektowaną, aby pomóc Czytelnikom w stosowaniu technologii Microsoft Windows do rozwiązywania rzeczywistych problemów biznesowych. Podczas gdy większość zestawów narzędzi jest skoncentrowana na indywidualnym produkcie lub zagłębia się we wnętrze technologii, ten zestaw narzędzi przyjmuje całkiem inny punkt widzenia. Przyglądamy się w nim, jak dostosować technologie Windows do celów naszej działalności.

Jeśli to brzmi jak coś, co normalnie trzeba zlecać konsultantowi, to tak właśnie jest! Bodźcem do powstania tej książki było wiele lat, jakie spędziłem jako konsultant i szkoleniowiec, pomagając klientom pokonać biznesowe i techniczne wyzwania. Wielokrotnie, przedstawiałem rozwiązania klientom, pytali mnie, „Czy jest jakaś książka na ten temat?". No więc teraz jest! Lubię myśleć o niej jako o „Konsultancie Windows w opakowaniu." Lub w książce, jak sądzę.

W tej książce zająłem się dziesiątkami rodzajów problemów, które widziałem w dużych i małych przedsiębiorstwach. Jej celem jest pomóc czytelnikom, aby ich praca była SMART[1]. Jest to akronim wymyślony przeze mnie w firmie Intelliem. Oznacza on:

- bezpieczna (Secure)
- zarządzana (Managed)
- sprawna (Agile)
- szybka (Rapid)
- solidna (Trustworthy)

Aby osiągnąć warunki SMART i rozwiązać problemy, potrzebujemy więcej niż wewnętrznych szczegółów pojedynczej technologii. Musimy więc zrobić jedną lub więcej z wymienionych poniżej rzeczy:

- Zgromadzić kilka technologii, złożyć je razem we właściwy sposób, aby osiągnąć pożądany rezultat.
- Zautomatyzować powtarzalne kroki za pomocą skryptów.
- Dostroić technologie do procesów biznesowych poprzez przepływy pracy i dostarczanie produktu.
- Zintegrować narzędzia innych firm z narzędziami firmy Microsoft.
- Obejść ograniczenia gotowych funkcji i narzędzi administracyjnych Windows.

Jeśli te dwa ostatnie punkty kogoś zaskoczyły, to niech wie, że zaskoczyły również mnie! Teraz naprawdę będę mógł, czarne na białym, przedstawić Czytelnikom przykłady scenariuszy, których po prostu nie można by przeprowadzić bez przyznania się do ograniczeń systemu Windows, a potem ich obejścia. Jeśli Czytelnicy doceniają ten rodzaj szczerości ze strony firmy Microsoft, to niech o tym dowiedzą ludzie z Microsoft Press! Podejście, które stosujemy w tym zestawie narzędzi, jest niezwykle użytecznym uzupełnieniem

[1] Akronim ten oznacza mądry, sprytny (przyp. tłum.).

dokumentacji i zasobów, które Microsoft dostarcza na temat swoich technologii. Wstawia on technologie do rzeczywistego kontekstu!

Gdy mówimy o rzeczywistości, to co dobrego wynikałoby z tych rozwiązań, jeśli miałyby one zastosowanie tylko do Windows Server 2008?

Ważne Większość rozwiązań w tym zestawie narzędzi została zaprojektowana do stosowania w rzeczywistym, mieszanym środowisku złożonym z systemów Windows XP SP2 lub późniejszym, Windows Vista, Windows Server 2003 SP2 lub późniejszym oraz Windows Server 2008.

„Rozwiązania" są tematem całego tego zestawu narzędzi. Niektóre z nich są proste, niektóre całkiem złożone. Niektóre rozwiązania bazują na innych, a niektóre są niezależne. Wiele rozwiązań składa się ze wskazówek – rodzaju porad, jakie otrzymujemy od konsultanta, które wskazują najlepsze praktyki i pomagają dostroić technologie i biznes. Inne rozwiązania składają się ze skryptów.

Skrypty, które zostały dostarczone na płytce towarzyszącej książce, różnią się od narzędzi zawartych w poprzednich zestawach narzędzi Windows. Nie są to wykonywalne programy użytkowe, które wykonują określone zadanie. Większość z nich to metanarzędzia wykonujące zadania dostarczania i automatyzacji. Zostały tak zaprojektowane, aby nauczyć Czytelników metod podniesienia poziomu administracyjnej wydajności. Aby osiągnąć ten cel ogromna większość tych narzędzi została napisana jako skrypty, w języku VBScript. Będzie więc można je otworzyć, przeczytać i dostosować! Osoby nieznające języka VBScript przekonają się, że większość skryptów wymaga tylko bardzo prostych zmian konfiguracji w naszym środowisku. Zaawansowani skrypciarze będą w stanie rozszerzyć te narzędzia, aby utworzyć rozwiązania, które są nawet mocniejsze i bardziej dostosowane do ich potrzeb.

Skrypty oraz rozwiązania nie kończą się na ostatniej stronie tej książki. Inną, wyraźną zmianą w stosunku do poprzedniego zestawu narzędzi Windows Resource Kits jest witryna społeczności, będąca wsparciem technicznym tych rozwiązań: *www.intelliem.com/resourcekit*. Tam można znaleźć korekty, dyskusje, zmiany i całkiem nowe rozwiązania dostarczone przez społeczność specjalistów IT, którzy czytają tę książkę.

Więcej na ten temat będzie można dowiedzieć się z Zestawu rozwiązań 1. A teraz, na koniec, podziękowanie dla Czytelników za całą pracę, jaką wykonali, aby wspomóc ich organizacje przy korzystaniu z technologii Windows. Wiadomo, że nie zawsze jest to łatwe – ja też jestem na początku drogi. Doceniam czas poświecony przez Czytelników na czytanie, interpretację, dostosowanie i wdrożenie rozwiązań, którymi dzielę się z nimi. I mam nadzieję, że oni będą dzielić się swoją wiedzą i doświadczeniem na witrynie internetowej. Jesteśmy na niej razem i cieszymy się, wnosząc zbiorową wiedzę tysięcy specjalistów, z którymi pracowałem przez lata, aby pomóc nam robić to mądrzej!

Konwencje zastosowane w dokumencie

W tej książce zastosowano następujące konwencje dla zwrócenia uwagi na szczególne funkcje lub zastosowanie.

Pomoce dla czytelnika

W całej książce są stosowane poniższe pomoce dla czytelników, aby zwrócić ich uwagę na przydatne informacje:

Pomoc dla czytelnika	Znaczenie
Uwaga	Podkreśla ważność określonej koncepcji lub zwraca uwagę na specjalny przypadek, który może nie mieć zastosowania w każdej sytuacji.
Ważne	Zwraca uwagę na zasadnicze informacje, które nie powinny być zlekceważone.
Ostrzeżenie	Podkreśla sprawę problematyczną lub kwestię bezpieczeństwa.
Najlepsze rozwiązanie	Przekazuje rady dotyczące strategii i technik, które optymalizują efektywność lub bezpieczeństwo technologii.
Wskazówka	Podsumowuje dyskusję, aby podać „zasadnicze kwestie.”

Przykłady wiersza poleceń

Następujące konwencje stylu są stosowane w całej książce przy dokumentowaniu przykładów wiersza poleceń:

Styl	Znaczenie
Czcionka pogrubiona	Stosowana do wskazywania danych (znaków, które są wpisywane dokładnie tak, jak pokazano).
Kursywa	Stosowana do wskazywania zmiennych, dla których trzeba podać określoną wartość (na przykład *file_name* może odnosić się do *każdej* właściwej nazwy pliku).
Czcionka o stałej szerokości (monospace)	Stosowana dla przykładów kodów i wyników wiersza poleceń.
%SystemRoot%	Stosowana dla zmiennych środowiskowych.

Wymagania systemu

Skrypty dostarczone w płytce towarzyszącej książce zostały w pełni przetestowane w systemie Windows Server 2008. Większość jest zgodna z Windows Server 2003 SP2 lub wersją późniejszą, Windows XP SP2 lub wersją późniejszą oraz Windows Vista, a jeśli istnieją jakieś znane ograniczenia, w zestawie narzędzi lub w samym skryptach zostały podane uwagi na ten temat.

Zawartość oparta na WWW

Poza treścią zawartą w zestawie narzędzi i na płytce towarzyszącej książce, dostępne są dodatkowe informacje na stronie *http://www.intelliem.com/resourcekit*.

Dodatkowe treści online

Gdy dostępny będzie nowy lub zaktualizowany materiał uzupełniający tę książkę, będzie on wysyłany online na stronę Microsoft Press Online Developer Tools Web. Typ materiału, jaki można tam znaleźć, obejmuje aktualizację zawartości książki, artykuły, łącza do treści uzupełniającej, erratę, przykładowe rozdziały i jeszcze więcej. Ta strona WWW będzie dostępna wkrótce pod adresem *www.microsoft.com/learning/books/online/developer* i będzie okresowo aktualizowana.

Obraz dysku towarzyszącego książce

Obraz dysku CD towarzyszącego książce jest dostępny na stronie wydawcy przy opisie książki w zakładce Dodatkowe informacje, pod adresem *http://www.ksiazki.promise.pl/aspx/produkt.aspx?pid=43258*. Płyta CD zawiera przydatne narzędzia i łącza, aby pomóc czytelnikom w instalacji Windows Server 2008. Oto zawartość płyty:

- **Cała książka w wersji elektronicznej.** Elektroniczna wersja *Windows Administration Resource Kit: Productivity Solutions for IT Professionals* w formacie PDF (w języku angielskim).
- **Skrypty.** Więcej niż 75 przykładowych skryptów, które pomogą zautomatyzować systemowe zadania administracyjne.
- **Narzędzia.** Wiele łączy do narzędzi dla IIS, PowerShell, System Center Data Operations i innych, które można natychmiast wykorzystać.
- **Informacje o produkcie.** Łącza do informacji o funkcjach i możliwościach Windows Server 2008, jak również wskazówki dotyczące produktu, które pomagają optymalizować administrację Windows w przedsiębiorstwie.
- **Źródła.** Łącza do dokumentów technicznych, podręczników, publikacji internetowych, grup dyskusyjnych i innych, które pomagają w użyciu i rozwiązywaniu problemów funkcji Windows Server 2008.
- **Przykładowe rozdziały.** Rozdziały z książki Windows Server 2008, począwszy od 15., zawierające całe bogactwo informacji i przegląd innych, wcześniej wydanych tytułów.

Użycie skryptów

Zastosowanie każdego skryptu udokumentowane jest w rozwiązaniu z zestawu narzędzi, które ten skrypt prezentuje. Większość skryptów została napisana w języku Visual Basic Scripting Edition (VBScript), więc można je uruchamiać z wiersza poleceń, pisząc **cscript *script_name*.vbs z następującymi po tym** parametrami wymaganymi przez skrypt. Kilka

skryptów to pliki wsadowe, które można wykonać z wiersza poleceń, wpisując nazwę skryptów i wszelkie parametry.

Na koniec, istnieje pewna liczba aplikacji HTML (HTA), które są skryptami z graficznym interfejsem użytkownika (GUI). HTA można uruchomić, dwukrotnie je klikając. Można również włączyć większość aplikacji HTA do przystawki Użytkownicy i komputery Active Directory przy użyciu kroków zawartych w rozwiązaniu 1.3, tak aby mogły rozszerzyć funkcjonalność naszych wbudowanych narzędzi administracyjnych.

Ważne Niezwykle istotne jest, aby traktować te skrypty jedynie jako przykłady. Każdy skrypt należy otworzyć w programie Notatnik lub edytorze skryptów, przeczytać go wraz z komentarzami, dokonać niezbędnych zmian konfiguracji i przede wszystkim *przetestować skrypt* w środowisku laboratoryjnym przed użyciem go w środowisku eksploatacyjnym. Ani Microsoft, ani też Intelliem nie dają gwarancji na przykłady skryptów lub narzędzia.

Jak wspomniano, będziemy mogli się przekonać, iż za pomocą niewielkich modyfikacji w Bloku konfiguracji skryptu można odnieść sukces w swojej pracy. Autor starał się, aby te skrypty były możliwie jak najbardziej czytelne i możliwe do dostosowania. Skrypty będą aktualizowane na stronie z zestawem narzędzi, *www.intelliem.com/resourcekit*. W razie znalezienia błędów lub dokonania użytecznych zmian w skryptach, proszę przesyłać swoje odkrycia na stronę WWW.

Przed wysłaniem raportu o błędzie należy koniecznie wykonać następujące czynności:

- Przeczytać całe rozwiązanie prezentowane przez skrypt. Upewnić się, że zostały wykonane wszystkie kroki przedstawione w rozwiązaniu.
- Sprawdzić część konfiguracyjną skryptu, która zwykle nosi nazwę *Blok konfiguracji*. W kilku przykładach konfiguracji została użyta fikcyjna domena *contoso.com*. Niektóre skrypty nie będą działać, zanim konfiguracja nie zostanie dopasowana do danego środowiska.
- Dwa razy sprawdzić, czy skrypt jest uruchamiany z uprawnieniami, które umożliwiają wykonanie zadania automatyzowanego przez skrypt. To zadanie może wymagać uruchomienia aplikacji HTA lub wiersza poleceń na wyższym poziomie uprawnień, szczególnie w Windows Vista oraz Windows Server 2008. Trzeba dwukrotnie kliknąć dane narzędzie lub ikonę wiersza poleceń w menu Start i wybrać polecenie Uruchom jako administrator.
- Przetestować skrypt na innych platformach systemu operacyjnego. Większość skryptów i aplikacja HTA będzie działać w wymienionych wcześniej systemach operacyjnych, lecz niektóre skrypty i HTA mają ograniczenia. Niektóre aplikacje HTA wymagają przeglądarki Internet Explorer 7 i nie będą działać z Internet Explorer 6.

Zasady dotyczące wsparcia technicznego dla zestawu narzędzi

Dokonano wszelkich starań, aby zapewnić dokładność tej książki oraz uzupełniającej ją zawartości na płycie. Poprawki do książki są podawane przez Microsoft Press na stronie internetowej pod adresem:

http://www.microsoft.com/learning/support/search.asp

Korekty będą także wysyłane na stronę Resource Kit, pod adresem *http://www.intelliem.com/resourcekit*. Wszelkie komentarze, pytania lub pomysły związane z tą książką lub zawartością towarzyszącej jej płytki, bądź też pytania, na które brakuje odpowiedzi w bazie wiedzy (Knowledge Base), prosimy przesyłać do Microsoft Press, korzystając z jednej z poniższych metod:

poprzez e-mail:

rkinput@microsoft.com

pocztą:

Microsoft Press
Windows Administration Resource Kit: Productivity Solutions for IT Professionals, Editor
One Microsoft Way
Redmond, WA 98052-6399

Prosimy o zwrócenie uwagi, iż poprzez podany wcześniej adres e-mail nie można uzyskać pomocy technicznej. Aby uzyskać informacje na temat wsparcia technicznego, trzeba odwiedzić witrynę Microsoft, Pomoc i wsparcie techniczne:

http://support.microsoft.com

Zestaw rozwiązań 1

Zarządzanie oparte na rolach

W tym rozdziale:

Zapraszamy do Zestawu rozwiązań w książce *Windows Administration Resource Kit: Productivity Solutions for IT Professionals*. W zestawie tym spróbujemy osiągnąć kilka celów jednocześnie.

Po pierwsze, przedstawimy kilka koncepcji i taktyk, które będą stosowane w całym zestawie zasobów. Do końca trzeciego rozwiązania odbędziemy podróż od określenia typowego problemu administracyjnego (w tym przypadku raportowania pełnego, zagnieżdżonego członkostawa użytkownika w grupie) do tworzenia rozwiązania skryptowego, do budowania narzędzia administracyjnego graficznego interfejsu użytkownika (GUI), do integrowania tego narzędzia GUI bezpośrednio w przystawce Użytkownicy i komputery Active Directory.

Umiejętności, jakich nabierzemy w czasie tej podróży, pozwolą implementować wiele innych rozwiązań w zestawie zasobów. Czytelnicy mający doświadczenie w pisaniu skryptów VBScript poznają kilka doskonałych wskazówek i metod wykorzystania swojej wiedzy. Osoby, które nie są doświadczonymi „skrypciarzami", nauczą się, jak wziąć gotowe skrypty i narzędzia zawarte na płytce dodanej do tej książki, zrobić proste modyfikacje, aby mogły one działać w ich przedsiębiorstwie i wykorzystać je w swojej pracy, nawet bez konieczności rozumienia VBScript lub innych języków do tworzenia skryptów.

Ta książka nie jest w zamierzeniu podręcznikiem pisania skryptów. W niektórych rozwiązaniach będziemy przechodzić bezpośrednio do *zastosowania* skryptów w celu rozwiązywania problemów. Wskażemy kilka znakomitych źródeł, które można użyć do nauki pisania skryptów, lecz nie trzeba być skrypciarzem, aby je stosować i czerpać korzyści z rozwiązań zawartych w zestawie zasobów.

W rzeczywistości dość dużo rozwiązań w zestawie narzędzi nie bazuje na skryptach. W dalszej części zestawu znajdziemy rozwiązania, które umożliwiają zarządzanie użytkownikami, komputerami, grupami, bezpieczeństwem – niemalże wszystkim – w sposób

bardziej efektywny. Te rozwiązania, które są skoncentrowane na zarządzaniu opartym na rolach, obejmują zdyscyplinowane zarządzanie grupami w Active Directory oraz dużo analiz i procedur związanych z biznesem. Żadne skrypty same w sobie nie są związane z projektowaniem i implementowaniem zarządzania opartego na rolach, chociaż w tym i innych zbiorach rozwiązań przedstawimy niektóre z nich. Został w nich wykorzystany model zarządzania opartego na rolach i dzięki nim zyskamy siłę supermena do zarządzania naszym przedsiębiorstwem.

A zatem, na końcu tego zestawu rozwiązań będziemy na dobrej drodze do wdrażania w swoim przedsiębiorstwie zarządzania opartego na rolach. To drugi cel tego zestawu. Więcej szczegółów podano w części „Scenariusze, starania i rozwiązania", lecz wystarczy powiedzieć, że zgodnie z doświadczeniami autora z jego klientami, zarządzanie oparte na rolach jest jedną z najbardziej wartościowych inwestycji czasu i myśli, jaką może zrobić specjalista IT.

Scenariusze, problemy i rozwiązania

Gdyby ktoś miał zamiar uczyć się administrowania siecią, pierwsza lekcja, miałaby tytuł: „Nie należy zarządzać przedsiębiorstwem metodą użytkownik po użytkowniku. Trzeba nim zarządzać za pomocą grup". Mamy nadzieję, że jeśli ktoś nie studiował administrowania siecią, ale uczył się pracując, już to po prostu wie!

Dużą wydajność, zarówno po stronie fachowców informatyków, jak i użytkowników tracimy jednak z powodu zarządzania użytkownikami, grupami, komputerami, bezpieczeństwem, konfiguracją, projektowaniem aplikacji i innymi komponentami przedsiębiorstwa informatycznego. Być może niektóre w poniższych scenariuszy będą dla Czytelników brzmiały znajomo:

- Kiedy nowy pracownik zostanie zatrudniony w firmie Contoso Ltd., mija kilka dni, zanim dostanie konto użytkownika. Ponadto, po przyjściu użytkownika do pracy, kilka dni może zająć dostarczenie wszystkich potrzebnych aplikacji oraz dostępu do wszystkich zasobów wymaganych w jego pracy.
- Kiedy jeden użytkownik potrzebuje takiego samego dostępu do zasobów jak inny użytkownik, nie ma prostej metody na sprawdzenie, do czego ma dostęp ten drugi.
- Kiedy użytkownik przenosi się do innej lokalizacji, proces przenoszenia danych i ustawień użytkownika na serwery w nowej lokalizacji nie jest bezproblemowy.
- Kiedy pracownik jest przenoszony do innego miejsca, działu lub pracy, nikt nie może zapewnić, że dostanie on wszystkie nowe aplikacje i zasoby, które są potrzebne. Całkiem często pracownik może nadal mieć dostęp do zasobów, których już nie potrzebuje.
- Kiedy użytkownik rezygnuje z pracy lub zostanie rozwiązana z nim umowa, trudno jest mieć pewność, że całe zarządzanie związane z tym użytkownikiem – na przykład nadane mu uprawnienia – zostały usunięte.
- Trudno jest przeanalizować, ile kopii aplikacji oprogramowania zostało wdrożonych, co utrudnia zarządzanie licencjami.
- Kiedy oprogramowanie jest przypisane do użytkownika – na przykład, gdy Janice potrzebuje Visio – program podąża za nią do sali konferencyjnej. Kiedy zaloguje się ona na innym komputerze, na przykład w sali konferencyjnej, Visio zostanie zainsta-

lowane, co nie powinno się zdarzyć – ponieważ licencja Visio jest na komputer, a nie na użytkownika.

- Trudno jest śledzić, gdzie znajduje się komputer lub użytkownik: w którym biurze, w której lokalizacji Active Directory i tak dalej.
- Wciąż trwają dyskusje na temat projektu jednostki administracyjnej (ang. organizational unit, OU) Active Directory oraz czy powinna być ona oparta na lokalizacjach biurowych, czy też typach obiektów.
- Kiedy użytkownik z określonego powodu potrzebuje dostępu do zasobów, jest on dodawany bezpośrednio do listy kontrolnej tego zasobu, co łamie zasadę „zarządzania poprzez grupy". Na przykład, jeśli Janice z działu marketingu jest ekspertem w pisaniu makr w Microsoft Office Excel i zostanie zatrudniona przez dział księgowości, aby pomóc przy tworzeniu skoroszytu budżetu na następny rok, będzie potrzebować uprawnień do tego skoroszytu. Grupa Accouting (Księgowość) otrzymała już uprawnienia, lecz Janice nie jest w grupie Accouting. Administrator nie chce dodać Janice do grupy Accouting, a ona potrzebuje uprawnień… Co się dzieje? Zwykle dostaje ona uprawnienia bezpośrednio na liście kontroli dostępu (ACL). To rozwiązanie może wystarczyć tylko dla skoroszytu Budget (Budżet). Co jednak ze skoroszytami Sales Forecast (Przewidywana Sprzedaż) i Operations Planning (Planowanie Działań) oraz wszystkimi pozostałymi skoroszytami, do których potrzebny jest „specjalny dostęp"? Widać, dlaczego tracimy nad tym kontrolę.
- Kontrolowanie członkostwa użytkownika w grupie jest trudne. Jeśli administrator zostanie zapytany, „Do jakiej grupy należy Janice?" – i chce, aby odpowiedź zawierała zagnieżdżone grupy – nie można tego stwierdzić albo przynajmniej jest to proces wieloetapowy.
- Kontrolowanie modelu bezpieczeństwa jest wyzwaniem. Jeśli administrator zostanie zapytany, „Kto może przeczytać plany budżetu na następny rok?" lub „Do czego może mieć dostęp Janice?", odpowiedź jest trudna lub nie można jej podać w sposób jednoznaczny.

Ilustruje to tylko kilka *problemów* – scenariuszy, w których tracona jest wydajność, marnowany czas lub pieniądze, zabezpieczenia nie są właściwe lub pojawiają się inne braki efektywności. Te i inne bolączki mogą być rozwiązane przy użyciu zarządzania opartego na rolach. Ten zbiór rozwiązań przygotuje nas do eliminowania wszystkich typów problemów. Niektóre z nich będą całkowicie rozwiązane za pomocą tego zestawu rozwiązań, podczas gdy inne będą wymagać dodatkowych rozwiązań, które przedstawiono w dalszej części tego zestawu narzędzi.

Uwaga Celem zarządzania opartego na rolach jest dostosowanie naszego modelu zarządzania technologią informacyjną do procesów i wymagań biznesowych.

Zarządzanie oparte na rolach pozwala zdefiniować użytkownika (czyli jego *role* – ten termin wyjaśniony jest dalej) i automatycznie umożliwić zarządzanie użytkownikiem, co obejmuje określenie następujących szczegółów:

- Co może lub czego nie może użytkownik (dostęp do zasobów)
- Co użytkownik może lub nie może zrobić w systemach (prawa użytkownika)

- Aplikacje, które są udostępnione użytkownikowi (dystrybucja oprogramowania)
- Środowisko Microsoft Windows użytkownika, takie jak konfiguracja interfejsu użytkownika, poziom „blokowania", odwzorowane dyski itd. (doświadczenia i konfiguracja użytkownika, w tym Zasady grup).
- Zarządzanie danymi użytkownika (przekierowane foldery i profile).
- Listy dystrybucji poczty elektornicznej, do których należy użytkownik.

W podobny sposób zarządzanie oparte na rolach pozwala definiować komputer według jego ról (na przykład komputer biurkowy lub laptop, komputer w sali konferencyjnej, współużytkowana stacja robocza, kiosk publiczny, serwer bazy danych, serwer plików) i dlatego też automatycznie umożliwia zarządzanie tym komputerem, wraz z określeniem następujących szczegółów:

- Co można lub nie można zrobić na tym komputerze (prawa użytkownika)
- Do czego można lub nie można mieć dostęp na tym komputerze (dostęp do zasobów)
- Konfiguracja i zabezpieczenia (Zasady grupy)
- Specjalna konfiguracja dla użytkowników danego komputera (na przykład przetwarzanie obiektów Zasad grupy w pętli zwrotnej)

Mam nadzieję, że niektóre ze scenariuszy przestawionych na początku tej części mogą wydawać się znajome. Są one odzwierciedleniem typów problemów, z jakimi zetknąłem się u każdego klienta, dużego i małego, który nie zaimplementował całkowicie zarządzania opartego na rolach. Porównajmy te scenariusze z wizją, jak mogłoby to wyglądać w przedsiębiorstwie stosującym zarządzanie oparte na rolach. Obraz pokazany w tym scenariuszu może pomóc nam zorientować się, co naprawdę oznacza zarządzanie oparte na rolach.

Ostatnio firma Contoso Ltd. wdrożyła model zarządzania opartego na rolach dla swojego przedsiębiorstwa w systemie Windows. Firma ta zatrudniła fantastycznego konsultanta, Dana Holme'a ... O, chyba trochę odbiegliśmy od tematu…

Teraz do raportowania statusu użytkownika lub komputera w firmie Contoso jest stosowane narzędzie o nazwie My Memberships.

Uwaga Plik My Memberships.hta znajduje się w folderze Scripts na płytce towarzyszącej książce. Jest on omówiony w Rozwiązaniu 1.6.

Ostatnio James Fine został zatrudniony jako księgowy w biurze Denver. W tym czasie zabezpieczenie jego konta użytkownika (jak jest wyjaśnione w Rozwiązaniu 7.7), zostało zdefiniowane w następujących rolach: Accounting i LOC_DEN. Dostał on komputer o nazwie DESKTOP432, który został przypisany do roli LOC_DEN.

Dzięki tym trzem zmianom jego całkowity status (widoczny na zrzucie ekranu My Memberships, na rysunku 1-1) w pełni pozwala mu na wykonywanie pracy księgowego.

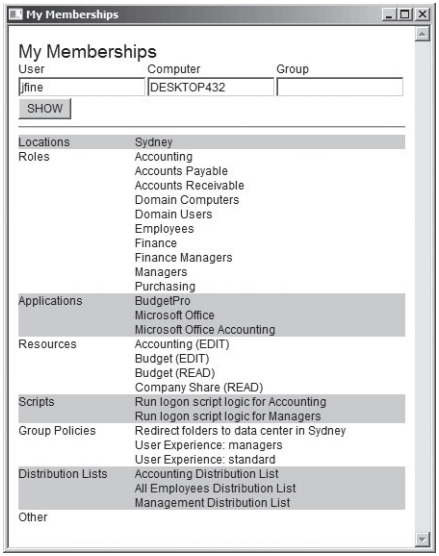
Rysunek 1-1 Zarządzanie użytkownikiem Jamesem Fine, księgowym w Denver

Zwróćmy uwagę, że wszystkie inne uprawnienia James przejął jako zdefiniowany użytkownik z przydzieloną rolą Accounting (Księgowość) w Denver: ma dostęp do kilku zasobów (łącznie z czytaniem zasobów Budget), oprogramowania, konfiguracji, takich jak skrypt logowania Accounting i foldery przekierowane do serwerów oraz listy e-mail. Żadne z tych uprawnień nie są mu przyznane w sposób bezpośredni. Wszystkie są częścią czegoś, co nazywamy zarządzaniem opartym na rolach: logiki biznesowej, która definiuje i obsługuje to, co jest potrzebne księgowym w Denver do ich pracy.

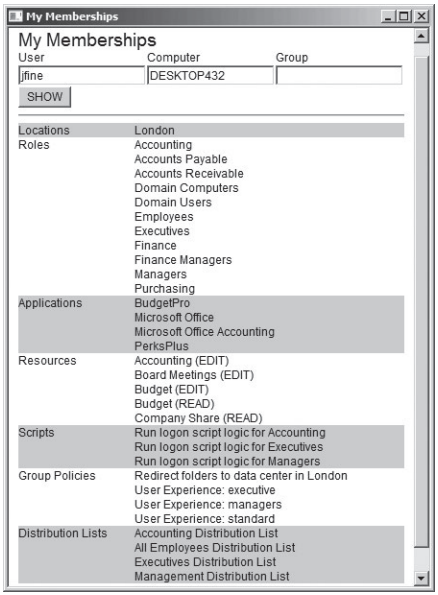
Sześć miesięcy później James dostał awans na menedżera finansowego. W związku z tym jego rola zmieniła się z Accouting na Finance Manager (Menedżer Finansowy). Logika biznesowa w firmie Contoso wskazuje jednak, iż menedżerowie finansowi są *również* księgowymi, a więc James nadal należy do roli Accouting, lecz nie w sposób bezpośredni, a poprzez członkostwo w zagnieżdżonej grupie. Został również przeniesiony do lokalizacji, w której może bardziej efektywnie wykonywać swoją pracę: do Sydney. Dzięki tym dwóm zmianom będzie miał wszystkie uprawnienia pokazane na rysunku 1-2.

Zwróćmy uwagę, że James może teraz *edytować* Budget i ma nową aplikację (BudgetPro), jego foldery zostały przekierowane do Sydney i należy do nowej listy dystrybycji poczty elektronicznej.

Jamesowi idzie tak dobrze, że po dwóch latach otrzymuje awans na dyrektora finansowego (CFO). Jako dyrektor finansowy ma teraz zdefiniowaną rolę Executive (Kierownictwo), jego biuro zostało przeniesione do Londynu, do globalnej centrali. Uzyskuje wszystkie pozostałe role i uprawnienia pokazane na rysunku 1-3.



Rysunek 1-2 Zarządzanie użytkownikiem James Fine, dyrektor finansowy, Sydney

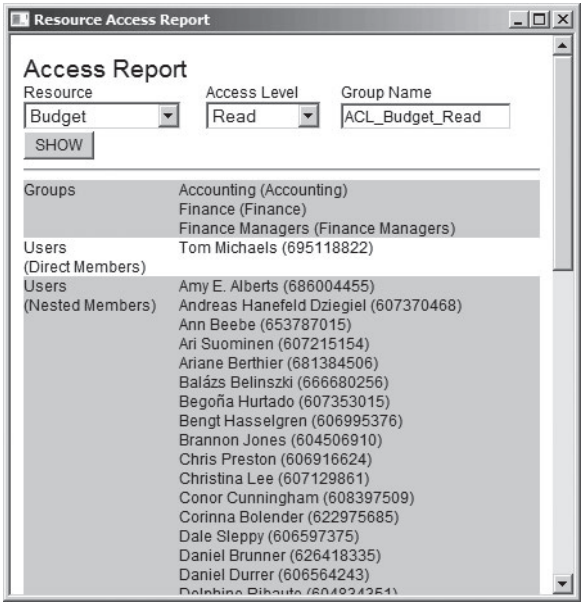


Rysunek 1-3 Zarządzanie użytkownikiem James Fine, dyrektor finansowy CFO/ Executives, Londyn

Jako członek grupy Executive, James zyskał nową aplikację (PerksPlus), dodatkowy dostęp do zasobów (do folderów Board Meeting – Spotkania zarządu), dodatkowe konfiguracje i członkostwo na liście dystrybucji poczty elektronicznej. To wszystko dzięki właśnie tym dwóm zmianom w definicji jego roli: Executive i London (Londyn). James jest zdefiniowany tylko poprzez trzy bezpośrednie role: Finance Managers (Menedżerowie finansowi),

Executives (Kierownictwo) i London (Londyn). Wszystkie pozostałe role i możliwości są dziedziczone z tych trzech ról.

Aby dostrzec następny problem, który pojawia się w kolejnych scenariuszach, spójrzmy, jak łatwe jest dla działu IT utworzenie raportu o tym, kto może mieć dostęp do zasobu na określonym poziomie, co pokazano na rysunku 1-4.



Rysunek 1-4 Raport dotyczący dostępu

Narzędzie pokazane na rysunku 1-4, Access Report, jest również dostępne na płytce dołączonej do książki. Nosi ono nazwę Members_AccessReport.hta i jest umieszczone w folderze Scripts. Jego omówienie znajduje się w Rozwiązaniu 1.6.

Ten scenariusz ilustruje, jak, mając wdrożone zarządzania oparte na rolach, można poprzez proste definiowanie użytkownika lub komputera według ich ról automatycznie dostarczać i zarządzać wszystkim, co zgodnie z wymogami biznesu ma wykonywać dany użytkownik lub komputer. Scenariusz zapewnienia tego jest prosty i można mieć nadzieję, że w głowach Czytelników zrodziło są teraz przynajmniej kilka pytań lub komentarzy, takich jak poniższe:

- „Dobrze, to brzmi wspaniale, ale jak to się *robi*?" Tym właśnie zajmujemy się w tym Zestawie rozwiązań, a także w innych rozwiązaniach w dalszej części książki, zbudowanych na bazie podstaw, jakie tutaj tworzymy. Zrobimy tu mały rzut okiem: ma to dużo wspólnego z zaawansowanym zarządzaniem grupami i równie zaawansowanym zarządzaniem zasobami, aplikacjami, zasadami grup, skryptami i nie tylko. Lecz „zaawansowane" nie oznacza „trudne". Idźmy więc dalej!
- „Moje przedsiębiorstwo jest bardziej skomplikowane…" Oni wszyscy tak mówią. Wierzcie mi. Widziałem już, jak to działa w bardzo dużych organizacjach. Naprawdę działa. Ten scenariusz jest uproszczony, lecz koncepcje, umiejętności i narzędzia, jakimi dysponujemy, pozwolą nam zastosować to rozwiązanie w naszej organizacji.

- „Jeśli to wszystko jest związane z grupami, czy nie skończymy z wielką ilością grup?" Tak! Ale gdy rzeczywiście zobaczymy, co dzieje się obecnie w naszym przedsiębiorstwie, stwierdzimy, że zarządzanie grupami jest znacznie łatwiejsze. A ponadto czytelnicy dostają narzędzia, dzięki którym to wszystko jest prostsze i bardziej automatyczne. Proszę uwierzyć: tak jest łatwiej!
- „Do ilu ról powinien należeć użytkownik?" To faktycznie zależy od liczby czynników i przyjrzymy się temu w dalszej części tego Zestawu rozwiązań. W ostatecznym rezultacie *typowy użytkownik* będzie prawdopodobnie zdefiniowany z małą liczbą, może nawet nie więcej niż kilkunastoma rolami *bezpośrednimi*. Inne role mogą być dziedziczone.

I tutaj może powstać jeszcze więcej pytań i komentarzy. To świetnie! Czytajmy dalej i próbujmy na nie wszystkie odpowiedzieć.

Zanim zagłębimy się w pierwszych rozwiązaniach, podamy tu jeszcze kilka innych uwag.

Reguła 80/20

Sprawdzoną wskazówką jest fakt, że często można rozwiązać 80 procent problemów za pomocą 20 procent wysiłków i na odwrót, rozwiązanie pozostałych 20 procent (czyli doprowadzenie do stanu „idealnego") zabiera 80 procent wysiłków. Jestem gorącym zwolennikiem tego, aby zabierać się do rozwiązywania problemów, mając tę regułę przed oczami. Jeśli możemy rozwiązać 80 procent problemów, zyskamy dzięki temu tak dużo czasu, że będziemy w stanie – przechodząc do innych wyzwań – zrobić dużo więcej pożytecznych rzeczy niż przy próbach zajęcia się każdym wyjątkiem od tej reguły, aby uzyskać wskaźnik 100 procent rozwiązanych problemów.

Na tym właśnie polega wydajność. Uwolnienie tych 80 procent wysiłków, aby wystawić głowę nad wodę, rozejrzeć się, zbudować i zająć się strategiami, by dodać wartość do swojego przedsiębiorstwa i prowadzić zdrowsze, bardziej spełnione życie zawodowe. Ponieważ mamy nadzieję, że ta książka będzie czytana przez miliardy ludzi (OK, to przesada, ale liczę, że przez bardzo wielu) w przedsiębiorstwach każdego typu, rozmiaru, lokalizacji i kultury, najlepsze, co można zrobić, to próba rozwiązania 80 procent wszystkich problemów! To będzie wyczyn godny uczczenia! Podejdźmy do tych rozwiązań, pamiętając o regule 80/20 i będziemy zdumieni, gdy stwierdzimy, ile zyskaliśmy wydajności. Prawdopodobnie tu i tam pojawią się jakieś luki: to szansa dla nas, aby doprowadzić rozwiązanie do ostatniego etapu i perfekcyjnie pracować dla swojej organizacji. Weźmy się do tego!

Skrypty i narzędzia na płycie dołączonej do książki

Płyta towarzysząca książce zawiera wiele dodatków. Są to jedynie przykłady, do których można się odwoływać. Oznacza to, że te narzędzia nie są w rzeczywistości obsługiwane przez Microsoft, autora lub kogoś innego. Należy więc je koniecznie przeanalizować, zrozumieć, *przetestować* (!) oraz przygotować plan wycofania każdego narzędzia, które implementujemy. Trzeba to robić samodzielnie.

Teraz, gdy nasi prawnicy są usatysfakcjonowani, będziemy oczywiście *próbować* uczynić te narzędzia tak użytecznymi, jak to tylko jest możliwe. Trzeba je otworzyć (większość z nich to aplikacje HTML lub pliki HTA, które można otworzyć za pomocą Notatnika) i dokonać wskazanych modyfikacji parametrów, takich jak nazwy firmy i domen. Nie jest

to wielka sprawa. Następnie oceniamy, poznajemy, *testujemy* (!) i przygotowujemy plan wycofania zmian dla każdego narzędzia, jakie zastosowaliśmy. Dobrej zabawy!

Społeczność internetowa, wspomniana w części „Społeczność internetowa Windows Administration Resource Kit" będzie miała dostęp do wszelkich aktualizacji i narzędzi, jak również dyskusji na temat każdego rozwiązania. Będą to bezcenne źródła. A poza wspomnianymi miejscami (w tej książce, w pliku narzędziowym README, w umowie licencyjnej narzędzi lub też w skrypcie bądź komentarzach do skryptu), zachęcamy do zmiany narzędzia i rozszerzenia go, aby spełnić swoje potrzeby. Trzeba będzie przestrzegać uwag i komentarzy dotyczących praw autorskich. Mile widziane będzie dzielenie się swoją pracą na stronie wspólnoty, aby mógł z niej skorzystać każdy jej członek. Jeśli przypomina to komuś trochę otwarty kod, to jest na właściwym tropie. Jesteśmy w tym razem: przestrzegamy zasad, tworzymy i korzystamy.

Microsoft i narzędzia innych firm

W całym tym zestawie narzędzi będą również wspomniane narzędzia pomocne przy rozwiązywaniu problemów, którym stawiamy czoła. Niektóre z nich mogą pozwolić na przebycie ostatnich 20. procent drogi do „doskonałości". Nie ma wątpliwości, że istnieją narzędzia opracowane przez Microsoft i innych twórców oprogramowania, które są bardzo wartościowe dla każdego fachowca systemu Windows.

Na przykład Microsoft Identity Lifecycle Manager (ILM) i System Center Configuration Manager to dwie aplikacje serwera Microsoft, które mogą mieć duże znaczenie w przejściu w kierunku zarządzania opartego na rolach. Takie narzędzia dodają ogromną wartość i prawdopodobnie mają wpływ na zmniejszenie całkowitego kosztu posiadania (ang. *total cost of ownership,* TCO) dla przedsiębiorstwa. Jednak w obliczeniach TCO i zwrotu z inwestycji (ang. *return on investment*, ROI) brany jest pod uwagę *czas* spędzony przez informatyków i użytkowników: ile czasu to zajęło, ile czasu zaoszczędzono i co jest to warte? Okazuje się, ze większość organizacji jest całkiem niezdolna lub niechętna do właściwego rozliczania wysiłków swoich pracowników. Większość organizacji ostatecznie patrzy tylko na koszty liczone w pieniądzach i oszczędności związane z zakupami. Dlatego nie można dokładnie obliczyć ROI, aby pokazać zmniejszony koszt posiadania narzędzia, a narzędzie nie zostanie zakupione. Szkoda, lecz taka postawa jest nazbyt częsta.

Jeśli ktoś ma szczęście i pracuje w organizacji, która ma wytyczne, jak rozliczać czas ludzkiej pracy, musi być pewny, aby liczyć nie tylko pieniądze *zaoszczędzone* dzięki rozwijaniu określonego rozwiązania. To jest łatwa sprawa. Może również być potrzebne obliczenie, ile wartości dodatkowej uzyskamy w końcowym rezultacie, gdy czas poświęcony czemuś, co pochłania wartość, zostanie przekierowany na kreatywne, strategiczne lub taktyczne myślenie i pracę, która tworzy *dodatkową* wartość dla organizacji.

Celem tej książki jest więc dostarczenie narzędzi, które można wdrażać nie ze względu na „twarde" koszty, lecz z nadzieją na optymalne zmniejszenie czasu i wysiłków. Tam gdzie możemy to zrobić bezpłatnie, zrobimy to.

Jeśli zaś istnieją narzędzia, które nie są wbudowane w Windows Server 2008, Windows Server 2003, Windows Vista lub Windows XP, również zostaną one tu przedstawione, zwłaszcza jeśli jest ono bezpłatne. Wiadomo, że ani autor tej książki, ani firma Microsoft, ani też nikt w najmniejszym stopniu związany z tym zestawem narzędzi, nie może zagwarantować, że określone narzędzie zapewni wartość w danej organizacji lub nawet że będzie

pracować zgodnie z założeniami. Można tylko powiedzieć o dobrych doświadczeniach z danym narzędziem u klientów lub o tym, co dobrego się na jego temat słyszało, lecz nie jest to ani poparcie, ani też gwarancja. Należy je przebadać, przetestować i podjąć decyzję właściwą dla swojego przedsiębiorstwa.

Lekcja wydajności w ROI

Kilka razy w życiu byłem – jako konsultant – świadkiem sytuacji, w których klient podawał liczby do obliczania ROI łącznie z czasem zaoszczędzonym przez pracowników. Czas pracownika będzie ogólnie wart więcej niż dwukrotna płaca pracownika po odjęciu podatków, dodatków i kosztów ogólnych.

Wnioski, jakie można było wyciągnąć z tych sytuacji były takie, że gdy ma się już szczęście i dostępne są takie liczby, prawdopodobnie konieczne będzie znacznie zaniżyć obliczenia ROI.

W jednej z takich sytuacji opracowaliśmy rozwiązanie, które podawało wskazówki dotyczące wydajności pracowników działu informacji przedsiębiorstwa. Rozwiązanie obejmowało głównie ukierunkowane szkolenia i internetowy portal samopomocy. W oparciu o analizę oszczędności czasu uzyskanej poprzez to rozwiązanie, która była mierzona kilkakrotnie i była zgodna z moimi doświadczeniami u innych klientów, określiliśmy, że każdy użytkownik oszczędzał 10 minut dziennie. Założyliśmy, że osiągamy tylko część tej siły roboczej, ponieważ nie wszyscy pracownicy w tym partycypują. ROI z oszczędzonymi ponad 10 minutami czasu na dzień wynosi ponad 1000 procent. Jest to *dziesięciokrotny* ROI. Większość firm publicznych stosuje wewnętrzną stopę zwrotu do oceny swoich inwestycji, a ta stopa zwrotu jest w zakresie od 10 do 20 procent. ROI równy 1000 procent jest absurdem (chociaż jest obsługiwany). Przedstawiliśmy wyniki dyrektorowi IT, a jego zespół zgodził się, że w podsumowaniu naszego projektu musimy ustawić ROI na wartość poniżej 100 procent (podwójny ROI). Nikt by nie uwierzył w ROI równy 1000 procent.

Jeszcze bardziej zadziwiający był fakt, że to ROI odzwierciedlało tylko odzyskaną stratę (odzyskanie czasu straconego przez nieefektywne wykonywanie zadań) przez pracowników. Nie zalicza się to do korzyści firmy, ponieważ ten czas był wykorzystany na czynności tworzące wartość. Jeśli mamy szczęście i możemy rozliczać czas pracy ludzkiej, a projekt obejmuje choć kilku pracowników w przedsiębiorstwie, stwierdzimy, że projekty dostosowania wydajności są najprostszym na świecie zadaniem.

Społeczność internetowa Windows Administration Resource Kit

Dla czytelników *Windows Administration Resource Kit* zgromadziliśmy wielkie internetowe zasoby. Wystarczy zalogować się na stronie *http://www.intelliem.com/resourcekit*, aby znaleźć się w społeczności specjalistów techniki informacyjnej, którzy pracują, aby implementować te i inne fantastyczne rozwiązania. Jeśli zajdą jakieś zmiany w skryptach lub narzędziach dostarczanych w tej książce lub na towarzyszących jej nośnikach, lub też znajdą się w nich jakieś (ojej!) błędy, zamieścimy je w tej witrynie. A Czytelnicy będą mogli dyskutować o tym, co działa, co nie działa i jakie rozwiązania wybrać do następnego etapu.

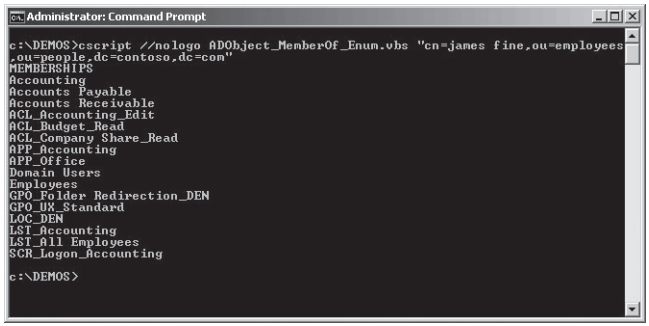
Już dość!

Jesteśmy więc już przygotowani na niektóre rzeczy, których doświadczymy w całym zestawie narzędzi. Jesteśmy gotowi na podstawowe zrozumienie problemów możliwości zarządzania, które pomoże nam rozwiązać zaganienie zarządzania opartego na rolach. Zostały podane informacje o płytce towarzyszącej książce oraz społeczności internetowej; wiadomo, czego życzą sobie pracownicy odnośnie skryptów i narzędzi wspomnianych w książce. Zagłębmy się w rozwiązania z tego zbioru! Najpierw rozgrzejmy nasze umiejętności projektowania niestandardowych narzędzi i potem zastosujemy je do zarządzania opartego na rolach.

1.1. Wyliczanie członkostwa użytkownika (lub komputera) w grupach

Ogólny opis rozwiązania

Typ rozwiązania	Narzędzie (skrypt)
Funkcje i narzędzia	Polecenia wiersza poleceń, VBScript
Krótki opis rozwiązania	Tworzy skrypt do wyliczenia grup, do których należy użytkownik (lub komputer).
Korzyść	Odpowiada na pytanie: „Do jakich grup należy ____?” Dzięki temu członkostwo w grupach staje się bardziej widoczne niż przy użyciu standardowych zestawów narzędzi GUI.
Podgląd	Rysunek 1-5 jest ilustracją wyniku skryptu, który raportuje przynależność użytkownika do grup.



Rysunek 1-5 Skrypt raportu o członkostwie w grupach

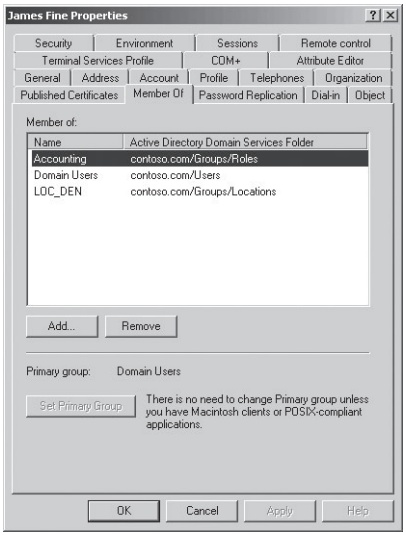
Wprowadzenie

Nasze pierwsze rozwiązanie stanowi przykład typowego podejścia do problemu rozwiązywanego w tej książce. Zaczynamy od problemu znanego większości administratorów i określamy ograniczenia standardowych narzędzi administracyjnych Windows. Następnie przyglądamy się rozszerzonym narzędziom, które zapewniają funkcjonalność pozwalającą obejść dany problem. Na koniec budujemy skrypt, który całkowicie rozwiązuje problem.

W przedstawionych poniższej rozwiązaniach zmieniamy skrypt w narzędzie administracyjne za pomocą interfejsu GUI. Następnie podłączamy narzędzie do standardowej przystawki Komputery i użytkownicy z Active Directory, aby nasza nowa, rozwiązująca problem funkcjonalność była dostępna z poziomu standardowego narzędzia administracyjnego. Śledząc i stosując te rozwiązania, Czytelnicy będą mieli podgląd unikatowych podejść tworzonych w całej książce.

Użytkownicy i komputery Active Directory

W życiu każdego administratora przychodzi taka chwila, gdy musi tworzyć raporty o grupach, do których należy użytkownik. Można by pomyśleć, że po dziesięciu latach w biznesie tworzenia narzędzi administracyjnych opartych na GUI dla pracy sieciowej Windows, standardowe zestawy narzędzi firmy Microsoft (obejmujące przystawkę Użytkownicy i komputery Active Directory) powinny zapewniać tę funkcjonalność. Lecz tego nie robią. Jesteśmy oczywiście w stanie otworzyć właściwości obiektu użytkownika i zobaczyć jego bezpośrednie członkostwa na zakładce Członek grupy, pokazanej na rysunku 1-6. Jednak ta lista nie ujawnia zagnieżdżonego członkostwa tego użytkownika. Na przykład nie można zobaczyć, że James jest członkiem grupy Accounts Payable (Płatności), ponieważ grupa Accouting jest członkiem grupy Accounts Payable.



Rysunek 1-6 Zakładka Member Of okna dialogowego Właściwości obiektu użytkownika

Można dwukrotnie kliknąć dowolną grupę na liście Członek grupy, aby otworzyć jej właściwości i kontynuować „drążenie", zanim znajdziemy grupę, która nie jest członkiem żadnej innej.

Uwaga Aby otworzyć właściwości dowolnej grupy na liście Member Of, można dwukrotnie kliknąć tę grupę na liście.

To byle jaka, powolna metoda odkrywania członkostwa użytkownika w grupach. Jeśli ktoś dostaje pieniądze za godzinę, może to być jego ulubiony sposób pracy. Reszta musi poszukać lepszej metody.

Polecenia DS

Wejdźmy do poleceń DS. Polecenia DS są narzędziami wiersza poleceń do uzyskiwania dostępu do Active Directory i wykonywania w nim operacji. Te polecenia wykorzystamy w kilku różnych rozwiązaniach w tej książce. Do celów tego zadania wykorzystamy dsget.exe, które pozwala uzyskać właściwość obiektu Active Directory. Wymagana składnia jest następująca:

```
dsget user UserDN -memberof -expand
```

Pierwszy parametr, *user*, wskazuje klasę obiektu, którego „dodaniem" jesteśmy zainteresowani. Drugi parametr, *UserDN*, jest pełną nazwą obiektu. Trzeci parametr, *memberof*, jest pożądaną przez nas właściwością; a czwarty parametr dla nas „magiczny" – informuje *dsget*, aby rozszerzać członkostwo, tak aby zawierało zagnieżdżone grupy. Tak więc, aby odkryć członkostwo naszego użytkownika, Jamesa Fine'a, stosujemy takie polecenie:

```
dsget user "cn=James fine,ou=employees,ou=people,dc=contoso,dc=com" -memberof -expand
```

Wpisywanie całej wyróżniającej nazwy użytkownika (ang. *distinguished name*, DN) nie należy do rzeczy przyjemnych, więc wykorzystamy inne polecenie DS do szukania użytkownika: dsquery.exe przeszukuje Active Directory przy użyciu podanych właściwości i zwraca pełną nazwę pasujących obiektów. Jeśli wprowadzimy

```
dsquery user -samid jfine
```

dostaniemy pełną nazwę obiektu użytkownika wraz z właściwością *samid* (nazwa logowania dla systemów starszych niż Windows), która pasuje do jfine, w szczególności:

```
cn=James fine,ou=employees,ou=people,dc=contoso,dc=com
```

Ponieważ *dsquery* tworzy pełną nazwę, której wymaga *dsget*, można „doprowadzić" wyniki *dsquery* do *dsget*, jak pokazano poniżej:

```
dsquery user -samid jfine | dsget user -memberof -expand
```

Zwrócmy uwagę, że polecenie *dsget* już nie zawiera wyróżniającej nazwy (DN). Jeśli pominiemy wyróżniającą nazwę obiektu, który chcemy uzyskać, *dsget* będzie oczekiwał jej wpisania. Spróbujmy wpisać wyróżniającą nazwę obiektu lub obiektów, ujętą w cudzysłów i nacisnąć Ctrl+C, aby wskazać zakończenie wpisywania. Aby dostarczyć DN, stosujemy tu standardowy strumień wejściowy, *StdIn*.

Symbol potoku (|) powoduje przekierowanie rezultatu wygenerowanego przez polecenie *dsquery* do *StdIn* polecenia *dsget*, zaoszczędzając nam konieczności wpisywania DN. Trzeba wpisać tylko nazwę logowania użytkownika w systemach starszych niż Windows 2000.

Nadal można ulepszyć rezultat naszego polecenia *dsquery* | *dsget*: grupy są wymienione jako nazwy wyróżniające, co nie wygląda zbyt ładnie. Zmodyfikujmy nasze polecenie, aby zwracało tylko nazwy grup:

```
dsquery user -samid jfine | dsget user -memberof -expand | dsget group -samid
```

Zwóćmy uwagę, że kierujemy potokowo listę wyróżniających nazw dostarczanych przez pierwsze polecenie *dsget* do drugiego polecenia *dsget*. Drugie polecenie *dsget* określa, że każda nazwa wyróżniająca będzie należeć do tej grupy i że chcemy odzyskać właściwość *samid*. Zauważmy, że faktyczna nazwa właściwości nazwy dla systemów starszych od Windows 2000, zgodnie ze schematem Active Directory, to *sAMAccountName*. Niestety polecenia DS nie są zgodne z własnymi standardami firmy Microsoft, więc *samid* jest stosowanym przełącznikiem.

A zatem rozwiązanie z pierwszego wiersza poleceń jest pakietem trzech poleceń:

```
dsquery user -samid UserSAMAccountName | dsget user -memberof -expand | dsget group
-samid
```

Jeśli chcemy znaleźć członkostwo w grupie dla komputera, zwyczajnie wstawiamy *computer* jako klasę obiektu w poleceniach DS i w zapytaniu o komputer, używając jego zwykłej nazwy:

```
dsquery computer -name ComputerName | dsget computer -memberof -expand | dsget group
-samid
```

Tworzenie skryptu wsadowego

Teraz, gdy znamy polecenia DS wymagane do wyliczania członkostwa w zagnieżdżonych grupach, możemy zamienić je w proste, jednowierszowe skrypty. Otwieramy Notatnik i wprowadzamy następujący wiersz:

```
dsquery user -samid %1 | dsget user -memberof -expand | dsget group -samid
```

Zapisujemy skrypt jako User_MemberOf_Enum.bat". Trzeba się upewnić, że nazwa pliku jest ujęta w cydzysłów; w przeciwnym razie Notatnik doda rozszerzenie.txt. Dobrze jest zapisać skrypt w łatwo dostępnym folderze, takim jak folder w katalogu głównym dysku.

Następnie w wierszu poleceń przechodzimy do folderu, w którym zapisaliśmy skrypt i uruchamiamy ten skrypt jak poniżej:

```
User_MemberOf_Enum UserSAMAccountName
```

Na przykład:

```
User_MemberOf_Enum jfine
```

Wpisanie *%1* w skrypcie spowoduje wstawienie pierwszego argumentu, jaki podamy po tym poleceniu (nazwę logowania użytkownika w systemach starszych niż Windows 2000 lub „SAM nazwa konta") na właściwe miejsce w sekwencji polecenia. Skrypt może być bardziej elegancki, jeśli przypiszemy pierwszy argument do zmiennej, a potem będziemy używać tej zmiennej w sekwencji polecenia, jak poniżej:

```
set userdn=%1
dsquery user -samid %userdn% | dsget user -memberof -expand | dsget group -samid
```

Pierwszy wiersz przypisuje argument do zmiennej o nazwie *userdn* (nazwy zmiennych nie mogą być słowami zarezerwowanymi), a do zmiennej odwołujemy się, otaczając jej nazwę znakami procentu (%) w sekwencji polecenia.

Wyliczanie członkostwa w grupach za pomocą języka VBScript

VBScript to mocny i elastyczny język do tworzenia skryptów, który udostępnia pewną liczbę rozwiązań administracyjnych. Ta książka nie jest przeznaczona do nauki języka VBScript. Osobom, które chcą nauczyć się pisania skryptów, poleca się *Microsoft Windows Administrator's Automation Toolkit* (Microsoft Press, 2005) lub inne książki napisane przez jej autora, guru pisania skryptów, Dona Jonesa.

Przy prezentowaniu skryptów i stosowaniu ich w tej książce, stwierdzimy, że w większości przypadków są one gotowe do użycia po niewielkich modyfikacjach, do których wykonania *nie jest wymagane doświadczenie w pisaniu skryptów*. Skrypty często będą sprawdzane na pewnym poziomie szczegółowości. Osoby nieprzygotowane do czytania tych szczegółów mogą opuścić tę część dyskusji. Będą podane instrukcje, jak *użyć* skryptu w sposób łatwy do zrozumienia. Zatem na przykład osoba niepisząca skryptów może opuścić poniższą dyskusję i przejść do punktu „Zastosowanie ADObject_MemberOf_Enum.vbs".

```
ADObject_MemberOf_Enum.vbs
```

Poniższy kod tworzy specjalny typ tabeli zwanej słownikiem i wypełnia ją członkostwem w grupach użytkownika, Jamesa Fine'a:

```
' Tworzenie obiektu słownika do przechowywania nazw grup
Set oMemberOfList = CreateObject("Scripting.Dictionary")
' Ustawianie trybu porównania do rozróżniania wielkości znaków
oMemberOfList.CompareMode = vbTextCompare

Set oUser = GetObject("LDAP://cn=James
fine,ou=employees,ou=people,dc=contoso,dc=com")
Call ADObject_MemberOf_Enum(oUser)

aMemberOfList = oMemberOfList.Keys
WScript.Echo "MEMBERSHIPS"
For Each sMemberOf In aMemberOfList
    WScript.Echo sMemberOf
Next

Sub ADObject_MemberOf_Enum(ByVal oObject)
     ' Wyliczanie właściwości obiektu MemberOf
     ' DANE:  oObject: obiekt, dla którego na być wyliczane MemberOf
     ' WYMAGANE:  obiekt słownika oMemberOfList z globalnym zakresem

    On Error Resume Next
    aMemberOf = oObject.GetEx("memberOf")
    If Err.Number <> 0 Then
        ' Zakładamy, że błąd jest spowodowany brakiem wyróżniającej nazwy memberOf
        Err.Clear
        On Error GoTo 0
        Exit Sub
    Else
        On Error GoTo 0
        For Each sMemberOf In aMemberOf
            Set oMemberOf = GetObject("LDAP://" & sMemberOf)
            sMemberOfSAM = oMemberOf.sAMAccountName
            If (oMemberOfList.Exists(sMemberOfSAM) = False) Then
                ' Nie widzieliśmy jeszcze tej grupy
                ' Dodajemy ją do obiektu słownika
                oMemberOfList.Add sMemberOfSAM, True
                ' Wyliczamy zagnieżdżone członkostwo tej grupy
                Call ADObject_MemberOf_Enum(oMemberOf)
            End If
        Next
    End If

End Sub
```

Motorem napędowym tego skryptu jest podprogram o nazwie *ADObject_MemberOf_Enum*. Procedura ta bierze obiekt Active Directory (użytkownika, grupę lub komputer) i sprawdza jego właściwość *MemberOf*. W przypadku znalezienia właściwości *MemberOf* zostanie sprawdzone, czy grupa została już dodana do listy członkostwa. Jeśli nie, będzie dodana do listy, a następnie procedura wywoła samą siebie, aby sprawdzić członkostwo w danej grupie. Ten iteracyjny, zagnieżdżony proces jest zwany rekurencją i umożliwia podanie całej

listy członkostwa oryginalnego obiektu w grupach, łącznie z członkostwem wynikającym z zagnieżdżenia grup.

Lista członkostwa utworzona przez ten skrypt jest prawie, lecz nie całkiem, kompletna. Każdy obiekt użytkownika i komputera w Active Directory ma główne członkostwo w grupie: użytkownicy zwykle należą do domeny Użytkownicy, a komputery do domeny Komputery. Ta podstawowa przynależność do grupy nie jest przechowywana we właściwości *MemberOf*, lecz we właściwości *PrimaryGroupID*. *PrimaryGroupID* musi być przekształcone na rzeczywistą nazwę grupy. Na przykład *PrimaryGroupID* 514 odpowiada domenie Użytkownicy.

Poniższy fragment kodu bierze użytkownika lub komputer Active Directory, reprezentowany przez *oObject* i dodaje główną grupę do naszego słownika listy członkostwa:

```
iPrimaryGroup = oObject.PrimaryGroupID
sPrimaryGroup = ""
Select Case iPrimaryGroup
    Case 513
        sPrimaryGroup = "Domain Users"
    Case 514
        sPrimaryGroup = "Domain Guests"
    Case 515
        sPrimaryGroup = "Domain Computers"
    Case 516
        sPrimaryGroup = "Domain Controllers"
End Select
If sPrimaryGroup > "" Then
    Call ADObject_MemberOf_Enum(sPrimaryGroup)
    oMemberOfList.Add sPrimaryGroup, True
End If
```

Mając te dwa ważne komponenty, możemy zbudować resztę skryptu. Skrypt ten nosi nazwę ADOBJECT_MEMBEROF_ENUM.vbs i – tak jak wszystkie rozwiązania ze skryptami zawarte w tej książce – można znaleźć go w folderze Scripts na płytce towarzyszącej książce. W książce nie zawsze będzie przedstawiany cały skrypt, tak aby więcej stron zostało na podanie większej liczby rozwiązań. Jednak w tym przypadku używamy tego skryptu do pokazania sposobu budowania skryptów w całej książce, stosowanych konwencji i zastosowania tych skryptów w swoim przedsiębiorstwie.

ADOBJECT_MEMBEROF_ENUM.vbs

```
Option Explicit
Dim oMemberOfList
Dim aMemberOfList
Dim sMemberOf
Dim sDomainDN
Dim sArgument
Dim sObjectDN

' BLOK KONFIGURACJI
sDomainDN = "dc=contoso,dc=com"

If WScript.Arguments.Count <> 1 Then
    ' podano złą liczbę argumentów
```

```
    WScript.Echo "USAGE: cscript //nologo <name of script> [DN | ADSPath | NAME]"
    WScript.Echo "where the object can be represented by a distinguished name,"
    WScript.Echo "ADSPath, pre-Windows 2000 logon name (user/group) or name
(computer)"
    WScript.Quit(501)
    ' 501 jest kodem wyjściowym, używanym do wskazania rodzaju błędu w napisanych
      skryptach
End If

' Pobranie argumentu
sArgument = WScript.Arguments(0)
' przekształcenie go na DN
sObjectDN = ADObject_DN_UCG(sArgument, sDomainDN)
If sObjectDN = "" Then
    WScript.Echo "ERROR--NOT FOUND: " & sArgument
    WScript.Quit (501)
End If

' Utworzenie obiektu katalogowego do przechowywania nazw grup
Set oMemberOfList = CreateObject("Scripting.Dictionary")
' Ustawienie trybu porównania na nierozróżnianie wielkości znaków
oMemberOfList.CompareMode = vbTextCompare

Call ADObject_MemberOf (sObjectDN)
aMemberOfList = oMemberOfList.Keys
aMemberOfList = Array_Sort(aMemberOfList)
WScript.Echo "MEMBERSHIPS"
For Each sMemberOf In aMemberOfList
    WScript.Echo sMemberOf
Next

' ======================
' FUNKCJE Z BIBLIOTEKI
' ======================
Function ADObject_DN_UCG(ByVal sObject, sSearchDN)
    ' Wersja 070706
    ' Pobiera dane (nazwa, DN lub ADsPath) użytkownika, komputera lub grupy,
    ' i zwraca wyróżniającą nazwę obiektu
    ' DANE:    sObject:   nazwa, DN lub ADsPath do użytkownika grupy lub komputera
    '           sSearchDN: DN w którym na być szukane (często DN domeny,
                           np. dc=contoso, dc=com)
    ' WYNIK:   ADObject_DN_UCG: wyróżniająca nazwa (cn=..). obiektu
    ' WYMAGANIA: Procedura ADObject_Find_UCG

    Dim sObjectName, oObject, sObjectADsPath, sObjectDN

    If Len(sObject) = 0 Then
        sObjectDN = ""
    ElseIf Len(sObject) < 3 Then
        ' to nie może być DN lub ADsPath – musi być nazwa
        sObjectADsPath = ADObject_Find_UCG(sObject, sSearchDN)
        If sObjectADsPath <> "" Then sObjectDN = mid(sObjectADsPath,8)
    ElseIf LCase(Left(sObject,3)) = "cn=" Then
        ' to jest DN – trzeba się upewnić, czy istnieje
        On Error Resume Next
```

```
        Set oObject = GetObject("LDAP://" & sObject)
        sObjectDN = oObject.distinguishedName
        If Err.Number <> 0 Then
            ' Błąd – nie można znaleźć obiektu
            sObjectDN = ""
            Err.Clear
        End If
        On Error GoTo 0
    ElseIf Len(sObject) < 8 Then
        ' to nie może być ADsPath i nie jest DN, musi być nazwa
        sObjectADsPath = ADObject_Find_UCG(sObject, sSearchDN)
        If sObjectADsPath <> "" Then sObjectDN = mid(sObjectADsPath,8)
    ElseIf Ucase(Left(sObject, 7)) = "LDAP://" Then
        ' to jest ADsPath – trzeba się upewnić, czy istnieje
        ' po pierwsze, trzeba się upewnić, że LDAP:// jest wielkimi literami, aby
uniknąć            błędu
        sObject = "LDAP://" & Mid(sObject, 8)
        On Error Resume Next
        Set oObject = GetObject(sObject)
        sObjectDN = oObject.distinguishedName
        If Err.Number <> 0 Then
            ' Błąd – nie można znaleźć obiektu
            sObjectDN = ""
            Err.Clear
        End If
        On Error GoTo 0
    Else
        ' to musi być nazwa
        sObjectADsPath = ADObject_Find_UCG(sObject, sSearchDN)
        If sObjectADsPath <> "" Then sObjectDN = mid(sObjectADsPath,8)
    End If

    ADObject_DN_UCG = sObjectDN

End Function

Function ADObject_Find_UCG(sObjectName, sSearchDN)
    ' VERSION 070706
    ' Zwraca pełną ADsPath (LDAP://..). użytkownika, komputera lub grupy
    ' Dane wejściowe:
    '      sObjectName: Unikatowy identyfikator dla klasy obiektu.  Skrypt obsługuje:

'         Użytkownika: sAMAccountName (nazwa użytkownika w systemie sprzed Windows
2000)
    '         Grupaę:    sAMAccountName (nazwa użytkownika w systmie sprzed Windows
2000)
    '         Komputer: Nazwa (przekształcana przez skrypt dla sAMAccountName
komputera
              poprzez dodanie znaku $)
    '      sSearchDN:   DN, w którym mamy szukać (często DN domeny np..
                        dc=contoso, dc=com)

    Dim oConnection
    Dim oRecordset
    Dim sLDAPObjectQuery
```

```
    Dim sLDAPIdentifierQuery
    Dim sLDAPQuery
    Dim sProperties
    Dim oADObject
    Dim aProperties
    Dim sProperty
    Dim sLDAPIdentifier
    Dim sSearchScope

    sLDAPIdentifier = "samAccountName"
    sProperties = "ADsPath"
    sSearchScope = "subtree"

    ' Otwieranie połączenia ADO przy użyciu pustych danych uwierzytelniających
    Set oConnection = CreateObject("ADODB.Connection")
    oConnection.Provider = "ADsDSOObject"
    ' w przypadku błędu kontynujemy przy etykiecie Next
    oConnection.Open "", vbNullString, vbNullString
    If oConnection.State = 0 Then ' 0 = adStateClosed
         ' Błąd obsługi kodu: nie można połączyć się z AD
         ADObject_Find_UCG = ""
         Exit Function
    End If

    ' Budowanie zapytania LDAP
    sLDAPQuery = "<LDAP://" & sSearchDN & ">;"
    sLDAPQuery = sLDAPQuery & _
        "(|(samAccountName=" & sObjectName & ")" & _
        "(samAccountName=" & sObjectName & "$));" & _
        sProperties & ";" & sSearchScope

    ' Wyszukanie zestawu wyników, zamknięcie połączenia i sprawdzenie,    ' czy
otrzymaliśmy przynajmniej jeden wynik
    Set oRecordset = oConnection.Execute (sLDAPQuery)
    If oRecordset.EOF and oRecordset.BOF Then
        ' Kod obsługi błędu: nie znaleziono obiektu
        ADObject_Find_UCG = ""
        Exit Function
    End If

    ADObject_Find_UCG = oRecordset.Fields("ADsPath")
    oRecordset.Close
    oConnection.Close

End Function

Sub ADObject_MemberOf(ByVal sObjectDN)
    ' Wersja 070701
    ' Wylicza członkostwo użytkownika lub komputera w grupach
    ' DANE:       sObjectDN: DN użytkownika lub komputera
    ' WYNIK:      oMemberOfList posortowana według każdej grupy,
    '             do której należy użytkownik lub komputer.
    ' WYMAGANIA: funkcja ADObject_MemberOf_Enum
    '             funkcja Array_Sort
    '             lista oMemberOf o zasięgu globalnym
```

```
    '             przykład
    '             Set oMemberOfList = CreateObject("Scripting.Dictionary")
    '             oMemberOfList.CompareMode = vbTextCompare
    '             bez rozróżniania wielkości liter

    Dim oObject
    Dim iPrimaryGroup
    Dim sPrimaryGroup
    Dim aMemberOf

    On Error Resume Next
    Set oObject = GetObject("LDAP://" & sObjectDN)
    If Err.Number<>0 Or oObject Is Nothing Then
        Err.Clear
        Exit Sub
    End If
    On Error GoTo 0

    If oObject.Class = "computer" Or oObject.Class = "user" Then
        iPrimaryGroup = oObject.PrimaryGroupID
        sPrimaryGroup = ""
        Select Case iPrimaryGroup
            Case 513
                sPrimaryGroup = "Domain Users"
            Case 514
                sPrimaryGroup = "Domain Guests"
            Case 515
                sPrimaryGroup = "Domain Computers"
            Case 516
                sPrimaryGroup = "Domain Controllers"
        End Select
        If sPrimaryGroup > "" Then
            Call ADObject_MemberOf_Enum(sPrimaryGroup)
            oMemberOfList.Add sPrimaryGroup, True
        End If
    End If

    On Error Resume Next
    aMemberOf = oObject.GetEx("memberOf")
    If Err.Number <> 0 Then
        ' Zakładamy, że błąd jest spowodowany brakiem pełnych nazw w memberOf
        Err.Clear
        On Error GoTo 0
        Exit Sub
    Else
        On Error GoTo 0
        Call ADObject_MemberOf_Enum(oObject)
    End If

End Sub

Sub ADObject_MemberOf_Enum(ByVal oObject)
    ' Wersja 070701
    ' Podaje właściwość MemberOf obiektu
    ' DANE:     oObject: obiekt, dla którego ma być wyliczane MemberOf
```

```
    ' WYMAGANIA: obiekt słownika oMemberOfList z globalnym zakresem
    '             np.
    '             Set oMemberOfList = CreateObject("Scripting.Dictionary")
    '             oMemberOfList.CompareMode = vbTextCompare
    '             bez rozróżniania wielkości liter

    Dim aMemberOf
    Dim sMemberOf
    Dim oMemberOf
    Dim sMemberOfSAM

    On Error Resume Next
    aMemberOf = oObject.GetEx("memberOf")
    If Err.Number <> 0 Then
        ' Zakładamy, że błąd jest spowodowany brakiem nazw wyróżniajacych
w memberOf
        Err.Clear
        On Error GoTo 0
        Exit Sub
    Else
        On Error GoTo 0
        For Each sMemberOf In aMemberOf
            Set oMemberOf = GetObject("LDAP://" & sMemberOf)
            sMemberOfSAM = oMemberOf.sAMAccountName
            If (oMemberOfList.Exists(sMemberOfSAM) = False) Then
                ' Nie widzieliśmy jeszcze tej grupy
                ' Dodajemy ją do obiektu słownika
                oMemberOfList.Add sMemberOfSAM, True
                ' Wyliczenie zagnieżdżonego członkostwa danej grupy
                Call ADObject_MemberOf_Enum(oMemberOf)
            End If
        Next
    End If

End Sub

Function Array_Sort(ByVal aArray)
    ' WERSJA 070701
    ' Bardzo szybkie sortowanie tabeli przy użyciu obiektów .NET
    ' DANE:        aArray: tabela (zawierająca obiekt słownika) wymagająca
sortowania
    ' WYNIK:       Array_Sort: posortowana
    ' WYMAGANIA:   .NET Framework 2.0

    Dim oNAL
    Dim nIdx

    Set oNAL = CreateObject("System.Collections.ArrayList")
    For nIdx = 0 To UBound(aArray)
        oNAL.Add aArray(nIdx)
    Next

    oNAL.Sort

    For nIdx = 0 To UBound(aArray)
```

```
        aArray(nIdx) = oNAL(nIdx)
    Next

    Array_Sort = aArray

End Function
```

Aby uruchomić skrypt, otwieramy wiersz poleceń, przechodzimy do folderu zawierającego skrypt i wprowadzamy polecenie:

```
cscript.exe ADOBJECT_MEMBEROF_ENUM.vbs "nazwa użytkownika, komputera lub grupy"
```

Aby skrypt był bardzo elastyczny, został on tak napisany, że umożliwia wskazanie obiektu, który chcemy analizować przy użyciu jednego z poniższych elementów:

- Pełna nazwa (DN) użytkownika, komputera lub grupy.
- Pełna ścieżka LDAP do obiektu (LDAP://CN=…), która ma prefiks „LDAP://", po którym następuje DN obiektu. Nosi ona nazwę ADsPath obiektu.
- Nazwa użytkownika dla systemu starszego niż Windows 2000: atrybut użytkownika *sAMAccountName*.
- Nazwa konta dla grupy w systemie starszym niż Windows 2000: atrybut grupy *sAMAccountName*.
- Nazwa komputera: atrybut komputera *Name*. Atrybut komputera *sAMAccountName* jest jego atrybutem *Name* z dodanym znakiem dolara ($), tak aby skrypt mógł otrzymać *sAMAccountName* na podstawie *Name komputera*.

Możliwość wprowadzania prostej nazwy jako argumentu została zapewniona z dwóch powodów. Po pierwsze, prostą nazwę jest łatwo wpisać, zwłaszcza w porównaniu z DN. Po drugie, w następnym rozwiązaniu będziemy uruchamiać skrypt nie z wiersza poleceń, lecz z konsoli zarządzania MMC (Microsoft Management Console), z wybranym konkretnym użytkownikiem (lub komputerem bądź grupą). Zależnie od tego, jak uruchamiany jest ten skrypt, jako argument będą podawane różne postaci nazwy obiektu, więc skrypt musi być gotowy do przyjmowania różnych typów nazw.

Przykłady poleceń do uruchamiania skryptu obejmują poniższe:

```
cscript //nologo ADOBJECT_MEMBEROF_ENUM.vbs "cn=James
fine,ou=employees,ou=people,dc=contoso,dc=com"
```

lub

```
cscript //nologo ADOBJECT_MEMBEROF_ENUM.vbs jfine
```

Przełącznik *//nologo* chowa typowy nagłówek z polecenia *cscript*. W pierwszym przykładzie jako argument skryptu jest podawana pełna nazwa Jamesa Fine'a. W drugim przykładzie podawana jest nazwa użytkownika (nazwa konta w systemie starszym niż Windows 2000). Jeśli ostatni argument zawiera spacje, potrzebny jest cudzysłów, tak jak w pierwszym przykładzie.

Skrypt zaczyna się od instrukcji *Option Explicit*, która wymaga, aby zadeklarować każdą zmienną przed jej użyciem (instrukcja *Dim*), jak widać w podanych poniżej wierszach. Najlepszą praktyką w pisaniu skryptów jest użycie *Option Explicit* i deklarowanie

zmiennych, ponieważ zmniejsza to prawdopodobieństwo literówek w nazwach zmiennych. W skryptach na dołączonej do książki płytce będą użyte *Option Explicit* i deklaracje, chociaż zwykle te wyrażenia są pomijane przy druku skryptu w książce, aby koncentrować się na funkcjonalnych aspektach skryptu.

Jeśli jedna z tych nazw zostanie przekazana do skryptu, będzie on szukał pasujących obiektów w domenie określonej przez *DomainDN* w bloku konfiguracji (*Configuration Block) skryptu*.

Ważne Wiele skryptów i narzędzi dodawanych w tym zestawie narzędzi zawiera blok konfiguracji, w którym będzie dokonywana większość lub wszystkie konieczne dostosowania dla potrzeb przedsiębiorstwa. Trzeba również czytać zawarte w skrypcie komentarze, wskazujące inne dostosowania, które mogą być konieczne. Komentarze w skrypcie VBScript zaczynają się od apostrofu (’), a komentarze w języku HTML są otaczane dwoma znacznikami: <!— na początku i --> na końcu.

Domenę kodujemy na sztywno, ponieważ niepowtarzalność atrybutu *sAMAccountName* jest gwarantowana tylko w pojedynczej domenie. Często stosowaną alternatywą dla określenia DN jest połączenie z obiektem *RootDSE* i sprawdzenie domyślnego kontekstu nazewnictwa, lecz to nie działa w lesie z pustym katalogiem głównym i podrzędną domeną ze wszystkimi użytkownikami i komputerami. Właśnie dlatego kodujemy na sztywno nazwę domeny.

Wprawni skrypciarze mogą wybrać inne metody dynamicznego określenia właściwej wartości *sDomainDN*. W skrypcie tym *sDomainDN* jest deklarowana jako zmienna, a nie nie jako stała, aby można było elastycznie przydzielać jej wartość.

WScript.Arguments() jest zbiorem argumentów, które są przekazywane do skryptu w wierszu poleceń. Właściwość kolekcji *Count* służy do zapewnienia, że podano jeden i tylko jeden argument skryptu. Jeśli tak nie jest (co oznacza, że albo nie został określony argument lub określono ich zbyt wiele), zostanie wyświetlone przypomnienie.

Jeśli argument został przekazany do skryptu, jest on przypisany do zmiennej *sArgument*. Zbiór *Arguments* rozpoczyna się od zera, więc *WScript.Arguments(0)* jest pierwszym argumentem.

Skrypt używa potem dwóch procedur z naszej biblioteki, aby przekształcić argument na DN wybranego obiektu. Te dwie procedury to funkcje: *ADObject_DN_UCG* i *ADObject_Find_UCG*. Te dwie funkcje razem biorą nazwę użytkownika, komputera lub grupy i zwracają nazwę wyróżniającą obiektu Active Directory. Nazwa, którą przyjmują funkcje może być nazwą wyróżniającą (na przykład CN= . .), ADsPath (na przykład, LDAP://CN= . .) lub *sAMAccountName* (nazwa konta w systemie wcześniejszym niż Windows 2000 Logon Name). Jeśli podamy nazwę wyróżniającą lub ADsPath, procedury potwierdzą, że obiekt istnieje i zwrócą jego nazwę wyróżniającą. Jeśli podamy *sAMAccountName*, fukcja *ADObject_Find_UCG* będzie szukać obiektu pasującego w zakresie wyszukiwania – zwykle całej domeny, która jest przekazywana do funkcji jako nazwa (na przykład DC=contoso,DC=com).

Skrypt wywołuje funkcję *ADObject_DN_UCG*, która jest odpowiedzialna za przekształcanie na nazwę wyróżniającą (DN) dowolnego z czterech typów nazw, które mogą być użyte dla obiektu. Jeśli podana nazwa nie jest DN ani ADsPath, funkcja wywołuje *ADObject_Find_UCG*, ktory przeszukuje określony zakres (w tym przypadku całą domenę), aby odszukać użytkownika, komputer lub grupę z nazwą, która pasuje do określonego argumentu. Następnie skrypt przygotowuje obiekt słownika *oMemberOfList*, przechowujący wszystkie

grupy, których członkiem jest użytkownik lub komputer. Kod wywołuje następnie podprogram *ADObject_MemberOf*, który zaczyna proces wyliczania członkostwa w grupach. Kod przekazuje do podprogramu pełną nazwę obiektu określoną wcześniej w tym kodzie.

Podprogramy wypełniają słownik *oMemberOfList* każdą grupą, do której należy obiekt (użytkownik, komputer lub grupa), łącznie z jego grupą główną. Podprogramy te są dalej omówione w szczegółach. Pole klucza obiektu wynikowego słownika jest kopiowane do tablicy, *aMemberOfList*, która jest potem sortowana przez funkcję *Array_Sort*. Tablica jest wyświetlana element po elemencie.

Za wypełnianie *oMemberOfList* odpowiadają dwa podprogramy. Piewszy, *ADObject_MemberOf*, tworzy odwołanie do obiektu i jeśli jest to użytkownik lub komputer, interpretuje jego grupę główną. Jeśli obiekt jest członkiem innych grup, jest on przekazywany do *ADObject_MemberOf_Enum*, który – jak opisano wcześniej w tym rozwiązaniu – sprawdza rekurencyjnie członkowstwo w grupach, dodając każdą grupę tylko raz do słownika *oMemberOfList*. Ten skrypt, tak jak inne skrypty w tej książce, daje możliwość umiarkowanego wyłapywania błędów. Kiedy klient pyta mnie, jaka jest różnica między pisaniem skryptu a programowaniem, często podkreślam, że skrypt wykonuje zadania i kropka. Aplikacja musi być bardziej rozbudowana, bardziej elastyczna, bardziej odporna i zdolna do obsługiwania szerszego zakresu danych. Pisząc skrypt, zawsze czynimy jakieś założenia i ustępstwa w imię praktyczności i efektywności. Oczywiście zachęcam wszystkich do rozszerzenia obsługi błędów, sprawdzania poprawności danych lub innych takich charakterystyk.

Użycie skryptu ADObject_MemberOf_Enum.vbs

Skrypt ADOBJECT_MEMBEROF_ENUM.vbs znajduje się w folderze Scripts na nośniku towarzyszacym tej książce. Aby uruchomić ten skrypt, otwieramy wiersz poleceń (cmd.exe) i wprowadzamy następujące polecenie:

```
cscript //nologo ADOBJECT_MEMBEROF_ENUM.vbs "nazwa użytkownika, komputera lub grupy "
```

Trzeba starannie testować ten i wszystkie pozostałe prezentowane tu skrypty. Skrypty są krótkimi, wydajnymi rozwiązaniami, a nie aplikacjami w pełni testowanymi. Trzeba więc mieć pewność, że rozumie się skrypt, rozumieć modyfikacje, które trzeba lub można wykonać i w pełni przetestować skrypt laboratoryjnie przed wdrożeniem go do eksploatacji.

Dlaczego VBScript?

Jak niektórzy czytelnicy może już wiedzą, Windows Server 2008 obsługuje Windows PowerShell jako interfejs do administracji i automatyzacji. Windows PowerShell może być również ładowany z witryny strony Microsoft Web i instalowany w Windows Vista, Windows XP oraz Windows Server 2003.

W większości rozwiązań w tej książce będziemy jednak wykorzystywać VBScript. Powód jest taki, że chcemy tworzyć kod, który jest na tyle elastyczny i możliwy do wielokrotnego użytku, jak to tylko jest możliwe. Na przykład podprogramy VBScript, które napisaliśmy wcześniej, mogą być w łatwy sposób włączone do skryptu logowania, do strony WWW lub do aplikacji HTML (.hta). Faktycznie następne rozwiązanie zawiera kod w typowym administracyjnym GUI, napisany jako plik .hta. W każdym z tych scenariuszy można uruchamiać VBScript, nie martwiąc się o to, czy Windows PowerShell jest, czy nie jest zainstalowany.

Nawet w Windows Server 2008 nie ma gwarancji, że PowerShell jest dostępny, ponieważ musi być zainstalony jako funkcja. Wreszcie PowerShell jest nowym produktem i nie zapewnia pełnego zestawu funkcji. Dużo naszych rozwiązań dotyczy Active Directory, dla którego na przykład PowerShell nie ma tak wielu możliwości jak dla innych komponentów, takich jak Windows Management Instrumentation (WMI).

Więc w celu uzyskania maksymalnej elastyczności, możliwości ponownego zastosowania, zgodności i bogatego zestawu funkcji, VBScript często – choć nie zawsze – jest wybieraną przez nas bronią. Można mieć pewność, że przyszłe wydania tej książki będą zawierały większą liczbę skryptów PowerShell, ponieważ PowerShell oraz technologie klienta i serwera Windows stają się coraz bardziej zintegrowane.

Kolejne kroki

Skrypt ADObject_MemberOf_Enum.vbs jest gotowym rozwiązaniem do wyliczania przynależności użytkownika, komputera lub grupy do grup. Jego funkcjonalność nie różni się niczym od sekwencji poleceń *dsquery | dsget | dsget*, poza sortowaniem wyniku. Rzeczywiste powody, dla których poświęcamy tyle czasu dla tego skryptu, to – po pierwsze – przedstawienie naszego podejścia do pisania skryptów w tej książce. Po drugie, w następnych dwóch rozwiązaniach będziemy korzystać z tego kodu, aby tworzyć naprawdę przydatne narzędzie administracyjne oparte na GUI. Więc warto czytać dalej!

Gdzie szukać więcej informacji

Aby dowiedzieć się więcej na temat VBScript, warto zacząć od niewiarygodnych zasobów w TechNet Script Center, na stronie *http://www.microsoft.com/technet/scriptcenter/default.mspx*. Gorąco polecamy również książkę *Microsoft Windows Administrator's Automation Toolkit* (Microsoft Press, 2005) lub dowolny inny tytuł na ten temat, autorstwa Dona Jonesa – guru pisania skryptów. Są tam niezliczone pomoce do nauki języka VBScript i znajdowania kodu w Internecie. Warto poszukać podając słowa kluczowe wskazujące, co próbujemy osiągnąć za pomocą skryptu i dołączając słowo kluczowe „VBScript". Na przykład, gdy będziemy wyszukiwać frazy **vbscript group membership**, pojawi się wielki wybór podejść do problemu, który właśnie rozwiązaliśmy. Oczywiście jedną z najbardziej cenionych i aktywnych witryn jest witryna Dona Jonesa: *www.ScriptingAnswers.com*.

Podsumowanie rozwiązania

Standardowe narzędzia administracyjne Windows, oparte na GUI nie podają całego członkostwa użytkownika lub komputera w grupach, ponieważ nie pokazują przynależności do grup zagnieżdżonych. Polecenie *dsget [użytkownik |kcomputer] -memberof -expand* wymieni przynależność do wszystkich grup, podobnie jak podany w tym rozwiązaniu skrypt , ADObject_MemberOf_Enum.vbs w języku VBScript. To rozwiązanie jest jednak na „pół surowe". Następne dwa rozwiązania zostały tak opracowane, aby utworzyć bardzo użyteczne rozszerzenie do przystawki Użytkownicy i komputery Active Directory.

1.2. Tworzenie narzędzia GUI do wyliczania członkostwa w grupach

Ogólny opis rozwiązania

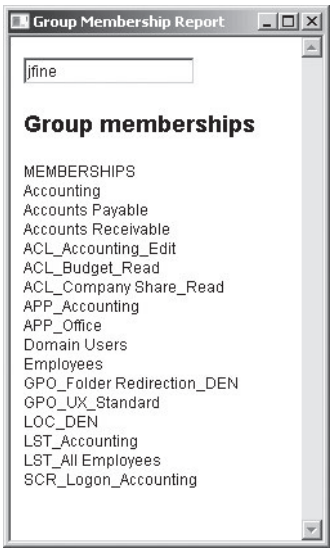
Typ rozwiązania Aplikacja HTML (HTA)

Funkcje i narzędzia HTA, HTML, VBScript

Krótki opis rozwiązania Tworzy narzędzie GUI do wyliczania członkostwa w grupach użytkownika (lub komputera) i uczy, jak zmienić skrypty w narzędzia administracyjne.

Korzyści Odpowiada na pytanie: „Do jakich grup należy ____?" Dzięki temu członkostwo w grupach staje się bardziej widoczne niż przy użyciu standardowych zestawów narzędzi GUI.

Podgląd Na rysunku 1-7 pokazano w działaniu raport HTA o członkostwie w grupach, wyliczający członkostwo użytkownika, którego nazwa konta w systemie starszym niż Windows 2000 to jfine.



Rysunek 1-7 Raport HTA o członkostwie w grupach

Wprowadzenie

Rozwiązanie 1.1 przedstawia skrypt VBScript, który wylicza, czyli tworzy listę członkostwa w grupach dla użytkownika, komputera lub grupy, włącznie z członkostwem w zagnieżdżonych grupach i główną grupą konta. To rozwiązanie przenosi skrypt do następnego poziomu, tworząc narzędzie administracyjne z graficznym interfejsem użytkownika (GUI) poprzez zaprojektowanie prostej strony WWW i aplikacji HTML Application (w skrócie HTA).

Narzędzia administracyjne oparte na WWW mają tę zaletę, iż można mieć do nich dostęp z dowolnego system używającego przeglądarki WWW, takiej jak Internet Explorer. Można aktualizować i zarządzać tym narzędziem w jednej lokalizacji i nie trzeba instalować komponentów w klientach. Problem związany z narzędziami opartymi na WWW jest jednak taki, że Internet Explorer ma pewną liczbę ograniczeń w zabezpieczeniach dotyczących typu kodu, który jest dopuszczony do wykonywania. Niektóre z tych ograniczeń można obejść, umieszczając stronę administracyjną WWW w strefie zabezpieczeń Lokalny Intranet. Można również programować narzędzia administracyjne jako strony ASP.NET, które działają w kontekście zabezpieczeń puli aplikacji witryny WWW i mogą wykonywać dużą ilość kodu po stronie serwera. W ten sposób dostarczamy klientowi złożoną stronę, której niewiele brakuje do tego, by była skryptem nieklienckim.

Aplikacje HTML

Jako narzędzia administracyjne można alternatywnie wykorzystywać aplikacje HTML (HTA). HTA są niezależnymi aplikacjami napisanymi za pomocą standardowego kodu HTML, VBScript i/lub JavaScript. Kod HTA jest identyczny jak kod HTML strony WWW z wyjątkiem dodatkowego znacznika *<HTA>*. Nie jest on jednak uruchamiany za pomocą przeglądarki Internet Explorer (iexplore.exe), lecz wykonywany przy użyciu programu mshta.exe, który zawiera mniej ograniczeń bezpieczeństwa dla kodu, jaki jest uruchamiany w HTA.

Ponieważ aplikacje HTA są niezależne, mogą również być uruchamiane, gdy nie ma dostępu do sieci. Mogą być uruchamiane z lokalnego dysku systemu, z klucza USB lub z nośników optycznych – aby je udostępniać, nie jest potrzebny serwer WWW.

Aby programować aplikacje HTA, trzeba dobrze znać języki pisania skryptów, takie jak VBScript i z HTML. Można użyć edytorów HTML, jak Microsoft Expression lub FrontPage 2003, aby ułatwić sobie pracę z HTML. Edytory skryptów mogą uprościć korzystanie z VBScript. Chociaż ta książka nie jest z założenia podręcznikiem pisania skryptów, to rozwiązanie pozwoli nabrać umiejętności potrzebnych do zamiany każdego skryptu w aplikację HTA.

Osoby, które nie są mocne w pisaniu skryptów, mogą nadal *korzystać z* aplikacji HTA, które napisaliśmy. Każda z nich znajduje się w folderze Scripts na płytce dołączonej do książki. Trzeba również dokonać modyfikacji w aplikacji HTA, co można zrobić poprzez otwarcie HTA w Notatniku lub innym edytorze. Nie należy korzystać z Microsoft Office Word lub innego edytora tekstowego, który może zmienić znaki lub formatowanie. Na przykład Word może zmienić cudzysłowy proste („) zaokrąglone („ ”). W każdej aplikacji HTA wbudowane są komentarze lub skrypt, który wskazuje, co trzeba zmienić, aby aplikacja HTA działała w danym otoczeniu.

Warto, aby nawet osoby niezbyt zaawansowane w językach HTML lub VBScript, przeczytały to rozwiązanie, przyglądając się, jak można w całkiem łatwy sposób włączyć dobry skrypt do aplikacji HTA. Trzeba zwrócić uwagę na to, jak kontrolki wejściowe, takie jak pola tekstowe, mogą być użyte zamiast argumentów wiersza poleceń skryptu do dostarczania parametrów do narzędzia GUI.

Tworzenie HTA

W rzeczywistości przekształcanie dobrze napisanego skryptu na aplikację HTA jest całkiem trywialne. Poniższe omówienie będzie dobrym startem do opanowania działań, które są do tego konieczne.

Najpierw piszemy kod funkcjonalny

Zanim zaczniemy się martwić, jak wygląda nasza aplikacja HTA –„interfejs" dla naszego narzędzia administracyjnego – uruchomimy nasz skrypt. Cały kod powinien być oparty na niezależnych podprogramach i funkcjach, co oznacza, że powinny one pobierać dane z parametrów przekazywanych do procedury lub znajdujących się w kontrolkach wejściowych (takich jak pola tekstowe) na stronie HTA i zwracać wyniki do kodu wywołującego lub wyświetlać je na stronie. Jeśli skrypt został tak napisany, aby przyjmował argumenty i efektywnie wyświetlał wyniki, powinno być to łatwe zadanie.

Na przykład nasz skrypt ADObject_MemberOf_Enum.vbs jest w dużym stopniu modularny, a rzeczywistą pracę wykonują jego podprogramy i funkcje. Funkcjonalność skrytu jest zatem prawie gotowa do migracji do kodu właściwego dla HTA. Tylko kod znajdujący się na samym początku procedury nie jest częścią podprogramu lub funkcji. Ten kod, bez zgłoszeń dla użytkownika, jest pokazany poniżej:

```
Option Explicit
Dim oMemberOfList
Dim aMemberOfList
Dim sMemberOf
Dim sDomainDN
Dim sArgument
Dim sObjectDN

' BLOK KONFIGURACJI
sDomainDN = "dc=contoso,dc=com"

' Pobieranie argumentu
sArgument = WScript.Arguments(0)
' Pryekształcanie go w DN
sObjectDN = ADObject_DN_UCG(sArgument, sDomainDN)
If sObjectDN = "" Then
        WScript.Echo "ERROR--NOT FOUND: " & sArgument
        WScript.Quit (501)
End If

' Tworzenie obiektu katalogu do przechowywania nazw grup
Set oMemberOfList = CreateObject("Scripting.Dictionary")
' Ustawianie trybu porównywania, aby nie rozróżniał wielkości liter
oMemberOfList.CompareMode = vbTextCompare

Call ADObject_MemberOf (sObjectDN)
aMemberOfList = oMemberOfList.Keys
aMemberOfList = Array_Sort(aMemberOfList)
WScript.Echo "MEMBERSHIPS"
For Each sMemberOf In aMemberOfList
        WScript.Echo sMemberOf
Next
```

Ten kod kontroluje przepływ w skrypcie. Wszystko, co musimy zrobić, to zamienić go w podprogram, otaczając go wyrażeniami *Sub* i *End Sub*:

```
Option Explicit
Dim oMemberOfList
Dim sDomainDN
' BLOK KONFIGURACJI
sDomainDN = "dc=contoso,dc=com"

Sub MainRoutine()
    Dim aMemberOfList
    Dim sMemberOf
    Dim sArgument
    Dim sObjectDN
    ' Pobieranie argumentu
    sArgument = WScript.Arguments(0)
    ' Przekształcanie go w DN
    sObjectDN = ADObject_DN_UCG(sArgument, sDomainDN)
    If sObjectDN = "" Then
        WScript.Echo "ERROR--NOT FOUND: " & sArgument
        WScript.Quit (501)
    End If

    ' Tworzenie obiektu słownika do przechowywania nazw grup
    Set oMemberOfList = CreateObject("Scripting.Dictionary")
    ' Ustawianie trybu porównywania, aby nie rozróżniał wielkości liter
    oMemberOfList.CompareMode = vbTextCompare

    Call ADObject_MemberOf (sObjectDN)
    aMemberOfList = oMemberOfList.Keys
    aMemberOfList = Array_Sort(aMemberOfList)
    WScript.Echo "MEMBERSHIPS"
    For Each sMemberOf In aMemberOfList
        WScript.Echo sMemberOf
    Next
End Sub
```

Zwróćmy uwagę, że deklarujemy zmienne *oMemberOfList* i *sDomainDN* na zewnątrz podprogramu. Powodem tego jest fakt, że dla tego konkretnego kodu potrzebujemy zmiennych o zakresie globalnym, które mogą zatem być współdzielone przez funkcje i podprogramy.

Tworzenie interfejsu użytkownika: HTML

Miejmy nadzieję, że mamy edytor HTML, taki jak FrontPage 2003 lub Microsoft Expression Web, aby wspomóc tworzenie HTML dla naszej aplikacji HTA. Jeśli nie, można użyć Notatnika. Poniżej pokazano istotne, wymagane wiersze kodu:

```
<html>
<head>
<title>My Administrative Tool</title>
<hta:application>

<script language="vbscript"> SCRIPT GOES HERE </script>
</head>
```

```
<body> USER INTERFACE HTML GOES HERE </body>
</html>
```

Znacznik <*title*> powoduje utworzenie tekstu w pasku tytułu przeglądarki. Kod HTML, który umieszczamy pomiędzy znacznikiem otwierającym <*body*> i zamykającym </*body*> powoduje utworzenie widocznych elementów interfejsu użytkownika. Wykonywany kod jest umieszczony między znacznikiem otwierającym <*script*> i zamykającym </*script*>. Te elementy są wspólne dla wszystkich narzędzi administracyjnych opartych na sieci WWW.

Aby utworzyć HTA, trzeba zapisać stronę z rozszerzeniem .hta. Ponadto dodajemy znacznik <*hta:application*>, pozwalający na uruchomienie aplikacji pod mshta.exe, bez ograniczeń w zabezpieczeniach, które zwykle dotyczą kodu opartego na WWW.

Można z pewnością zrobić więcej, dodając style, metadane i inne znaczniki do strony, lecz tu jest „sucha" struktura HTA. Więcej informacji na temat aplikacji HTA można znaleźć na stronie *http://msdn2.microsoft.com/en-us/library/ms536471.aspx*. Tam znajdziemy przegląd aplikacji HTA i szczegóły dotyczące znacznika <*hta:application*>.

Tworzenie interfejsu użytkownika:dane wejściowe

Do migracji na kod właściwy dla HTA potrzebnych będzie tylko kilka wierszy naszego skryptu, które przetwarzają dane wejściowe:

```
sArgument = WScript.Arguments(0)
sObjectDN = ADObject_DN_UCG(sArgument, sDomainDN)
```

Ten kod przypisuje pierwszy przekazany do skryptu argument zmiennej o nazwie *sArgument*, a następnie przekształca ten argument na pełną nazwę przy użyciu funkcji *ADObject_DN_UCG*. W aplikacji HTA parametry, argumenty lub dane wejściowe do kodu będą dostarczane przy użyciu kontrolek wejściowych, takich jak pola tekstowe, listy rozwijalne, pola listy i pola wyboru. Te kontrolki będą częścią treści strony, zawartej między znacznikami <*body*>.

Na przykład poniższy wiersz kodu, umieszczony po znaczniku otwierającym <*body*>, utworzy pole tekstowe, do którego można wprowadzać pełną nazwę użytkownika lub komputera, których członkostwo w grupie chcemy podać:

```
<input type="text" name="txtObjectDN" id="txtObjectDN" size="20">
```

Atrybuty znacznika <*input*> tworzą pole tekstowe o wielkości 20 znaków, z nazwą (*name*) oraz identyfikatorem (*id*) *„txtObjectDN"*. Atrybuty *name* i *id* powinny być ustawione na tę samą wartość. Konkretna nazwa nie ma znaczenia, chociaż nie można stosować pewnych zarezerwowanych słów (takich jak *body*) i według konwencji proponowane jest użycie prefiksu do wskazania typu kontrolki (na przykład *txt* dla pola tekstowego).

Atrybuty *name* oraz *id* naszych kontrolek wejściowych mogą być potem użyte przez kod do odczytania, co użytkownik wprowadził do tych kontrolek. Jeśli zmienimy poniższe wiersze naszego skryptu

```
sArgument = WScript.Arguments(0)
sObjectDN = ADObject_DN_UCG(sArgument, sDomainDN)
```

na

```
sObjectDN = ADObject_DN_UCG(txtObjectDN.value, sDomainDN)
```

dokonamy migracji wejściowej części skryptu na WWW. Kiedy kod zostanie uruchomiony, wartość wprowadzona do pola tekstowego będzie przekazana to procedury *ADObject_DN_UCG* i przekształcona na pełną nazwę. Całkiem proste, prawda?

Uruchamianie zdarzeń

Lecz co faktycznie powoduje, że kod działa? Inaczej niż w niezależnym skrypcie, który wykonuje kod, gdy zostanie otwarty, aplikacje HTA i strony WWW są sterowane przez zdarzenia. To oznacza, że kod działa tylko wtedy, gdy pojawi się zdarzenie wywołujące go. Są dwie metody wiązania zdarzenia z kodem, który chcemy uruchomić – procesu zwanego uruchamianiem zdarzeń.

Pierwszą możliwością jest dodanie atrybutu procedury obsługi zdarzenia do znacznika HTML:

```
<input type="text" name="txtObjectDN" id="txtObjectDN" size="20"
onchange="MainRoutine()"/>
```

Pola tekstowe i wiele innych kontrolek wejściowych stają się obiektem zdarzenia, gdy wartość kontrolki zostaje zmieniona. Atrybut procedury obsługi zdarzenia *onchange* powoduje, że klient wykonuje określony kod, gdy to zdarzenie zajdzie.

Podobnie można dodać do strony przycisk, który po kliknięciu wykonuje podprogram:

```
<input type="button" name="btnSubmit" id="btnSubmit" value="Go"
onclick="MainRoutine()"/>
```

Drugą opcją jest nadanie nazwy podprogramowi do automatycznego uruchamiania procedury obsługi zdarzenia. Zamiast nadawania naszemu podprogramowi nazwy *MainRoutine()*, możemy go nazwać *txtObjectDN_onChange()*. W nazwie tej nie są rozróżniane wielkie i małe litery, lecz sama nazwa musi mieć postać *control ID_event*. Jeśli mamy przycisk, lecz pominiemy atrybut *onclick*, możemy nazwać nasz podprogram *btnSubmit_onClick()*. Gdy nadamy podprogramowi nazwę o takim formacie, zdarzenie to będzie automatycznie wywoływać kod.

Do wyzwolenia kodu można nawet użyć zdarzeń na poziomie strony. W wielu naszych aplikacjach HTA można znaleźć atrybut procedury obsługi zdarzenia zawarty w znaczniku <*body*>, tak jak w poniższym przypadku:

```
<body onload="Initialize()">
```

Gdy strona zostanie załadowana, jest wykonywana procedura do przeprowadzenia zadań inicjalizacji, takich jak zmiana rozmiaru okna lub inicjalizacja zmiennych.

Tworzenie interfejsu użytkownika: wyjście

Przy tworzeniu aplikacji HTA do dostarczenia wyniku nie będą używane instrukcje *WScript.Echo*. W rzeczywistości *Script.Echo* nie jest dostępny do użycia, natomiast *MsgBox* jest: procedura *MsgBox* jest stosowana do wyświetlania komunikatów o błędach, jeśli nazwa wprowadzona w polu tekstowym nie jest właściwa.

Jednak dla większości wyników będziemy zwykle zmieniać kod HTML wyświetlanej strony. Na przykład poniższy skrypt wyświetla komunikat, „The page loaded" (Strona załadowana) w obszarze (*div*) strony HTML, określonym jako *„MessageBox"*:

```
<html>
<head>


<script language="vbscript">
Sub DisplayMessage(sMessageHTML)
   MessageBox.innerHTML = sMessageHTML
End Sub
</script>

</head>

<body onload="DisplayMessage('<h1>The page loaded</h1>')">

Results:
<div id="MessageBox" name="MessageBox"> </div>

</body>
</html>
```

W pokazanym właśnie kodzie zdarzeniem wywołującym podprogram *DisplayMessage* jest ładowanie strony – atrybut znacznika treści dla procedury obsługi zdarzenia *onload*.

Aby przedstawić inne zdarzenie, dodajmy poniższy wiersz kodu powyżej znacznika zamykającego *</body>*:

```
<input type="button" name="btnTest" id="btnTest" onclick="DisplayMessage('<i>You just
 clicked the button.</i>')" value="Click Me"/>
```

Ten wiersz powoduje utworzenie przycisku. Atrybuty przycisku *name* oraz *id* są ustawione na *btnTest*. Procedura obsługi zdarzenia *onclick* wywołuje podprogram *DisplayMessage*. Etykieta na przycisku, czyli *value*, jest ustawiona tak, że wyświetlany jest napis „Click Me" (Kliknij mnie).

Ten rodzaj techniki będziemy stosować do tworzenia wyjść, więc sprawdźmy, czy nasz skrypt wygeneruje wyniki, które będą efektywnie wyświetlane na stronie HTML, bez instrukcji *WScript.Echo*. Nasz skrypt do wyliczania członkostwa w grupach zawiera kod, który wyświetla listę członkostwa:

```
WScript.Echo "MEMBERSHIPS"
For Each sMemberOf In aMemberOfList
   WScript.Echo sMemberOf
Next
```

Można również zmienić kod, tak aby utworzyć listę członkostwa, a następnie wyświetlić ją na stronie WWW, tak jak poniżej:

```
sMessageHTML = "MEMBERSHIPS"
For Each sMemberOf In aMemberOfList
   sMessageHTML = sMessageHTML & "<br/>" & sMemberOf
Next
Call DisplayMessage(sMessageHTML)
```

Ostateczna aplikacja HTA

Po włożeniu do naszego kodu funkcjonalnego podprogramów i funkcji, utworzeniu interfejsu użytkownika z kontrolkami wejściowymi, podłączeniu zdarzeń do wywoływania kodu i zapewnieniu mechanizmu wyświetlania wyników, jesteśmy całkiem blisko „doskonałej" aplikacji HTA. Nosi ona nazwę ADObject_MemberOf_Enum.hta i można ją znaleźć w folderze Scripts na płytce towarzyszącej książce.

Zwócimy uwagę na dodatkowy element. Do elementu strony *<head>* został dodany znacznik *<style>*:

```
<style>
   body, tr, td, table, p, input {font-family: arial; font-size: 9pt;}
</style>
```

To spowoduje ustawienie czcionki HTA na 9-punktowy Arial.

Ostateczny kod HTA jest pokazany poniżej, z tym że kilka funkcji, które są częścią naszej biblioteki funkcji, nie zostało rozwiniętych, tak aby łatwiej można było ocenić główne elementy i działanie HTA:

ADObject_MemberOf_Enum.hta

```
<html>
<head>
<title>Group Membership Report</title>
<hta:application>
<script language="vbscript">
Option Explicit
Dim oMemberOfList
Dim sDomainDN
' BLOK KONFIGURACJI
sDomainDN = "dc=contoso,dc=com"

Sub MainRoutine()
    Dim aMemberOfList
    Dim sMemberOf
    Dim sResults
    Dim sObjectDN

    sObjectDN = ADObject_DN_UCG(txtObjectDN.value, sDomainDN)
    If sObjectDN = "" Then
        MsgBox "Could not find " & txtObjectDN.value
    End If

    ' Tworzenie obiektu słownika do przechowywania nazw grup
    Set oMemberOfList = CreateObject("Scripting.Dictionary")
    ' Ustawienie trybu porównania na nierozróżnianie wielkości liter
    oMemberOfList.CompareMode = vbTextCompare

    Call ADObject_MemberOf (sObjectDN)
    aMemberOfList = oMemberOfList.Keys
    aMemberOfList = Array_Sort(aMemberOfList)
    sResults = "MEMBERSHIPS"
    For Each sMemberOf In aMemberOfList
        sResults = sResults & "<br/>" & sMemberOf
```

```
    Next

    divResults.innerHTML = sResults
End Sub
' ======================
' FUNKCJE Z BIBLIOTEKI
' ======================

Sub ADObject_MemberOf(ByVal sObjectDN)
Sub ADObject_MemberOf_Enum(ByVal oObject)
Function Array_Sort(ByVal aArray)
Function ADObject_DN_UCG(ByVal sObject, sSearchDN)
Function ADObject_Find_UCG(sObjectName, sSearchDN)

</script>

<style>
   body, tr, td, table, p, input {font-family: arial; font-size: 9pt;}
</style>

</head>
<body>
<input type="text" name="txtObjectDN" id="txtObjectDN" size="20"
onchange="MainRoutine()"/>
<h3>Group memberships</h3>
<div id="divResults" name="divResults"> </div>
</body>
</html>
```

Wystarczy tylko dwukrotnie kliknąć aplikację HTA, aby ją uruchomić! W polu tekstowym zdarzenie *OnChange* jest używane do wywoływania *MainRoutine*, więc wprowadzamy nazwę użytkownika, komputera lub grupy, a następnie naciskamy Tab, aby zobaczyć do jakich grup należy obiekt.

Gdzie szukać więcej informacji

Aby dowiedzieć się więcej o aplikacjach HTA, należy zacząć od przejrzenia MSDN na stronie *http://msdn2.microsoft.com/en-us/library/ms536471.aspx*. Następnie trzeba dokładnie przestudiować witrynę HTA TechNet Scripting Center: *http://www.microsoft.com/technet/scriptcenter/hubs/htas.mspx*.

Podsumowanie rozwiązania

Do utworzenia listy członkostwa w grupach tworzymy niestandardowe narzędzie administracyjne oparte na GUI. Dokonujemy tego poprzez migrację funkcjonalnego kodu VBScript opracowanego w Rozwiązaniu 1.1 do HTA. Aby utworzyć aplikację HTA, cały kod musi być zawarty w podprogramach i funkcjach. Tworzymy oparty na WWW interfejs użytkownika i podłączamy zdarzenia do wykonania kodu. Wyniki kodu mogą być wyświetlane poprzez wprowadzenie kodu HTML do znacznika * *.

1.3. Rozszerzenie przystawki Użytkownicy i komputery Active Directory do wyliczania członkostwa w grupach

Ogólny opis rozwiązania

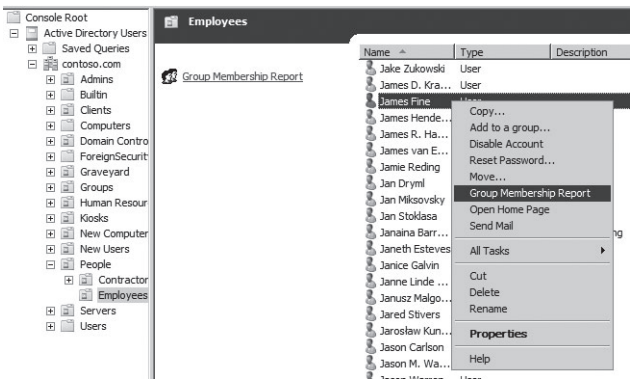
Typ rozwiązania Narzędzie (HTA)

Funkcje i narzędzia HTA, bloki zadań przystawki MMC, specyfikatory wyświetlania Active Directory

Krótki opis rozwiązania Poszerza przystawkę Użytkownicy i komputery Active Directory, tak aby można było prawym przyciskiem myszy kliknąć, użytkownika, komputer lub grupę i utworzyć raport o ich członkostwie w grupach, obejmujący grupy zagnieżdżone. Uczy, jak włączyć niestandardowe narzędzia do przystawek MMC.

Korzyści Odpowiada na pytanie: „Do jakich grup należy ____?" Dzięki temu członkostwo w grupach staje się lepiej widoczne niż przy użyciu standardowych zestawów narzędzi GUI.

Podgląd Istnieją dwie metody włączania naszej niestandardowej aplikacji do przystawki Użytkownicy i komputery Active Directory. Po wybraniu użytkownika, na lewym marginesie okienka szczegółów MMC pojawia się zadanie. Alternatywnie może pojawić się polecenie w menu kontekstowym. Zwykle będziemy stosować jedną z tych metod. Rysunek 1-8 przedstawia obie metody.



Rysunek 1-8 Integrowanie niestandardowego narzędzia z przystawką Użytkownicy i komputery Active Directory

Wprowadzenie

W rozwiązaniach 1.1 i 1.2 zaprojektowaliśmy skrypt, który tworzy listę członkostwa w grupach i przekształciliśmy ten skrypt w niezależną aplikację HTML (HTA), aby zapewnić interfejs GUI. Jednak musimy jeszcze znaleźć i uruchomić HTA oraz ręcznie wprowadzić nazwę użytkownika lub komputera. Czy nie byłoby przyjemnie wybrać użytkownika w narzędziu administracyjnym, takim jak Użytkownicy i komputery Active Directory i jednym lub dwoma kliknięciami utworzyć raport o członkostwie w grupach dla tego użytkownika? Tego właśnie dokona to rozwiązanie.

Być może ważniejsze są umiejętności, których nabędziemy i które umożliwiają integrację niestandardowych narzędzi za pomocą standardowej przystawki MMC. Przy użyciu tych narzędzi będziemy w stanie wprowadzać do MMC dowolny skrypt, aplikację HTA, plik wykonywalny, program użytkowy innej firmy lub inne narzędzie, aby tworzyć supermocny zestaw narzędzi administracyjnych.

Argumenty i aplikacje HTA

Jeśli zamierzamy teraz uruchomić naszą niestandardową aplikację HTA i oczekujemy, że w magiczny sposób utworzy ona raport o członkostwie w grupach dla wybranego użytkownika, będziemy musieli przekazać nazwę użytkownika do HTA „zakulisowo" , w postaci argumentu. HTA będzie więc wymagać możliwości odebrania tego argumentu i przetworzenia go.

Niestety, aplikacje HTA nie zawierają szczególnie eleganckiej metody zrobienia tego. Nie ma kolekcji *Arguments()*, takiej jak w VBScript. Musimy więc szybko utworzyć jakieś rozwiązanie.

Załóżmy, że próbowaliśmy uruchomić HTA z powłoki poleceń. Zastosowaliśmy następującą składnię:

```
mshta.exe "\\path-to-HTA\HTA-name.hta"
```

Jeśli chcemy przekazać argument, taki jak nazwa użytkownika, dodajemy go do polecenia:

```
mshta.exe "\\path-to-HTA\HTA-name.hta" "jfine"
```

Jak wspomniano wcześniej, HTA nie może w łatwy sposób rozpoznać argumentu *„jfine"*. Jednak aplikacja HTA zna wiersz poleceń, z którego była uruchamiana. Wiersz poleceń jest właściwością samej aplikacji, a aplikacja jest określana przez atrybut *id* elementu *<hta:application>*. A zatem, przykładowo, rozważmy aplikację, która ma następujący element *<hta:application>*:

```
<hta:application
   id="oMyHTA">
```

W tym przypadku aplikacja jest reprezentowana przez obiekt, *oMyHTA* i można wejść do jej wiersza poleceń, nadając mu właściwość *oMyHTA.commandline*, na przykład:

```
sHTACommandLine = oMyHTA.commandline
```

Wszystko, co wobec tego trzeba zrobić, to przeanalizować wiersz poleceń. Aplikacja HTA będzie wywoływana poprzez nazwę (na przykład *[mshta.exe] „path\MyHTA.hta”*), więc możemy poszukać rozszerzenia .hta i przyjąć, że cokolwiek po tym się znajduje, jest argumentem. Następnie analizujemy składnię pozostałej części rozdziału przy użyciu znaku spacji jako separatora argumentów i cudzysłowów jako kwalifikatorów, które otaczają argumenty zawierające spacje – tak jak w zwykłym wierszu poleceń.

Jeśli przyjrzymy się kodowi HTA_Arguments.snippet na towarzyszącej książce płytce, zobaczymy dwie funkcje. Pierwsza z nich, *HTA_Arguments*, jest wywoływana za pomocą odwołania do obiektu HTA. Znajduje ona wiersz poleceń, szuka rozszerzenia .hta, aby sprawdzić, czy istnieją jakieś argumenty, a następnie przegląda pozostałą część wiersza poleceń przy użyciu drugiej funkcji, *Text_Delimited_Parse()*. Ta druga funkcja jest bardzo przydatna do analizy składniowej dowolnego ustalonego tekstu, takiego jak wiersze z pliku tekstowego oddzielanego przecinkami (.csv) lub – w tym przypadku – argumentów oddzielonych spacją, gdzie argumenty zawierające spację są wyznaczone (otaczane) przez cudzysłowy. Kiedy te dwie funkcje wykonają swoją pracę, tablica zwracana przez *HTA_Arguments()* będzie zawierać wszystkie argumenty, które były przekazane do HTA w wierszu poleceń.

Wszystko zatem, co trzeba zrobić, to wywołać te funkcje przy uruchamianiu HTA. Poniższy kod, wewnątrz znacznika <*script*> naszej aplikacji HTA, wykona to zadanie:

```
Dim aArguments        ' deklarujemy to na zewnątrz naszego podprogramu,
                        ' tak aby miał on globalny zakres i był dostępny
                        ' dla innego kodu w HTA
Sub Window_Onload     ' procedura z tą nazwą będzie automatycznie uruchamiana
                        ' po rozpoczęciu wykonywania aplikacji HTA
   aArguments = HTA_Arguments(oMyHTA)
   ' Modyfikujemy oMyHTA w powyższym wywołaniu funkcji, tak aby zgadzało się
   ' z wartością atrybutu "id" elementu <hta:application>
End Sub
```

Tablica *aArguments* zawiera wszystkie argumenty. Ponieważ jest ona deklarowana z zakresem globalnym, można mieć dostęp do *aArguments* z każdej innej części kodu.

Uwaga To rozwiązanie do otrzymywania argumentów w HTA jest bardzo ***słabo*** udokumentowane. Lepiej będzie, gdy napiszemy aplikacje HTA.

Dla naszego ADObject_MemberOf_Enum.hta musimy dodać atrybut *id* do znacznika <*hta:application*>, tak jak poniżej:

```
<hta:application
   id="oMyHTA">
```

Można wtedy dodać następujący kod wewnątrz znaczników <*script*>:

```
Dim aArguments
Sub Window_Onload
   aArguments = HTA_Arguments(oMyHTA)
   If uBound(aArguments) >= 0 Then
          ' Argument został przekazany do HTA
          ' Wcześniejsze wypełnianie pola tekstowego i uruchomienie głównej procedury
```

```
            txtObjectDN.value = aArguments(0)
            Call MainRoutine()
    End If
End Sub
' włączenie HTA_Arguments.snippet, który zawiera te dwie funkcje:
Function HTA_Arguments()
Function Text_Delimited_Parse()
```

Ten kod jest uruchamiany po otwarciu HTA i bada on wiersz poleceń, aby stwierdzić, czy zostały podane jakieś argumenty. Jeśli tak jest, kod zakłada, że pierwszy argument (tablice rozpoczynają się od miejsca zerowego) jest nazwą użytkownika lub komputera, dla którego chcemy określić członkostwo w grupach. Nazwa ta zostaje najpierw wprowadzona do pola tekstowego na stronie, a potem uruchamiany jest kod, który raportuje członkostwo w grupach.

Rezultatem jest nasz finalny raport HTA o członkostwie w grupach (Group Membership Report), o nazwie MemberOf_Report.hta. Został w nim także przemycony dodatkowy podprogram, który ustawia rozmiar i centruje okna HTA, jak również etykieta objaśniająca i przycisk w części HTML aplikacji HTA. Ten dodatkowy kod można zobaczyć, otwierając HTA w Notatniku lub w innym edytorze HTA.

Zastosowanie MemberOf_Report.hta

Ta gotowa do użycia w wierszu poleceń wersja HTA nosi nazwę MemberOf_Report.hta i znajduje się w folderze Scripts na nośniku towarzyszącym książce. Uruchamiamy ją przy użyciu poniższej składni polecenia:

```
path\MemberOf_Report.hta "name of user, computer, or group"
```

gdzie nazwa użytkownika, komputera lub nazwa grupy to DN, ADsPath lub *sAMAccountName* obiektu.

Integrowanie niestardowej aplikacji HTA z zdaniami przystawki MMC

Teraz przyszedł czas na przyjemną cześć! Trzeba było długo czekać lub z mozołem przebrnąć przez mnóstwo pisania skryptów, aby dotrzeć do tej części. Warto było poczekać! Gdy to zrobimy, będziemy w stanie wybierać użytkownika, komputer lub grupę i jednym kliknięciem generować raport o członkostwie dla tego obiektu.

Mamy aplikację HTA, którą można uruchomić z wiersza poleceń. Oto pierwszy krok – uzyskanie narzędzia, które akceptuje argumenty.

Zapisywanie HTA w dostępnej lokalizacji

Aplikacja HTA (MemberOf_Report.hta z folderu Scripts na płytce towarzyszącej książce) musi być dostępna dla wszystkich administratorów, którzy będą z niej korzystać. Można przechowywać ją w udostępnianym folderze lub dystrybuować na lokalne dyski administracyjne stacji roboczych.

Jeśli zapiszemy HTA (lub inny skrypt i narzędzia) w udostępnianym folderze, prawdopodobnie napotkamy ograniczenia w zabezpieczeniach, które nie dopuszczą do uruchomienia kodu lub będą wyświetlać monity. Lepsze jest rozprowadzanie lokalnych kopii

tego narzędzia, tak jak to się robi ze standardowymi narzędziami administracyjnymi firmy Microsoft. Jeśli jednak ktoś chce zapisać narzędzia w udostępnianej lokalizacji, poniższe akapity zawierają wyjaśnienia, co trzeba zrobić.

Po pierwsze, trzeba mieć pewność, że serwer, na którym została zapisana aplikacja HTA, jest na naszym komputerze w strefie zabezpieczeń Lokalny intranet. W programie Internet Explorer klikamy Narzędzia, a następnie Opcje internetowe i zakładkę Zabezpieczenia, wybieramy Lokalny intranet i klikamy Witryny. Klikamy Zaawansowane (jeśli ten przycisk jest widoczny, co zależy od systemu operacyjnego, w którym działamy), a następnie wprowadzamy adres *\\servername.company.com*. (Trzeba się upewnić, że używamy w pełni kwalifikowanej nazwy domeny serwera). Klikamy Dodaj i wprowadzony przez nas adres zostanie przekształcony na plik formatu:*//servername.companyname.com*. Klikamy Zamknij i OK.

Jeśli teraz uruchomimy HTA, zostanie wyświetlony monit z ostrzeżeniem zabezpieczeń ADO. Aby uniknąć tego ostrzeżenia, dostosujmy ustawienia strefy zabezpieczeń Lokalnego intranetu, włączając ustawienie Dostęp do źródła danych poprzez domeny. Ma to takie konsekwencje dla zabezpieczeń, że cokolwiek robimy, osłabiamy bezpieczeństwo, trzeba więc poświęcić trochę czasu na przejrzenie źródeł internetowych, by stwierdzić, czy to chcemy zrobić. W innym przypadku trzeba pogodzić się z klikaniem ostrzeżeń zabezpieczeń ADO.

Niektóre aplikacje HTA, wraz z większością mocnych aplikacji HTA, które zostały dostarczone na płytce towarzyszącej książce, będą tworzyć tak dużo podobnych ostrzeżeń, że jeśli nie złagodzimy zabezpieczeń, użycie tego narzędzia będzie żmudne. Zależnie od kodu, który wykonuje to narzędzie, może być również konieczne złagodzenie innych ustawień w strefie Lokalnego intranetu.

Polecaną alternatywą jest upewnienie się, że HTA znajduje się na lokalnej ścieżce w systemie każdego administratora (na przykład C:\Program Files*Company*Tools\MemberOf_Report.hta). W folderze Program Files tworzymy folder, taki jak *Company*Tools i kopiujemy do tej lokalizacji skrypty, aplikacje HTA i skrypty i inne narzędzia. To najlepszy sposób, aby uniknąć najpoważniejszych ograniczeń w zabezpieczeniach dla aplikacji HTA i skryptów.

Najlepsze rozwiązanie Niestandardowe skrypty i narzędzia administracyjne należy dystrybuować do stacji roboczych administratorów, tak jak dowolne inne aplikacje.

Tworzenie bloku zadań i zadania powłoki do uruchomienia HTA

Teraz włączymy nasze narzędzie do standardowych narzędzi administracyjnych Windows. Ponieważ to narzędzie jest powiązane z użytkownikami, komputerami i grupami, sensowne jest włączenie go do przystawki Użytkownicy i komputery Active Directory. Te kroki działają z prawie każdą przystawką.

Trzeba jednak użyć tej przystawki w niestandardowej konsoli – nie można tego zrobić przy użyciu przystawki Użytkownicy i komputery Active Directory w standardowej konsoli MMC. Etapy tworzenia bloku zadań i zadań są podane w kolejnych akapitach. Jeśli potrzebnych jest więcej szczegółów dotyczących działań, można spojrzeć na funkcję Pomoc w mmc.exe.

Po pierwsze, trzeba utworzyć niestandardową konsolę z przystawką Użytkownicy i komputery Active Directory:

1. **Klikamy** Start i wybieramy polecenie Uruchom (Windows XP) lub klikamy pole Wyszukaj (Windows Vista/Windows Server 2008), wpisujemy **mmc.exe**, a następnie naciskamy Enter.
2. **Pojawia się** pusta konsola MMC. Wybieramy Plik, Dodaj/Usuń przystawkę i dodajemy przystawkę Użytkownicy i komputery Active Directory.
3. **Zapisujemy konsolę, wybierając** Plik, a potem Zapisz. Trzeba zapisać ją w udziale sieciowym, w którym jest umieszczona aplikacja HTA, lub w innym udostępnionym folderze, który jest łatwo dostępny dla administratorów.

Teraz trzeba utworzyć coś, co jest nazywane blokiem zadań:

1. Rozwijamy okienko szczegółów konsoli dla jednostki organizacyjnej (OU), która zawiera użytkowników.
2. Prawym przyciskiem klikamy jednostkę organizacyjną i wybieramy Nowy widok bloku zadań.
3. Pojawia się Kreator nowego widoku bloku zadań. Klikamy Dalej.
4. Na stronie Styl bloku zadań akceptujemy wszystkie ustawienia domyślne i klikamy Dalej.
5. Na stronie Obiekt docelowy bloku zadań wybieramy select Wybrany element drzewa i klikamy Dalej.
6. Na stronie Nazwa i opis akceptujemy lub zmieniamy domyślną nazwę oraz klikamy Dalej.
7. Na stronie Kończenie upewniamy się, czy pole wyboru zostało zaznaczone i klikamy Zakończ.

Faktycznie zakończyliśmy tworzenie bloku zadań, a drugi kreator zostaje uruchomiony, aby pomóc w tworzeniu zadania w bloku zadań:

1. **Pojawia się** Kreator nowego zadania. Klikamy Dalej.
2. **Na stronie Typ polecenia** wybieramy Polecenie powłoki i klikamy Dalej.
3. W polu Polecenie wpisujemy **mshta.exe**.
4. W polu **Parametry** piszemy „*Path to the HTA*\MemberOf_Report.hta” ze spacją po parametrze. Jeśli zapisujemy narzędzie w udostępnianym folderze, używamy w pełni kwalifikowanej nazwy domeny serwera – na przykład: **\\server01.contoso.com\admintools\memberof_report.hta**.
5. Z kursorem ustawionym za spacją klikamy strzałkę, która jest przyciskiem przeglądania.
6. Wybieramy **Nazwa logowania** w systemie starszym niż Windows 2000.

   Wybieramy Nazwa, aby wykonać to zadanie dla komputerów, albo Nazwa logowania w systemie starszym niż Windows 2000 dla użytkowników oraz grup.
7. Pole **Parametry** powinno wyglądać jak poniżej:

```
"Path to the HTA\MemberOf_Report.hta" $COL<9>
```

Dla komputera będzie to wyglądać tak:

```
"Path to the HTA\MemberOf_Report.hta" $COL<0>
```

8. W polu **Rozpocznij w**, wpisujemy „**\\path to the HTA**”. Jeśli zapisujemy narzędzie w udostępnianym folderze, używamy w pełni kwalifikowanej nazwy domeny serwera – na przykład: „**\\server01.contoso.com\admintools**”.
9. Klikamy **Dalej**.
10. Na stronie **Nazwa i opis**, w polu Nazwa zadania, wpisujemy **Group Membership Report**. Opcjonalnie wprowadzamy opis, taki jak „Display sorted list of all group memberships, including nested groups and primary group” (Wyświetlanie posortowanej listy wszystkich członkostw w grupach, łącznie z grupami zagnieżdżonymi i grupą główną).
11. Klikamy **Dalej**.
12. **Na stronie Ikona zadania** wybieramy ikonę. Istnieje kilka ikon, które przywołują „grupę”.

    Jeśli klikniemy Ikona niestandardowa, możemy wyszukać ikonę, która jest częścią pliku wykonywalnego lub jest w jednej z bibliotek ikon, takich jak C:\Windows\System32\Shell32.dll lub, w Windows Vista oraz Windows Server 2003, C:\Windows\System32\Imageres.dll.
13. Klikamy Dalej.
14. Klikamy Zakończ.

Teraz pozostał jeszcze jeden bardzo ważny krok. O tym kroku można tak łatwo zapomnieć, a jeśli się o nim zapomni, to potem tak trudno jest znaleźć sposób rozwiązania problemu, że lepiej o nim nie zapominać! Dowolna kolumna, do której odwołujemy się jako do parametru zadania *musi* być widoczna, gdyż w przeciwnym przypadku zadanie nie pojawi się, gdy wybierzemy obiekt.

1. Klikamy Widok, a następnie Dodaj/usuń kolumny.
2. Wybieramy Nazwa logowania w systemie starszym niż Windows 2000 i klikamy Dodaj.
3. Klikamy OK, aby zamknąć okno dialogowe Dodaj/usuń kolumny.
4. Zapisujemy konsolę.

Zastosowanie aplikacji HTA

Dokonaliśmy tego! Teraz wybieramy dowolnego użytkownika w jednostce organizacyjnej i klikamy łącze Group Membership Report na liście zadań, które pojawia się na lewym maginesie okienka szczegółów. Gotowe! A jeśli ktoś myśli, że to jest takie świetne, niech poczeka do następnego podrozdziału, gdzie będziemy dodawać polecenie do menu kontekstowego, które pojawia się, gdy klikamy użytkownika prawym przyciskiem.

Czy nie było to świetne? Zmusza do myślenia, prawda! O tak, w tej książce włączymy dużo innych pożytecznych rozwiązań do naszych standardowych narzędzi Windows.

A ponieważ zapisaliśmy zarówno konsolę, jak i aplikację HTA w udostępnianym folderze i budujemy zadanie przy użyciu poleceń z parametrami zawierającymi ścieżkę sieciową do HTA, każdy administrator może teraz uruchomić niestandardową konsolę MMC i użyć niestandardowej aplikacji HTA.