holger SCHWICHTENBERG

Windows Scripting lernen

Von WINDOWS SCRIPT HOST und VISUAL BASIC SCRIPT bis zur WINDOWS POWERSHELL

HANSER

Im Internet: Scripte, Windows Add-ons, Scripting-Komponenten und Editoren

www.IT-Visions.de
Dr. Holger Schwichtenberg

Schwichtenberg

Windows Scripting lernen

Holger Schwichtenberg

Windows Scripting lernen

Von Windows ScriptHost und Visual Basic Script bis zur Windows PowerShell

HANSER

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Sylvia Hasselbach
Copy editing: Petra Kienle, Fürstenfeldbruck
Herstellung: Irene Weilhart
Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München
Umschlagrealisation: Stephan Rönigk
Gesamtherstellung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-44800-1
E-Book-ISBN: 978-3-446-44944-2

Inhalt

Vorwort

Vorwort zur sechsten Auflage (2016)

Liebe Leserinnen, liebe Leser,

mittlerweile gibt es Windows 10 und Windows Server 2016. Aber den Verlag Addison-Wesley, bei dem dieses Buch zehn Jahre lang in fünf Auflagen erschienen ist, gibt es nicht mehr. Nun hat das Buch im Carl Hanser Verlag eine schöne neue Heimat gefunden.

Neu in dieser Auflage ist die Version 5.0 der Windows PowerShell. Zur PowerShell gibt es viele neue Skripting-Beispiele. Ebenso wurde das Buch auf Windows 10 und Windows Server 2016 aktualisiert. Bewusst sind aber nicht alle Bildschirmabbildungen mit diesen neuesten Betriebssystemen gemacht, da das Buch sich weiterhin als von der Betriebssystemversion unabhängiges Werk versteht, das auch Leser anspricht, die nicht die neueste Betriebssystemversion verwenden können oder wollen.

Viel Erfolg mit diesem Buch wünscht Ihnen

Dr. Holger Schwichtenberg

Essen, im Januar 2016

Vorwort zur fünften Auflage (2012)

Liebe Leserinnen, liebe Leser,

das Erscheinen von Windows 8 und Windows Server 2012 nehmen wir zum Anlass für eine erneute Aktualisierung dieses Buchs. Auch hier ist der Windows Script Host weiterhin enthalten und ein wichtiges Werkzeug für die automatisierte Systemadministration.

„Windows Scripting Lernen“ ist das letzte verbliebene Buch zum Windows Scripting Host (WSH) auf dem deutschen Markt.

Neben einigen Aktualisierungen zum WSH (insbesondere hinsichtlich der restriktiveren Vergabe administrativer Rechte in Windows 8) bietet diese 5. Auflage Ihnen vor allem mehr Inhalte zum Thema Windows PowerShell. Behandelt wird die PowerShell 3.0, die in Windows 8 und Windows Server 2012 enthalten ist und auf Windows 7 und Windows Server 2008 (inkl. R2) als Zusatz installierbar ist.

Ich danke Ihnen für Ihre Treue zu diesem Buch.

Dr. Holger Schwichtenberg

Essen, im September 2012

Vorwort zur vierten Auflage (2009)

Liebe Leserinnen, liebe Leser,

das Erscheinen von Windows 7 und Windows Server 2008 R2 nehmen wir zum Anlass für eine Aktualisierung des Buchs. Die weiterhin hohen Verkaufszahlen im Zeitalter von Vista und PowerShell zeigen, dass der Windows Scripting Host (WSH) in den Unternehmen noch aktiv genutzt wird, selbst wenn er in den letzten Jahren keine großen Veränderungen mehr erfahren hat.

Natürlich ist der Markt kleiner geworden. In Deutschland gab es einmal sechs Bücher zum WSH. Von diesen sind nur noch „Windows Scripting Lernen“ und der große Bruder „Windows Scripting“ übrig geblieben.

Mittelfristig wird die PowerShell größere Bedeutung als der WSH erlangen. In diesem Buch gebe ich Ihnen einen Ausblick auf die PowerShell 2.0. Aber dieser Ausblick ist hier bewusst kurz gewählt. In meinen Büchern „Windows Scripting (6. Auflage)“ und „Windows PowerShell 2.0 – Das Praxishandbuch“ (beide bei Addison-Wesley erschienen) gehe ich genauer auf die PowerShell ein.

Ich danke Ihnen für Ihre Treue zu diesem Buch.

Dr. Holger Schwichtenberg

Essen, im Oktober 2009

Vorwort zur dritten Auflage (2007)

Liebe Leserinnen, liebe Leser,

auch mit Erscheinen von Windows Vista und der Windows PowerShell ist der Windows Script Host (WSH) noch aktuell, und er wird es auch für die nächsten Jahre noch bleiben. Windows Vista basiert entgegen früheren Ankündigungen noch nicht auf dem .NET Framework, sondern weiterhin komplett auf dem Component Object Model (COM) und noch älteren C/C++-Techniken. Der WSH und seine COM-basierten Scripting-Komponenten sind also das primäre Instrument für die automatisierte Systemadministration unter Vista.

Die Windows PowerShell, die Microsoft als gemeinsamen Nachfolger von WSH und Windows-Kommandozeilen-Shell ansieht, ist zwar mächtig und einfach, steht aber hinsichtlich des direkt nutzbaren Funktionsumfangs (in Form der Commandlets) noch weit hinter dem WSH zurück. Noch muss man hier in vielen Fällen in die Tiefen von .NET einsteigen. Erst mit kommenden Windows-Versionen und anderen Microsoft-Produkten wird es für die PowerShell einen Funktionsumfang geben, der auch Scripting-Einsteiger anspricht.

Grund genug also, diesem meistverkauften deutschen Scripting-Buch eine neue Auflage zu spendieren. In dieser Neuauflage finden Sie neben vielen kleinen Verbesserungen neue Texte zu folgenden Themen:

- Scripting-Neuerungen in Windows Vista (Kapitel 19)
- Einführung in die Windows PowerShell (Kapitel 20)
- Prozessverwaltung per Skript, insbesondere die Kommunikation zwischen Skripten und Konsolenanwendungen (Kapitel 16)

- Mehr Beispiele zur Verwendung von Text- und XML-Dateien als Ein- und Ausgabedateiformat für Skripte (in mehreren Kapiteln)

Beim Alten geblieben ist die Website, auf der Sie sich registrieren können für die Foren, Zusatz-Downloads und den Newsletter:

http://www.Windows-Scripting.de

Weiterhin viel Spaß beim Skripten wünscht Ihnen

Dr. Holger Schwichtenberg

Essen, im Mai 2007

Vorwort zur zweiten Auflage (2004)

Dass viele Administratoren sich ein Einsteigerbuch zum WSH wünschten, war mir klar, als ich dieses Buch Ende 2002 zusammen mit meinen drei Co-Autoren geschrieben habe. Dass wir damit die Position des Marktführers unter den Scripting-Büchern in Deutschland einnehmen würden, hätte ich nicht erwartet. Natürlich freuen wir uns sehr über die positive Resonanz.

Bereits Ende 2003 ist ein korrigierter Nachdruck erschienen, in dem wir die restlichen kleinen Tippfehler in dem Buch und auf der CD beseitigt haben. Nun liegt eine überarbeitete und erweiterte zweite Auflage vor Ihnen. Komplett neu in diesem Buch sind die Kapitel 16 („Scripting der Gruppenrichtlinien“) und 17 („Sicheres Scripting“).

Herzlich bedanken möchte ich mich bei allen Lesern, die durch ihr Feedback geholfen haben, diese zweite Auflage noch besser zu machen.

Ausdrücklich hinweisen möchte ich Sie auf die Website zu diesem Buch:

http://www.Windows-Scripting.de

Aktuell bietet Ihnen diese Website folgende Informationen und Dienste:

- Umfangreiches Windows Scripting-Glossar
- FAQ zum WSH
- Diskussionsforum zum Windows Scripting (Fragen von registrierten Lesern werden von den Autoren dieses Buchs vorrangig beantwortet!)
- Verzeichnis von Scripting-Komponenten
- Klassenreferenz für die Windows Management Instrumentation (WMI)
- Feedback-Fragebogen zu diesem Buch
- Aktualisierungen zu den Skripten in diesem Buch (sofern sich technische Änderungen in Windows ergeben oder Verbesserungen von uns oder den Lesern gefunden werden)
- Skriptarchiv mit über 200 weiteren WSH-Skripten
- Scripting-News und -Newsletter
- Und last, but not least: Informationen zu unseren Scripting-Schulungen und zum Support bei Fragen rund um den WSH.

Ich wünsche Ihnen nun viel Erfolg mit diesem Buch und würde mich freuen, Sie auf meiner Website begrüßen zu dürfen!

Dr. Holger Schwichtenberg

Essen, im Mai 2004

Vorwort zur ersten Auflage (2002)

WSH

Zur automatisierten Systemadministration ist der Windows Script Host (WSH) eine sehr mächtige Alternative gegenüber der schon etwas angestaubten Windows-Batch-Programmierung. Unsere Erfahrungen aus Scripting-Schulungen und Beratungsterminen in den letzten vier Jahren haben aber gezeigt, dass es vielen Administratoren schwerfällt, sich in die Welt des Scriptings einzuarbeiten – oft auch gehemmt durch die Tatsache, dass das Scripting zum Bereich Programmierung/Softwareentwicklung gezählt wird.

Zielgruppe

Scripting ohne Vorkenntnisse

„Windows Scripting Lernen“ wendet sich an Administratoren ohne Programmierkenntnisse. Dieses Buch enthält eine schrittweise Einführung in die Entwicklung von Skripten. Auch ohne Vorerfahrung in der Programmierung lernen Sie durch dieses Buch die Möglichkeiten zur automatisierten Administration von Unternehmensnetzwerken mit dem Windows Script Host (WSH), Visual Basic Script und verschiedenen sogenannten COM-Komponenten kennen.

Methodik

Didaktischer Aufbau

Das Buch hat eine didaktische Struktur mit aufeinander aufbauenden Kapiteln. Bewusst wird darauf verzichtet, detaillierte Hintergründe sowie jede Möglichkeit und jede Option vorzustellen. Dieses Buch fokussiert auf das Wesentliche, um Ihnen einen leichten Einstieg in das Windows Scripting zu ermöglichen.

Einführung

Alle grundlegenden Konzepte der Programmierung wie Variablen, Fallunterscheidungen, Schleifen, Fehlerbehandlung und die Arbeit mit Komponenten, Klassen und Objekten werden von Grund auf eingeführt. Außerdem finden Sie eine ausführliche Erklärung zur Installation und Konfiguration der Skripte und Komponenten sowie Hinweise auf mögliche Probleme oder Fehlersituationen.

Die Beispiele sind bewusst einfach gehalten. Dennoch werden Sie lernen, alle wesentlichen Aufgaben der System- und Netzwerkadministration durch Skripte zu lösen. Der deutliche Schwerpunkt dieses Buches liegt nicht auf dem Scripting im Heimeinsatz, sondern auf dem Scripting in Unternehmensnetzwerken. Daher finden Sie hier auch Themen wie das Scripting des Active Directory, der Netzwerkkonfiguration und von Ereignisprotokollen.

Am Ende jedes Kapitels stehen Aufgaben, die Sie einsetzen können, um Ihr Wissen zu vertiefen und praktisch zu üben. Gewisse Wiederholungen sind in diesem Einsteigerbuch übrigens kein Fehler, sondern didaktische Absicht.

Wie Sie dieses Buch lesen sollten

Leseanleitung

Aufgrund des didaktischen Konzepts sollten Sie die ersten fünf Kapitel dieses Buches unbedingt sequenziell in der vorgegebenen Reihenfolge lesen. Ab Kapitel 6 werden dann verschiedene Gebiete des Scriptings aufgabenorientiert behandelt. Die Kapitel 6 bis 16 müssen Sie nicht notwendigerweise sequenziell lesen. Hier können Sie durchaus direkt zu den Kapiteln springen, die Sie besonders interessieren. Die von uns gewählte Reihenfolge beinhaltet aber eine Steigerung im Schwierigkeitsgrad.

Am Ende eines jeden Kapitels gibt es einen Aufgabenteil; die passenden Lösungen stehen zusammenhängend im Anhang, sodass das „Spicken“ etwas erschwert wird.

Weitere Unterstützung im WWW

Website

Als Leser dieses Buches haben Sie Zugriff auf einen zugangsbeschränkten Bereich der deutschen Windows Scripting-Website, die Sie unter *http://www.Windows-Scripting.de* finden. Ein Service dieser Website ist, dass Sie den Autoren dieses Buches verbliebene Fragen stellen können.

Der große Bruder „Windows Scripting“

Weiterführende Literatur

Im Buchhandel werden Sie einen „großen Bruder“ zu diesem Buch finden, der schon zwei Jahre länger auf dem Markt ist: „Windows Scripting“ geht parallel zu diesem „Windows Scripting Lernen“ bei Addison-Wesley in die dritte Auflage. Dieser Titel aus der WinTec-Reihe ist ein umfassendes Nachschlagewerk zu allen Bereichen der Skriptprogrammierung und richtet sich an Entwickler und Administratoren, die bereits Vorkenntnisse in mindestens einer Programmiersprache besitzen. Wenn Sie nach der Lektüre von „Windows Scripting Lernen“ noch Wissensdurst verspüren, dann sollten Sie zum großen Bruder greifen.

Dank

Mein herzlicher Dank gilt

- meinen Co-Autoren Sven Conrad, Thomas Gartner und Oliver Scheer, die tatkräftig mitgeholfen haben, den umfangreichen Stoff aus dem „Windows Scripting“-Buch auf die „Lernen“-Reihe herunterzubrechen,
- Ayşe Aruca und Georg Meindl für ihre kritischen Anmerkungen als Testleser dieses Buches,
- meiner Korrektorin Astrid Schürmann, die wieder einmal mit hoher Präzision nicht nur die sprachlichen Fehler aus unserem Text entfernt, sondern uns auch auf inhaltliche Inkonsistenzen hingewiesen hat,
- und meiner Lektorin Sylvia Hasselbach für ihre Geduld bei der doch langwierigen Geburt dieses Werks.

Ich wünsche Ihnen nun viel Erfolg mit diesem Buch.

Holger Schwichtenberg

Essen, im November 2002

Website für Leser

Zu diesem Buch gibt es eine eigene Website:

http://www.windows-scripting.de

Sie als Leser haben neben den öffentlichen Bereichen auch die Möglichkeit, auf einen geschützten Bereich zuzugreifen, der besondere Informationen enthält:

Downloads: die aktuellen Versionen der in diesem Buch abgedruckten Skripte sowie weitere Skripte und Codebeispiele

Verzeichnis	Inhalt
\Skripte	Alle Skripte aus dem Buch, geordnet nach Kapiteln
\Install	Erweiterungen, Komponenten, Sprachen und Tools für das Windows Scripting (zum Teil als Vollversionen, zum Teil als Demoversionen)
\Weitere Informationen	Dieses Verzeichnis enthält zusätzliche Dokumentationen zu Visual Basic Script, dem WSH und einigen der besprochenen Komponenten.
\Über den Autor	Informationen über den Autor dieses Buchs

Foren: Wenn Sie Fragen haben oder Ihre Meinung zu einem Thema dieses Buchs äußern möchten, dann können Sie hier auf Reaktionen anderer Nutzer hoffen.

Leser-Bewertung: Vergeben Sie Noten für dieses Buch und lesen Sie nach, was andere Leser von diesem Buch halten.

Bug-Report: Melden Sie hier Fehler, die Sie in diesem Buch gefunden haben! Hier können Sie auch nachlesen, welche Fehler anderen nach Drucklegung aufgefallen sind.

Newsletter: Alle registrierten Leser erhalten in unregelmäßigen Abständen einen Newsletter.

Der URL für den Zugang zum Leser-Portal lautet:

http://www.windows-scripting.de/leser

Bei der Anmeldung müssen Sie das Erstzugangskennwort Defiance Skies angeben (Defiance ist eine Science-Fiction-Serie).

Bitte beachten Sie, dass das Leser-Portal eine freiwillige, private Leistung des Autors ist, auf die es keinen Rechtsanspruch gibt.

Über den Autor
Dr. Holger Schwichtenberg

www.IT-Visions.de®
Dr. Holger Schwichtenberg

5Minds
IT-SOLUTIONS

- Studienabschluss Diplom-Wirtschaftsinformatik an der Universität Essen
- Promotion an der Universität Essen im Gebiet komponentenbasierter Softwareentwicklung
- Seit 1996 selbstständig als unabhängiger Berater, Dozent, Softwarearchitekt und Fachjournalist
- Leiter des Berater- und Dozententeams bei *www.IT-Visions.de*
- Leitung der Softwareentwicklung im Bereich Microsoft/.NET bei der 5minds IT-Solutions GmbH & Co. KG *(www.5minds.de)*
- Lehrbeauftragter an den Fachhochschulen in Münster und Graz
- 60 Fachbücher bei zahlreichen Verlagen und mehr als 800 Beiträge in Fachzeitschriften
- Gutachter in den Wettbewerbsverfahren der EU gegen Microsoft (2006 – 2009)
- Ständiger Mitarbeiter der Zeitschriften iX (seit 1999), dotnetpro (seit 2000) und Windows Developer (seit 2010) sowie beim Online-Portal heise.de (seit 2008)
- Regelmäßiger Sprecher auf nationalen und internationalen Fachkonferenzen (z. B. Microsoft TechEd, Microsoft Summit, Microsoft IT Forum, BASTA, BASTA-on-Tour, .NET Architecture Camp, Advanced Developers Conference, Developer Week, OOP, DOTNET Cologne, VS One, NRW.Conf, Net.Object Days, Windows Forum)

- Zertifikate und Auszeichnungen von Microsoft:
  - Microsoft Most Valuable Professional (MVP)
  - Microsoft Certified Solution Developer (MCSD)
- Thematische Schwerpunkte:
  - Microsoft .NET Framework, Visual Studio, C#, Visual Basic
  - .NET-Architektur/Auswahl von .NET-Technologien
  - Einführung von .NET Framework und Visual Studio/Migration auf .NET
  - Webanwendungsentwicklung mit IIS, ASP.NET und JavaScript sowie TypeScript
  - Verteilte Systeme, Webservices, Enterprise .NET
  - Relationale Datenbanken, XML, Datenzugriffsstrategien
  - Objektrelationales Mapping (ORM), insbesondere ADO.NET Entity Framework
  - Windows PowerShell (WPS) und Windows Management Instrumentation (WMI)
- Ehrenamtliche Community-Tätigkeiten:
  - Vortragender für die International .NET Association (INETA)
  - Betrieb diverser Community-Websites *www.dotnetframework.de*, *www.entwickler-lexikon.de*, *www.windows-scripting.de*, *www.aspnetdev.de* u. a.
- Weblog: *www.dotnet-doktor.de*
- Kontakt: *buero@IT-Visions.de* sowie Telefon 02 01 64 95 90-0

Hinweise für den Leser

Schreibweisen in diesem Buch

Alle Skripte sind in `nichtproportionaler Schrift` geschrieben. Damit Sie Befehle von Fachbegriffen unterscheiden können, sind alle Erwähnungen von Befehlen im Text auch mit `nichtproportionaler Schrift` ausgezeichnet.

Kursiv gesetzt sind Dateinamen und Pfadangaben sowie Bildschirmelemente wie Registerkarten, Menüeinträge und Schaltflächen.

Wichtige Hinweise und Einschübe sind mit einem grauen Kasten hinterlegt.

Zusätzlich werden vier Symbole verwendet, um Ihre Aufmerksamkeit zu wecken:

Das Achtung-Symbol warnt vor Bugs oder möglichen Schwierigkeiten.

Interessante Hintergrundinformationen werden durch das Hinweis-Symbol gekennzeichnet.

Das Website-Symbol verweist auf die Buch-Website:
www.windows-scripting.de

Dieses Symbol steht für Tipps, die Sie schneller zum Ziel bringen können oder Ihnen helfen, Schwierigkeiten zu vermeiden.

Hinweise zu den Listings

Alle Beispiele in diesem Buch sind in Visual Basic Script geschrieben und im Windows Script Host lauffähig. Als Testplattform wurden der Windows Script Host Version 5.7/5.8 und Visual Basic Script 5.7/5.8 auf Windows XP, Windows Vista, Windows 7, Windows 10, Windows Server 2003 R2, Windows Server 2008 R2, Windows Server 2012 R2 und Windows Server 2016 verwendet.

Jedes Listing beginnt mit einem Listing-Header. Darin finden Sie folgende Informationen:

- Name der Skriptdatei in den Downloads zu diesem Buch,
- Kurzbeschreibung zum Zweck des Skripts,
- verwendete Scripting-Komponenten, die zum Funktionieren des Skripts notwendig sind,
- Trennlinie

```
' Dateiname.vbs
' Beschreibung
' verwendete Komponenten: WMI, ADSI
' ================================
```

HINWEIS: Wenn Skripte bei Ihnen nicht laufen, beachten Sie folgende Hinweise:

- Stellen Sie sicher, dass Sie den WSH 5.7 oder 5.8 und VBScript 5.7 oder 5.8 installiert haben.
- Vergewissern Sie sich, dass Sie die verwendeten Komponenten in den aktuellsten in diesem Buch beschriebenen Versionen installiert haben.
- Prüfen Sie, ob es im Skript Angaben (Computernamen, Pfade, Benutzernamen) gibt, die in Ihrem Netzwerk möglicherweise nicht gültig sind.

Es ist möglich, dass Sie trotzdem Probleme haben, weil es diverse Unterversionen auf verschiedenen Windows-Versionen der Scripting-Bausteine gibt, die unterschiedliche Funktionen (und Bugs) haben. Im Zweifel stellen Sie bitte eine Frage auf der Leser-Website.

1 Einführung in den Windows Script Host (WSH)

Lernziele

Dieses Kapitel erklärt einige grundlegende Begriffe zum Windows Scripting und zeigt Ihnen schrittweise die Erstellung von zwei einfachen Skripten auf.

1.1 Der Windows Script Host

DOS-Batch

Microsoft hat die automatisierte Systemadministration lange vernachlässigt. Die Windows-Batch-Kommandozeilensprache ist nicht nur kompliziert, sondern auch nicht mächtig genug für viele Aufgaben der Systemadministration. Die Lücke wurde von Drittanbietern gefüllt, die sich aber mangels Bekanntheit und Marktmacht nicht durchsetzen konnten. Dieses fehlende Durchsetzungsvermögen hatte auch die negative Konsequenz, dass es kaum vorgefertigte Automatisierungslösungen gab.

WSH

Ende der Neunzigerjahre hat Microsoft mit dem Windows Script Host (WSH) endlich eine Alternative zur Windows-Kommandozeilenprogrammierung veröffentlicht. Der WSH ist Teil der sogenannten Active-Scripting-Architektur, zu der auch das Scripting im Internet Explorer und die Active Server Pages (ASP) im Internet Information Server (IIS) sowie zahlreiche andere Microsoft-Produkte gehören. In diesem Einsteigerbuch wird aber nur der WSH behandelt. Mehr über andere Scripting-Hosts und die Gesamtarchitektur erfahren Sie in [SCH07a].

HINWEIS: Der Windows Script Host hieß in seiner ersten Version noch Windows Scripting Host. Sie werden beide Begriffe synonym in Microsoft-Dokumentationen und im Internet finden.

1.2 Scripting versus Programmierung

Gibt es einen Unterschied zwischen den Worten Programmierung und Scripting? Ja und nein, je nach Standpunkt. Scripting ist eine Unterdisziplin der Programmierung, die sich durch folgende Eigenschaften auszeichnet:

Eigenschaften des Scriptings

- Die Sprache dient dem Ad-hoc-Gebrauch, d. h., es gibt wenige Voraussetzungen (Systemanforderungen) für ein Skript.
- Die Syntax der Sprache ist einfach zu verwenden und zu erlernen.
- Die Sprache wird interpretiert, d. h., es gibt keinen expliziten Übersetzungsvorgang der Sprache in eine andere Sprache. Die Übersetzung der Befehle der Skriptsprache in die Maschinensprache des Computers erfolgt automatisch und ständig während der Ausführung des Skripts.
- Es gibt nur ein sehr schwaches Typsystem; Sie müssen also als Skriptentwickler kaum Unterscheidungen zwischen Zeichenketten, Zahlen und Datumsangaben machen.
- Die Abstraktion von technischen Details der Maschinensprache (der Sprache des Mikroprozessors) ist sehr hoch.
- Eine Skriptsprache ist der Maschinensprache und der Computerhardware ferner als eine normale Programmiersprache und kann daher einen Computer nicht so leicht zum Absturz bringen.

Wenn wir in diesem Buch von Programmierung reden, dann meinen wir immer die Unterdisziplin Scripting. Lassen Sie sich von dem Begriff Programmierung nicht abschrecken, sondern verstehen Sie sich als Entwickler von Skripten als eine Art Programmierer. Scripting ist der beste Einstiegsweg für die Weiterentwicklung zum „richtigen“ Programmierer. Mit „richtigem Programmieren“ sei hier die Entwicklung mit Nicht-Skriptsprachen gemeint.

HINWEIS: Ein Compiler ist das Gegenstück zu einem Interpreter. Während ein Interpreter ein Programm fortlaufend und immer wieder in Maschinensprache übersetzt, führt ein Compiler die Übersetzung einmalig aus und speichert das Ergebnis. Ein interpretiertes Programm ist schneller änderbar als ein kompiliertes Programm, weil der Kompilierungsschritt entfällt.

1.3 Voraussetzungen

WSH-Installation

Der Windows Script Host (WSH) gehört zum Standardinstallationsumfang der neueren Microsoft-Betriebssysteme. Bei den älteren muss er als Erweiterung installiert werden. Inzwischen sind fünf Versionen des WSH im Umlauf:

- Version 1.0 (interne Versionsnummer 5.0)

- Version 2.0 (interne Versionsnummer 5.5)
- Version 5.6 (entspricht auch der internen Versionsnummer 5.6)
- Version 5.7 (entspricht auch der internen Versionsnummer 5.7)
- Version 5.8 (entspricht auch der internen Versionsnummer 5.8)

Microsofts Versionszählung

Diese Versionsnummern führen zu der berechtigten Frage, wie Microsoft zu solch einer Zählweise kommt. Der Grund dafür ist, dass man sich bei Microsoft oftmals nicht einigen kann, ob man bei neu eingeführten Betriebssystembestandteilen die Versionszählung des Betriebssystems übernimmt oder aber eine eigene Versionszählung beginnt. Der WSH wurde für Windows 2000 (NT 5.0) entwickelt, und das, was sich im Setup mit WSH 1.0 meldet, versteht sich intern auch als WSH 5.0. Als dann bis zur Veröffentlichung von Windows 2000 eine weitere Version des WSH fällig war, bekam diese die Nummer 2.0 (intern: 5.5). Mit der dritten Version hat Microsoft dann die interne und die externe Versionsnummer angeglichen, wobei die Versionsnummer dann nicht mehr zum Betriebssystem passte, denn Windows XP war NT 5.1.

Tabelle 1.1: Verfügbarkeit des WSH in verschiedenen Betriebssystemversionen

Betriebssystem	Verfügbarkeit des Windows Script Hosts
Windows 95	WSH nicht enthalten; WSH 1.0, 2.0 oder 5.6 können nachträglich installiert werden
Windows 98	WSH 1.0; Aktualisierung auf 2.0 oder 5.6 möglich
Windows ME	enthält WSH 2.0; Aktualisierung auf 5.6 möglich
Windows NT 4.0	WSH nicht enthalten; WSH 1.0, 2.0 oder 5.6 können nachträglich installiert werden
Windows 2000	enthält WSH 2.0; Aktualisierung auf 5.6 möglich
Windows XP	enthält WSH 5.6 (Version 5.6.6626); Aktualisierung auf Version 5.7 verfügbar im Rahmen des Updates „Windows Script 5.7“ (wird installiert mit Service Pack 3)
Windows Server 2003	enthält WSH 5.6 (Version 5.6.8515); Aktualisierung auf Version 5.7 verfügbar im Rahmen des Updates „Windows Script 5.7“ (wird installiert mit Service Pack 3)
Windows Preinstallation Environment (WinPE)	enthält WSH 5.6
Windows Vista	enthält WSH 5.7
Windows Server 2008	enthält WSH 5.7
Windows 7	enthält WSH 5.8
Windows Server 2008 R2	enthält WSH 5.8
Windows 8	enthält WSH 5.8
Windows Server 2012 und Server 2012 R2	enthält WSH 5.8
Windows 10	enthält WSH 5.8
Windows Server 2016	enthält WSH 5.8

Wenn Sie unsicher sind, welche WSH-Version auf Ihrem System installiert ist, dann betrachten Sie die Dateieigenschaften der Datei WScript.exe in Ihrem System- bzw. System32-Verzeichnis. Wenn diese Datei fehlt, ist der WSH nicht installiert. Das sollte aber in moderneren Windows-Versionen gar nicht mehr vorkommen.

Später in diesem Kapitel werden Sie noch ein Skript kennenlernen, mit dem Sie die Versionsnummer des WSH per Programmcode ermitteln können.

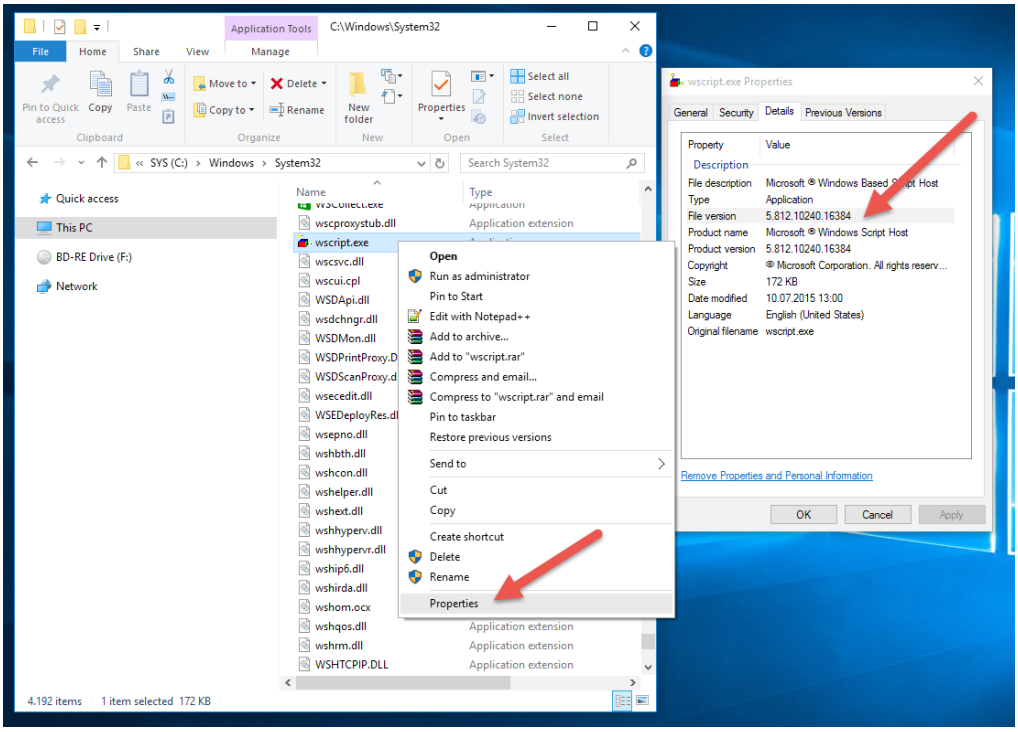

Abbildung 1.1: Ermittlung der Versionsnummer des installierten WSH (hier in Windows 10)

1.4 Die Sprache Visual Basic Script

Sprachen für den WSH

Der Windows Script Host besitzt keine eigene Programmiersprache, sondern arbeitet mit zahlreichen verschiedenen Programmiersprachen zusammen. Microsoft liefert die Sprachen Visual Basic Script (VBScript) und JScript. Von anderen Herstellern kann man Perl, REXX, Python, Haskell und Ruby in Versionen bekommen, die jeweils mit dem WSH zusammenarbeiten.

ACHTUNG: Nicht jede Programmiersprache arbeitet automatisch mit dem WSH zusammen. Eine Programmiersprache muss eine spezielle Schnittstelle zur Active-Scripting-Architektur besitzen, damit sie „WSH-fähig“ ist.

Für dieses Buch kamen nur diejenigen Sprachen infrage, die eine große Verbreitung im Bereich des Windows Scriptings haben. JScript ist eine Variante von JavaScript, das im Bereich des Scriptings in Webbrowsern marktbeherrschend ist. Im Bereich des WSH sind allerdings schätzungsweise 90% der Skripte, die man von Microsoft bekommt oder im Internet findet, in VBScript geschrieben. Dafür gibt es zwei Gründe:

- VBScript ist einfacher als JScript/JavaScript.
- VBScript ist eine Light-Version der Sprache Visual Basic, die unangefochten die am meisten verwendete Programmiersprache unter Windows ist.

Daher wählt auch dieses Buch VBScript als Programmiersprache. Die Besprechung einer weiteren Programmiersprache kommt für ein Einsteigerbuch wie dieses aus didaktischen Gründen nicht infrage. Dass sich VBScript nicht beim Browser-Scripting durchgesetzt hat, liegt übrigens daran, dass es nicht von allen Webbrowsern standardmäßig unterstützt wird.

HINWEIS: In der Microsoft-Dokumentation ist auch oftmals der Begriff Visual Basic Scripting Edition zu finden, andere gebräuchliche Abkürzungen sind VBScript und VBS.

VBScript muss nicht separat installiert werden, denn die VBScript.dll, in der die Sprache realisiert ist, wird zusammen mit dem WSH installiert. Es gibt verschiedene Versionen dieser DLL, die aktuelle ist Version 5.8.

HINWEIS: Visual Basic Script ist eine Interpreter-Sprache, dennoch meldet VBScript manchmal einen „Kompilierungsfehler“. Dieser Begriff ist nicht korrekt, es sollte besser „Skriptstruktur-Fehler“ heißen. Das bedeutet, dass Ihr Skript nicht korrekt aufgebaut ist und daher gar nicht erst gestartet werden kann.

1.5 Das erste Skript

Wenn Sie noch nie ein Skript unter Windows erstellt haben, wird Ihnen das folgende Beispiel erste Erfolgserlebnisse bereiten.

Voraussetzung

Voraussetzung für das erste Beispiel ist, dass Sie den Windows Script Host in der Version 1.0, 2.0 oder 5.6/5.7/5.8 installiert haben. Empfohlen ist, dass Sie die aktuelle Version 5.8 nutzen. Diese ist seit Windows 7 und Windows Server 2008 schon enthalten. Auf älteren Betriebssystemen wird WSH 5.7 empfohlen (siehe Downloads zu diesem Buch [\Install\WSH VBScript JScript\WSH 5.7]). WSH 5.8 ist auf älteren Betriebssystemen nicht verfügbar.

Das nachfolgende Skript befindet sich – wie alle Skripte in diesem Buch – natürlich in den Downloads zu diesem Buch im Verzeichnis /Skripte. Das /Skripte-Verzeichnis ist nach Kapiteln in Unterverzeichnisse unterteilt, damit man die Beispiele schneller finden kann.

Bei diesem ersten Skript sollten Sie sich aber durchaus die Mühe machen, die Skriptdateien selbst zu erstellen.

Ihr erstes Skript

So erstellen Sie Ihr erstes Skript für den WSH in der Sprache Visual Basic Script:

- Legen Sie eine Textdatei an, indem Sie irgendwo auf dem Desktop oder in einem Verzeichnis im Dateisystem im Kontextmenü Neu|Textdatei wählen. Es erscheint eine Datei Neue Textdatei.txt.
- Benennen Sie die Datei in ErstesSkript.vbs um. Bestätigen Sie die Nachfrage des Betriebssystems, ob die Dateinamenerweiterung wirklich geändert werden soll.
- Wählen Sie aus dem Kontextmenü der Datei Bearbeiten, sodass sich der Windows-Editor „Notepad“ öffnet. (Sofern Sie einen anderen Editor installiert haben, mag jetzt dieser gestartet werden.)
- Geben Sie Folgendes in die erste Zeile ein:

```
WScript.Echo "Ab heute kann ich skripten!"
```

  Bitte beachten Sie das Leerzeichen (Leertaste) nach Echo und die Anführungszeichen (").
- Speichern Sie die Änderungen ab. Sie können den Editor schließen, müssen es aber nicht.
- Doppelklicken Sie auf die Datei ErstesSkript.vbs. Wenn Sie alles richtig gemacht haben und das System Ihnen wohlgesonnen ist, wird das nachstehend abgebildete Dialogfenster erscheinen.

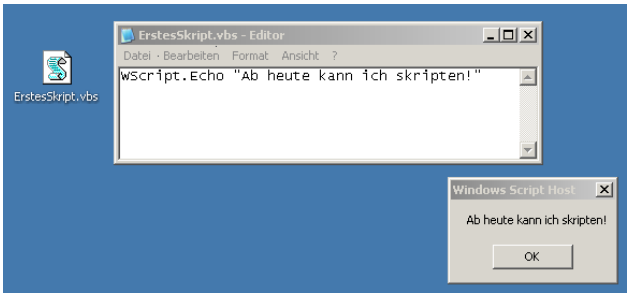

Abbildung 1.2: Skriptdatei, Quellcode und Ausgabe des ersten Skripts in Windows XP

In den neusten Windows-Betriebssystemen sieht die Anzeige nur leicht anders aus. Sofern Sie dieses einfache Skript von Ihrem lokalen Desktop ausführen, ist die Funktionsweise gleich. Die ab Windows Vista enthaltene erhöhte Sicherheit verhindert jedoch, dass Skripte von Netzwerklaufwerken und Skripte, die Veränderungen im System vornehmen, ohne Weiteres ausgeführt werden können. Bitte lesen Sie dazu das Kapitel 19.2.

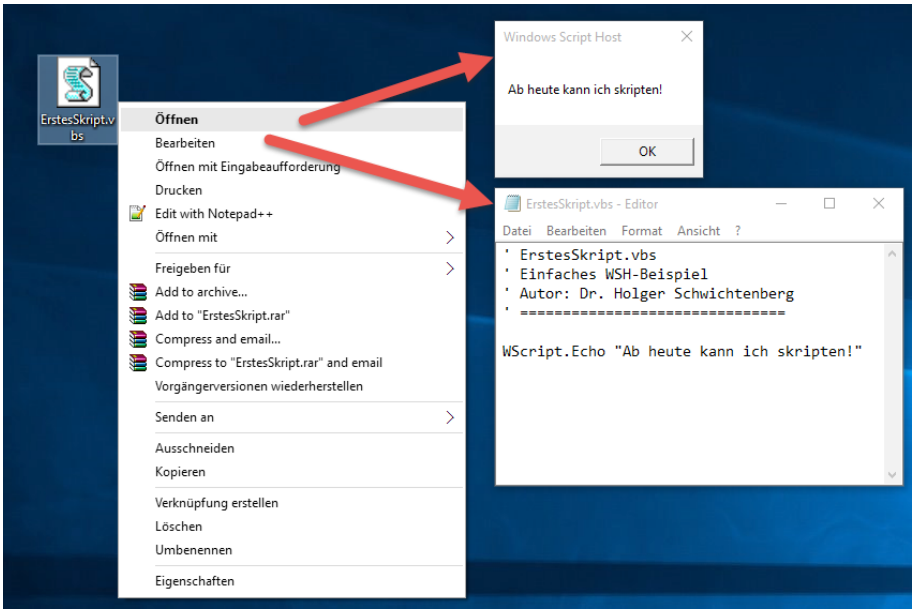

Abbildung 1.3: Skriptdatei, Quellcode und Ausgabe des ersten Skripts in Windows 10

Bildschirmausgabe

Das Skript besteht nur aus einem einzigen Befehl: WScript.Echo. Der Punkt trennt dabei das Objekt WScript und die Operation (alias „Methode“) Echo. Die Bedeutung des Punkts zwischen den Wörtern WScript und Echo wird später in diesem Buch noch näher erläutert. Nach dem Befehl folgt ein Leerzeichen, um den Befehl von seinen Parametern zu trennen. Danach folgt ein Parameter, in diesem Fall eine Zeichenkette. WScript.Echo erzeugt eine Ausgabe auf dem Bildschirm. Der Parameter gibt an, was ausgegeben werden soll.

TIPP: Die Groß- und Kleinschreibung der Befehle ist übrigens irrelevant. Visual Basic Script unterscheidet (im Gegensatz zu einigen anderen Programmiersprachen) nicht zwischen einem Groß- und einem Kleinbuchstaben. Bezüglich der auszugebenden Meldung ist die Unterscheidung natürlich wichtig, weil der WSH die Meldung genau so anzeigt, wie Sie sie eingegeben haben.

In diesem Buch werden Sie an einigen Skripten den Hinweis finden, dass die Groß- und Kleinschreibung doch relevant ist: Eine Komponente des Windows-Betriebssystems verlangt, dass die Begriffe „WinNT“ und „LDAP“ genau so und nicht anders im Skriptcode verwendet werden.

1.6 Scripting im Kommandozeilenfenster

Der WSH ist genau genommen nicht nur ein Scripting Host, sondern er umfasst zwei eng verwandte Scripting Hosts: WScript und CScript. Beide Scripting Hosts sind hinsichtlich ihres Befehlsumfangs fast identisch. Sie unterscheiden sich lediglich darin, wohin die Ausgaben gehen:

WSH für Windows

- Bei WScript (WScript.exe) erfolgt die Ausführung als Windows-Anwendung. Alle Ausgaben werden in Form von Dialogfenstern dargestellt. Wenn das Skript viele Ausgaben macht, kann dies sehr lästig sein, da jedes Dialogfenster einzeln bestätigt werden muss. Zudem ist jede Dialogbox modal: Das Skript hält an und wartet auf die Bestätigung. WScript eignet sich also für die unbeaufsichtigte Ausführung nur dann, wenn das Skript keine Ausgaben macht. Gut geeignet ist WScript jedoch, wenn der Benutzer über jeden einzelnen Schritt informiert werden und dabei die jeweils erfolgten Veränderungen überprüfen möchte (beispielsweise beim Debugging, also bei der Fehlersuche).

WSH für die Kommandozeile

- Bei CScript (implementiert in CScript.exe) erfolgt die Ausführung des Skripts im Kontext einer Kommandozeile (auch Konsole oder „DOS-Box“ im Administratorenjargon). Die Form der Ausgabe hängt von den verwendeten Ausgabebefehlen ab: Alle Ausgaben über die Methode WScript.Echo erfolgen in das Konsolenfenster. Alle Ausgaben über die spracheigenen Ausgabemethoden (z. B. MsgBox() in VBScript) werden weiterhin als modale Dialogfenster dargestellt. Ein Vorteil von CScript ist, dass es mit der Methode WScript.StdIn.ReadLine das Einlesen von Eingaben des Benutzers im Kommandozeilenfenster unterstützt. Ausgaben können mit dem Kommandozeilenbefehl für Umleitungen (>) in eine Textdatei oder an einen Drucker umgeleitet werden.

Wenn Sie nicht vorher Veränderungen an Ihrem System vorgenommen haben, dann wurde das erste Beispiel mit WScript gestartet. Denn WScript ist auf einem „normalen“ Windows die Standardeinstellung.

Um das Skript mit CScript zu starten, gehen Sie wie folgt vor:

- Öffnen Sie die Windows-Eingabeaufforderung.
- Tippen Sie „cscript “ ein. Bitte beachten Sie das Leerzeichen (Leertaste) nach dem Befehl!
- Geben Sie dahinter den Pfad zu Ihrem Skript ein, z. B. „b:\code“. Wenn der Pfad Leerzeichen enthält, müssen Sie ihn in Anführungszeichen (") setzen.
- Als Gesamtbefehl erhalten Sie dann:

```
cscript "b:\code\ErstesSkript.vbs"
```

- Führen Sie den Befehl durch Drücken der Eingabetaste (Enter) aus.

Sie werden feststellen, dass die Ausgabe nun in das Kommandozeilenfenster geht.

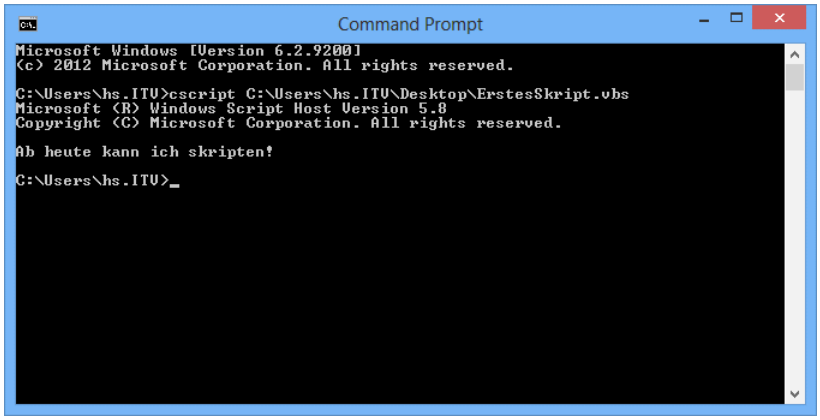

Abbildung 1.4: Ausführung des ersten Skripts mit CScript im Kommandozeilenfenster

TIPP: Sie haben das Skript auf dem Desktop erstellt und das Eintippen des (langen) Pfads zu dem Skript finden Sie sehr lästig? Sie müssen den Pfad nicht eingeben, denn die Windows-Eingabeaufforderung fügt automatisch den Pfad ein, wenn Sie eine Datei per Ziehen & Fallenlassen (Drag & Drop) auf das Kommandozeilenfenster ziehen. Sie tippen also nur „cscript“ gefolgt von einem Leerzeichen (**Leertaste**). Danach ziehen Sie die Datei ErstesSkript.vbs mit der Maus auf das Kommandozeilenfenster und lassen die linke Maustaste los. Danach müssen Sie nur noch die Eingabetaste (**Enter**) drücken.

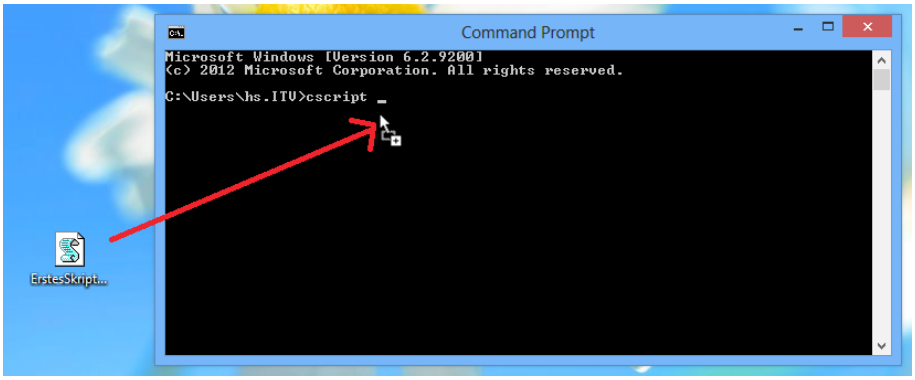

Abbildung 1.5: Ziehen & Fallenlassen einer Skriptdatei in das Konsolenfenster (hier in Windows 8)

ACHTUNG: Leider funktioniert der Ziehen & Fallenlassen-Trick in Windows Vista und Windows 7 nicht, wenn das Konsolenfenster und der Windows-Desktop mit verschiedenen Rechten laufen, z. B. der Desktop unter den Einschränkungen der Benutzerkontensteuerung (siehe Kapitel 6) und das Konsolenfenster mit vollen administrativen Rechten. Hier kann man nur Pfade über den Menüpunkt Bearbeiten/Einfügen einsetzen, wenn man vorher den Pfad (nicht die Datei!) in der Zwischenablage hat.

TIPP: Wenn Sie Gefallen daran gefunden haben, dass Ihre Skripte im Kommandozeilenfenster ausgeführt werden, dann ist Ihnen vielleicht auch das Eintippen von „CScript“ mit der Zeit lästig und Sie möchten CScript zu Ihrem Standard-Scripting-Host machen. Das ist möglich, indem Sie an der Kommandozeile einmalig folgenden Befehl eingeben:

```
cscript //H:cscript (Enter)
```

Standard-Script-Host

Danach werden alle Skripte automatisch mit CScript gestartet. Wenn Sie eine Skriptdatei per Doppelklick im Windows Explorer starten, werden Sie sehen, dass sich ein Kommandozeilenfenster öffnet. Dieses Kommandozeilenfenster schließt sich aber auch sofort nach Skriptende wieder. Wenn Sie die Ausgaben betrachten wollen, müssen Sie

- entweder vorher selbst eine Windows-Eingabeaufforderung öffnen und dort das Skript starten (hier brauchen Sie dann kein einleitendes „cscript“ mehr, es reicht der Pfad zum Skript),
- oder Sie stellen den Befehl MsgBox("Ende") an das Ende des Skripts. Dann bekommen Sie nach Ablauf des Skripts auf jeden Fall ein Dialogfenster. Das Kommandozeilenfenster ist dann so lange sichtbar, bis Sie das Dialogfenster bestätigt haben.

Die Einstellung von CScript als Standard-Scripting-Host können Sie auch rückgängig machen:

```
cscript //H:wscript (Enter)
```

TIPP: Wenn Sie die Benutzerkontensteuerung eingeschaltet haben, müssen Sie dafür die Eingabeaufforderung als Administrator starten, da Sie sonst nicht die Rechte haben, die Registrierungsdatenbank zu verändern, wo diese Einstellung abgelegt wird.

TIPP: In Windows 10 und Windows Server 2016 kann die Eingabeaufforderung endlich die Zwischenablagefunktionen Kopieren und Einfügen mit den Tastenkombinationen **STRG+C** und **STRG+V** ausführen! Microsoft hat die conhost.exe erweitert, die sowohl die Basis für die klassische Eingabeaufforderung (cmd.exe) als auch die Windows PowerShell (PowerShell.exe) ist. Weiterhin gibt es aber weder Ausschneiden (**STRG+X**) noch Kopieren und

Einfügen über Kontextmenüs, wie man es aus anderen Windows-Anwendungen kennt. Das alte Verfahren zum Kopieren und Einfügen über Markieren und EINGABE-Taste (Kopieren) bzw. rechter Mausklick (Einfügen) geht weiterhin auch in Windows 10.

1.7 Das zweite Skript: Versionsnummern ermitteln

Das zweite Skript dient dazu, die Versionsnummer des installierten WSH und der installierten Sprachversion von Visual Basic Script zu ermitteln.

Listing 1.1: Ermittlung der Versionsnummern
(Dateiname in den Downloads: /Skripte/Kapitel01/wsh-versionsnummer.vbs)

```
' wsh-versionsnummer.vbs
' Ausgabe der Versionsnummern des WSH und von VBScript
' verwendete Komponenten: WSH, VBS
' ================================

WScript.Echo _
"Dies ist der " & WScript.Name & _
" Version " & WScript.Version

WScript.Echo _
"Dies ist die Sprache " & ScriptEngine & _
" Version " & _
ScriptEngineMajorVersion & "." & _
ScriptEngineMinorVersion & "." & _
ScriptEngineBuildVersion
```

Kommentare

Geben Sie dieses Skript genauso ein wie das erste. Die ersten vier Zeilen müssen Sie nicht mit eingeben; sie sind nur Kommentare. Sie werden diese vier Kommentarzeilen vor jedem Skript in diesem Buch finden. Die Zeilen sagen Ihnen nicht nur etwas über das Anwendungsgebiet des Skripts, sondern nennen Ihnen auch den Namen des Skripts in den Downloads zu diesem Buch und welche Installationen Sie benötigen, damit das Skript ablaufen kann.

Kaufmännisches Und (&)

In dem Skript kommen neben dem WScript.Echo-Befehl auch verschiedene andere Befehle vor, die Informationen über die Versionsnummer liefern. Das kaufmännische Und (&) verknüpft dabei die Ausgabetexte mit diesen Befehlen und erzeugt eine gemeinsame Ausgabezeichenkette.

Unterstriche

Achten Sie aber auf die Unterstriche (_) am Zeilenende: Diese sind notwendig, um den Zusammenhalt der Zeile zu definieren. Das Skript besteht eigentlich nur aus zwei Befehlen und jeder Befehl muss in genau einer Zeile stehen. Mit dem Unterstrich kann man eine Zeile aber aufspalten und dies ist hier sinnvoll, weil man in einem Buch nicht so viel in eine Zeile bekommt wie auf dem Bildschirm im Notepad. Ohne die Unterstriche

machen Einsteiger an dieser Stelle oft den Fehler, dass sie etwas im Editor in zwei Zeilen tippen, weil es im Buch aus satztechnischen Gründen in zwei Zeilen steht. Um dieses Problem zu vermeiden, sind die Zeilen hier durch die Unterstriche aufgetrennt.

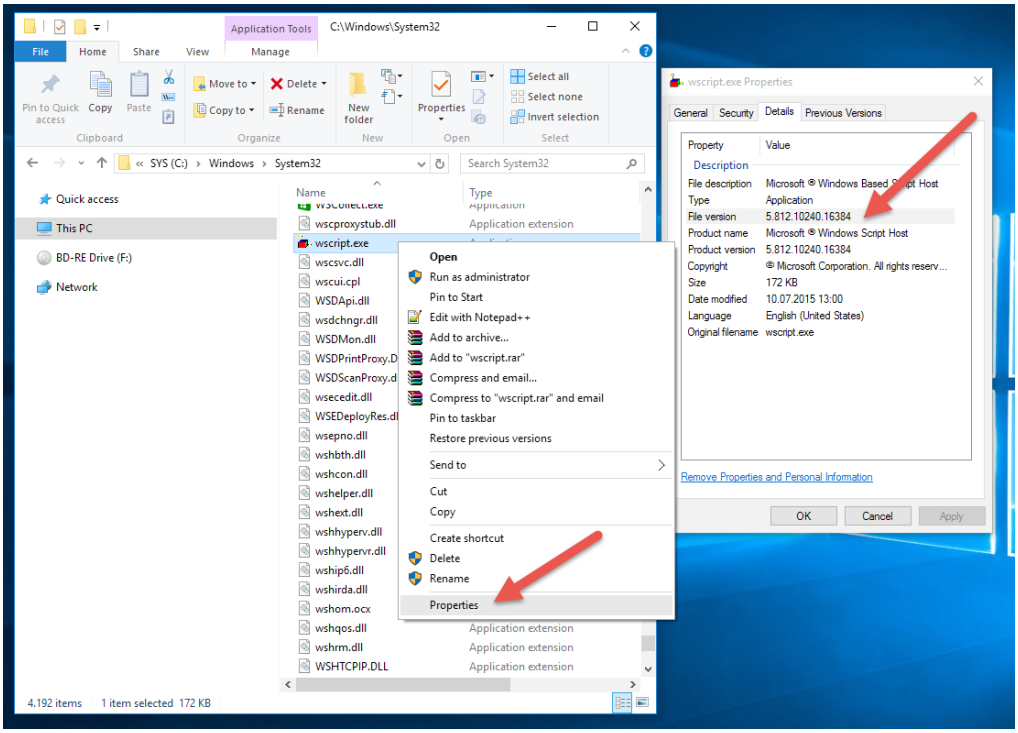

Abbildung 6.6: Ausgabe des Versionsnummern-Skripts (hier in Windows 10)

1.8 Ein Wort zur Sicherheit

Windows-Sicherheit

Eine häufig gestellte Frage ist, welche Aktionen man per Skript ausführen kann. Grundsätzlich gilt: Die Windows-Sicherheit wirkt natürlich auch für Skripte, d. h., ein Benutzer kann per Skript nur die Aktionen ausführen, die er mittels eines geeigneten Werkzeugs auch von der grafischen Benutzerschnittstelle ausführen könnte.

In diesem Buch finden Sie sowohl Aktionen, die unter normalen Rechte-Einstellungen jeder Benutzer ausführen kann (z. B. Dateien beschreiben, Inhalte eines Ordners auflisten, Netzwerklaufwerk verbinden), als auch Aktionen, die Administratoren vorbehalten sind (Benutzer anlegen, Netzwerkkonfiguration ändern).

ACHTUNG: Wenn einige der Skripte in diesem Buch nicht funktionieren, sollten Sie zunächst prüfen, ob Sie Sicherheitsbeschränkungen (z. B. durch System- oder Gruppenrichtlinien) unterliegen, die diese Aktion verbieten.

ACHTUNG: Windows Vista warnt beim Starten und Bearbeiten von Skripten (bei WSH-Skripten ebenso wie bei Skripten in HTML-Anwendungen) von einem Netzlaufwerk und fordert zur expliziten Bestätigung des Skriptstarts auf. Die Warnung erscheint sowohl beim Start an der Windows-Oberfläche als auch beim Start von der Kommandozeile; sie erscheint jedoch nicht bei Skripten, die im lokalen Dateisystem liegen. Dieses Verhalten ist mit Windows Server 2008 und Windows 7 dann wieder abgeschafft worden.

1.9 Wie geht es weiter?

Was Sie wissen müssen

Damit Sie Funktionen des Betriebssystems skripten können, benötigen Sie Wissen in drei Bereichen:

- Editoren, mit denen Sie Befehle eingeben können
- Befehle der Skriptsprache VBScript. VBScript selbst stellt nur allgemeine Befehle zum Programmablauf bereit. Befehle zum Zugriff auf Betriebssystemfunktionen gibt es hier nicht.
- Befehle für den Zugriff auf das Betriebssystem. Diese Befehle werden in sogenannten Klassen bereitgestellt, die in Form von sogenannten Komponenten veröffentlicht werden.

In welcher Reihenfolge Sie es lernen

Sie haben in diesem Kapitel den Windows-Editor „Notepad“ verwendet. Sie möchten aber vielleicht einen besseren Editor einsetzen, um mehr Komfort zu haben. Daher folgt zunächst eine Vorstellung verschiedener Editoren.

Um Klassen nutzen zu können, braucht man Programmierbefehle aus VBScript. Deshalb folgt logischerweise in Kapitel 3 zunächst die schrittweise Einführung in die Befehle von VBScript.

Das vierte und das fünfte Kapitel geben Ihnen einen Überblick über die oben erwähnten Objekte und Komponenten.

Ab Kapitel 6 ist das Buch dann aufgabenorientiert aufgebaut: Sie lernen nacheinander verschiedene Gebiete des Scriptings kennen und werden mit zahlreichen Beispielen versorgt.

HINWEIS: Sie werden wahrscheinlich feststellen, dass dieses Buch nicht alle Aufgaben enthält, die Sie gerne per Skript ausführen möchten. Das hat zwei Gründe:

- Dies ist ein Einsteigerbuch, das darauf fokussiert, Ihnen einen Einstieg in das Windows Scripting zu vermitteln. Dieses Buch kann und will nicht vollständig sein.
- Auch wenn Microsoft inzwischen zahlreiche Scripting-Möglichkeiten bietet, so gibt es immer noch unzählige Funktionen des Betriebssystems, die entweder überhaupt nicht oder nur unter erheblichem zusätzlichem Programmieraufwand in einer professionellen Programmiersprache wie C++ gelöst werden können.

Wenn Sie wissen möchten, ob eine Funktion „skriptbar“ (per Skript steuerbar) ist, sei Ihnen ein Blick in das große „Windows Scripting“-Buch [SCH07a] oder das Microsoft TechNet Script Centers [MST02] empfohlen.

1.10 Fragen und Aufgaben

Nehmen Sie sich bitte etwas Zeit, um die nachfolgenden Fragen zu beantworten. Sie helfen Ihnen, das Wissen aus diesem Kapitel zu wiederholen und praktisch zu üben. Die richtigen Antworten bzw. Musterlösungen finden Sie im Anhang.

1. Wodurch unterscheiden sich der Windows Scripting Host und der Windows Script Host?
2. Wird der WSH 5.8 automatisch mit Windows 10 installiert?
3. Wird Visual Basic Script nur deshalb sehr häufig beim Scripting verwendet, weil es Compiler-Sprache ist?
4. Was ist an diesem Befehl falsch?

```
WScript.Echo Keine Tippfehler machen!"
```

5. Was ist der wesentliche Unterschied zwischen WScript und CScript?
6. Wie erreicht man, dass alle Skripte im Kommandozeilenfenster gestartet werden?
7. Kann sich ein Befehl über mehrere Zeilen erstrecken?
8. Schreiben Sie ein Skript, das ausgibt: „Am Ende dieser Zeile steht die Versionsnummer des installierten WSH: x.x“ (wobei x.x durch die jeweilige Versionsnummer ersetzt wird).

2 Scripting-Werkzeuge

Lernziele

Die Effizienz und der Spaß bei der Entwicklung von Skripten hängen wesentlich davon ab, wie einfach und komfortabel die Erfassung der Skripte und die Fehlersuche in den Skripten sind. In diesem Kapitel stellen wir drei Editoren (das Windows-Werkzeug Notepad sowie die kostenpflichtigen Drittanbieterprodukte PrimalScript und SystemScripter) und den Microsoft Script Debugger zur Fehlersuche vor. PrimalScript und SystemScripter sind leider keine zufällige Auswahl aus einer umfangreichen Produktpalette. Nein, diese beiden Editoren sind die einzige Wahl für alle, die regelmäßig Skripte entwickeln. Alle anderen Editoren auf dem Markt bieten keine erwähnenswerten Funktionen für das Windows Scripting mit dem WSH!

Microsoft lässt die Skriptentwickler im Stich

Auch wenn Produktnamen und Marketing aus Redmond etwas anderes suggerieren: Es gibt keinen WSH-Skripteditor von Microsoft. Der in Microsoft-Office-Produkte integrierte Microsoft Script Editor ist nur eine Entwicklungsumgebung für Skripte in HTML-Seiten. Gleiches gilt für die Entwicklungsumgebung Visual InterDev 6.0, die sich zwar mit ein paar Tricks zu etwas Unterstützung für den WSH hinreißen lässt (vgl. [SCH07a]), aber dennoch weit davon entfernt ist, die Wünsche des Skriptentwicklers zu erfüllen. Zudem hat Microsoft dieses Tool seit Jahren nicht mehr weiterentwickelt.

Kein WSH-Skripteditor aus Redmond

Der Umgang mit den Editoren Notepad, PrimalScript und SystemScripter wird anhand des folgenden Skripts kurz beschrieben. In diesem Beispiel werden die zwei Konstanten HALLO und WELT in der Zeichenfolgevariablen Ausgabe zusammengeführt. Diese Ausgabe wird dann mithilfe eines Dialogfensters (MsgBox) ausgegeben.

Listing 2.1: HalloWelt.vbs

```
' HalloWelt.vbs
' Hallo Welt Beispiel
' ==============================

Const HALLO = "Hallo"
Const WELT = "Welt"
Dim Ausgabe
' Zusammensetzen des Ausgabetexts
```

```
Ausgabe = HALLO & " " & WELT

' Die eigentliche Ausgabe
MsgBox Ausgabe
```

Das Skript ist bewusst sehr einfach gehalten, da der Fokus dieses Kapitels auf den Editor-Funktionen liegt. Schwierigere Skripte lernen Sie in den folgenden Kapiteln kennen.

2.1 Nur zur Not: Notepad

Nur zur Not

Der „ultimative“ Editor „Visual“ Notepad ist auch in der aktuellen Version immer noch ungeschlagen einfach und unkonventionell. Er „brilliert“ mit Standardfunktionen wie Suchen, Ersetzen und Gehe zu. Dabei lässt er doch geschickt alle wesentlichen Wünsche an eine Skriptentwicklungsumgebung offen. Aber Sie wissen ja, warum der Windows-Editor „Notepad“ heißt? Zur Not ist ein gutes Pad …

Fangen wir also mit dem einen (einzigen) Vorteil an, den Notepad bietet: Er ist auf allen Windows-Installationen vorhanden. Egal, an welchem PC man gerade arbeitet, man kann sich darauf verlassen, dass Notepad da ist. Das ist in vielen (Not-)Situationen ein nicht zu unterschätzender Vorteil!

Nun zu dem Rest: Notepad unterstützt nicht einmal Standardfunktionen normaler Editoren wie Zeilennummern und Mehrfachfenster. Kenner älterer Windows-Versionen wissen, dass es sich bei der Funktion Gehe zu – zum Springen in eine bestimmte Zeile – um die Luxusfunktion von Notepad schlechthin handelt. Nach speziellen WSH-Funktionen wie der farblichen Markierung der VBScript-Befehle und der direkten Ausführung der Skripte aus der Entwicklungsumgebung heraus braucht man unter diesen Vorzeichen im Notepad gar nicht erst zu suchen.

Die folgende Bildschirmabbildung zeigt denn auch, wie wenig ein Skriptentwickler von Notepad erwarten kann. WSH-Skriptdateien bieten im Kontextmenü den Eintrag Bearbeiten, der direkt in den Notepad führt.

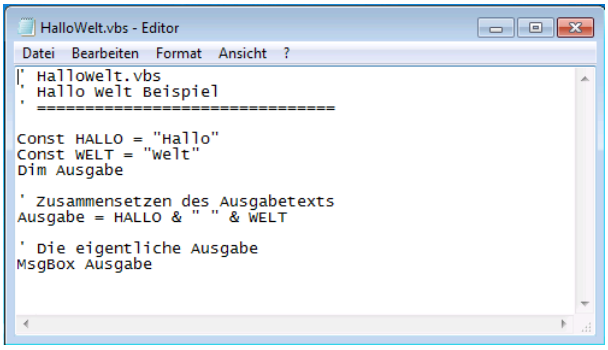

Abbildung 2.1: Das Hallo Welt-Skript im Notepad

Starten

Um das Skript nach dem Bearbeiten zu starten, muss man mühevoll den Weg über die Kommandozeile oder über eine neue Funktion im Kontextmenü Senden an beschreiten.

Über die Kommandozeile muss die Datei als Parameter der entsprechenden WSH-Variante (wscript.exe oder cscript.exe) angegeben werden, um von diesem Host ausgeführt zu werden.

Abbildung 2.2: Starten eines Skripts über die Kommandozeile

Fehlersuche

Das Behandeln von Fehlern mithilfe des Notepad erweist sich insofern als sehr aufwendig, als es keine direkte Verbindung zwischen Editor-Funktion und dem WSH gibt. Skripte können nicht direkt aus dem Editor gestartet werden; daher erfährt der Editor auch nichts von den Fehlern und kann nicht an die Fehlerstelle springen. Fehlermeldungen, die in der Kommandozeile oder über ein Dialogfenster ausgegeben werden, müssen genau untersucht werden. Die dort angegebene Zeilennummer lässt sich anschließend innerhalb des Notepads dazu verwenden, zumindest in die Zeile zu springen (Bearbeiten|Gehe zu …), die den Fehler verursacht hat.

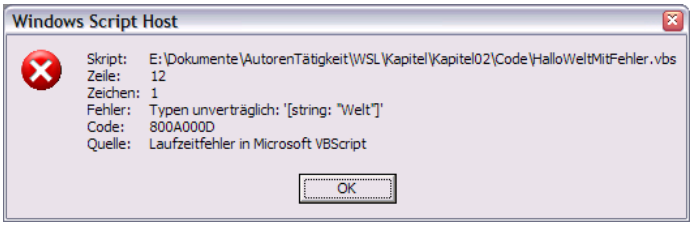

Abbildung 2.3: Mögliche Fehlermeldung eines VBScripts

Da sich Microsoft seit Langem beharrlich weigert, mehr Funktionalität in den Notepad einzubauen, werden in den folgenden Unterkapiteln zwei kommerzielle Alternativen besprochen. Wie bereits eingangs dieses Kapitels gesagt: PrimalScript und SystemScripter sind die beiden einzigen echten WSH-Skripteditoren.

HINWEIS: Wenn Sie keinen speziellen WSH-fähigen Editoren erwerben wollen, aber dennoch etwas mehr Komfort als beim Notepad wünschen, können Sie natürlich auch jeden anderen Texteditor anstelle des Notepads einsetzen. Es existieren zahlreiche kommerzielle und nichtkommerzielle Texteditoren. Einige Beispiele dafür nennt die nachfolgende Tabelle.

Tabelle 7.1: Eine Auswahl von Texteditoren

Editor	Hersteller	Bezugsquelle
Atom	Freeware	*https://atom.io/*
Eclipse	Eclipse Foundation	*http://www.eclipse.org*
EditPlus	ES-Computing	*http://www.editplus.com*
Multiedit	American Cybernetics	*http://www.multiedit.com*
Notepad++	Freeware	*https://notepad-plus-plus.org/*
Scintilla	Freeware	*http://www.scintilla.org*
Textpad	Helios Software Solutions	*http://www.textpad.com*
UltraEdit	IDM Computer Solutions	*http://www.ultraedit.com*
VIM	Freeware	*http://www.vim.org*

2.2 Einer für alles: PrimalScript

Sapien

Bei PrimalScript der Firma Sapien handelt es sich um einen erstklassigen, aber kommerziellen Universal-Editor, der einige spezielle Funktionen für die Entwicklung von WSH-Skripten bietet. PrimalScript (aktuelle Version: „2014“) unterstützt nicht nur direkt den Windows Script Host (WSH) und sogenannte Windows Script Components (WSC), sondern auch eine Vielzahl weiterer Sprachen und Scripting-Umgebungen, wie z. B. Windows PowerShell, REXX, HTML, ASP, XML, Python, Tcl, lotusScript, Batch, KiXtart, C#, Java, SQL, JavaScript etc. Allerdings hat dieser große Funktionsumfang auch seinen Preis (389 Dollar).

PrimalScript bietet sehr umfangreiche Funktionen an, die den Entwickler bei seiner Arbeit unterstützen. Dazu gehört u. a. eine Übersicht vieler Sprachkonstrukte, die sich als Vorlage direkt in das eigene Skript übertragen lassen (vgl. den linken Bereich der folgenden Abbildung). Zusätzlich verfügt PrimalScript über Syntax Coloring (automatische Farbunterscheidung), umfangreiche integrierte Hilfefunktionalitäten und Eingabehilfe. Trotz des sehr breiten Leistungsumfangs benötigt PrimalScript 6 MB auf der Festplatte und ist somit sehr schlank.

Installation/ Bezugsquelle

Nach dem Download der selbstentpackenden Installationsdatei von PrimalScript über die Website von Sapien *(http://www.sapien.com/)* führt diese automatisch durch ein Setup, das den Editor vollständig einsatzbereit macht. Anschließend befindet sich ein entsprechender Eintrag im Startmenü.

Beim Anlegen einer neuen Datei hat man die Qual der Wahl des Skript- bzw. Anwendungsformats. Wie bereits erwähnt unterstützt PrimalScript noch weitere Skriptsprachen, in unserem Fall wählt man allerdings VBScript. Abhängig von der gewählten Sprache bietet PrimalScript angepasste Eingabe- und Syntaxhilfen an.

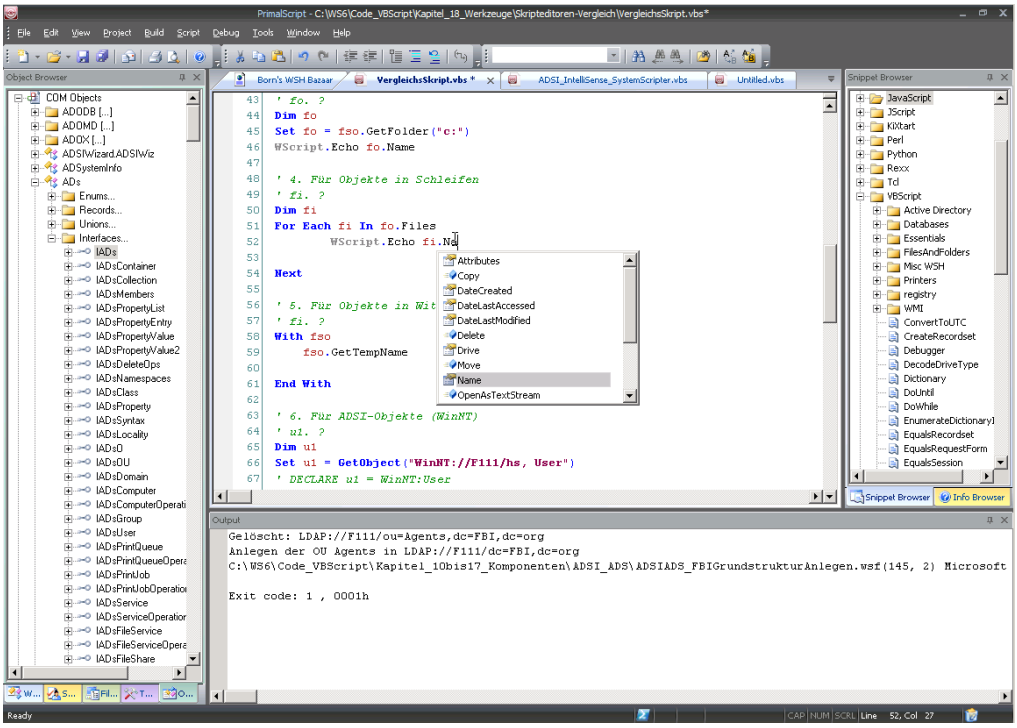

Abbildung 2.4: Editieren eines Skripts mit PrimalScript

Eingabehilfen

PrimalScript verfügt über verschiedene Hilfefunktionen. Über ein spezielles Fenster, das als Nexus bezeichnet wird (vgl. obige Abbildung links), kann man über mehrere Registerkarten direkt auf das Dateisystem zugreifen, diverse Hilfequellen aufrufen, frei definierbare externe Werkzeuge einblenden, Codeteile übernehmen und die Inhalte von Komponenten betrachten.

HINWEIS: PrimalScript bietet für viele beim Scripting verwendete Objekte eine Auswahlfunktion der jeweils zur Verfügung stehenden Befehle an. Mehr dazu erfahren Sie in Kapitel 4 im Zusammenhang mit Objekten, weil an dieser Stelle die notwendigen Grundbegriffe noch nicht erklärt sind.

Abspeichern

Nach der erfolgreichen Eingabe des Skripts muss dieses selbstverständlich gespeichert werden – es sei denn, man will es nicht für die Ewigkeit aufbewahren. Über das Menü Tools|Options lässt sich auch festlegen, ob das Skript bei Ausführung automatisch gespeichert und die vorherige Version in eine .bak-Datei umbenannt werden soll.

Start des Skripts

Um das Skript auszuführen, reicht anschließend das Drücken der (F7)-Taste oder des entsprechenden Symbols in der Symbolleiste. Vom Skript erzeugte Ausgaben werden automatisch in einem eigenen Ausgabefenster angezeigt. Diese Ausgaben sind auch nach der Ausführung noch verfügbar, sodass sie später ausgewertet werden können.

Ausgabe von Meldungen

Innerhalb des Ausgabefensters werden alle Meldungen des Skripts ausgegeben. Tritt ein Fehler auf, wird dieser mit einer Meldung und der Angabe der Zeilennummer ebenfalls dort ausgegeben. Mithilfe der Zeilennummer lässt sich dann der Fehler im Code besser finden und untersuchen.

2.3 Der WSH-Spezialist: SystemScripter

Unser deutscher Kollege Dr. Tobias Weltner ist angetreten, PrimalScript Konkurrenz zu machen. Sein SystemScripter ist hervorgegangen aus dem Vorprodukt Scripting Spy und trägt aktuell die Versionsnummer 6.0 aus dem Jahre 2008. Leider ist dies auch die letzte Version. Dafür gibt es die Software inzwischen ohne Kosten [SysScripter].

Vergleich mit PrimalScript

Der SystemScripter unterscheidet sich in den folgenden Punkten von PrimalScript:

- bessere IntelliSense-Funktion für Scripting-Objekte,
- IntelliSense-Funktion auch für Klassen der Windows Management Instrumentation (vgl. Kapitel 5.6),
- Codeprüfung bereits während der Eingabe,
- Laufzeitfehler werden als Tooltipps angezeigt,
- Suchfunktion über alle Scripting-Komponenten,
- Generator für Skriptcode-Beispiele,
- Encoding und Decoding von Skripten, um den Skriptcode für Menschen unleserlich zu machen (vgl. Kapitel 18).

Fehlende Funktionen

Gegenüber PrimalScript existieren aber auch drei Einschränkungen beim SystemScripter:

- Der SystemScripter unterstützt nur VBScript und keine anderen Skriptsprachen.
- Er unterstützt nur den WSH und keine anderen Skriptumgebungen.
- Mit dem Skripteditor können nur einfache Skriptdateien (.vbs) bearbeitet werden. XML-basierte Skriptdateien mit der Dateinamenerweiterung .wsf (die in diesem Einsteigerbuch nicht behandelt werden) und Dokumentenformate wie HTML und XML sind nicht editierbar.

Folglich ist der SystemScripter eine gute Wahl, wenn Sie wirklich nur VBScript-Dateien für den WSH editieren wollen. Universeller, aber für WSH-Skriptentwickler an einigen Stellen auch etwas weniger komfortabel ist PrimalScript.

Bezugsquelle

Den SystemScripter gibt es mittlerweile kostenfrei [SysScripter].

Installation

Der SystemScripter trägt sich mit dem Punkt mit SystemScripter öffnen in das Kontextmenü von .vbs-Dateien im Windows Explorer ein. Außerdem hängt sich der System-

Scripter als Tooltipp für .vbs-Dateien in den Explorer ein und zeigt die ersten Zeilen des Quelltexts eines Skripts schon beim Überfahren mit der Maus.

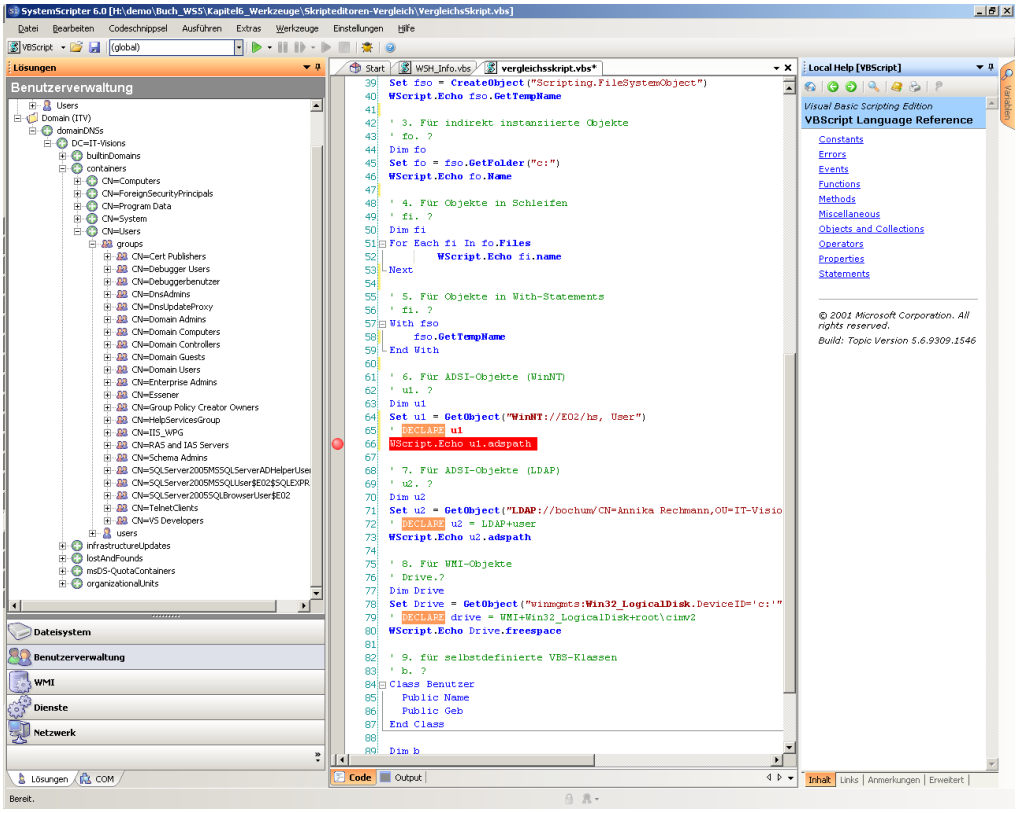

Abbildung 2.5: Skripte editieren im SystemScripter Version 6.0

Grundfunktionen

Der SystemScripter bietet eine Zeilennummerierung und eine gute Farbunterscheidung zwischen VBScript-Sprachbefehlen, Bezeichnern, Operatoren und Literalen. Leider sind die Farben nicht konfigurierbar. In den sehr spärlichen Editor-Optionen kann man nur die Schriftart und die Schriftgröße wählen.

Eingabehilfen

Als Eingabehilfen liefert der SystemScripter Auto-Vervollständigen nicht nur für VBScript-Befehle, sondern auch für bereits verwendete Variablen. Für Objekte bietet der Editor eine Auswahlliste der verfügbaren Befehle, die bei mehr Objekten funktionieren als bei PrimalScript.

Codegeneratoren

Der SystemScripter enthält zahlreiche Codegeneratoren, die Skriptfragmente erzeugen. Einige Codegeneratoren sind über den Kontextmenüeintrag Codeschnipsel erreichbar, andere hingegen über die Fensterleiste Lösungen.

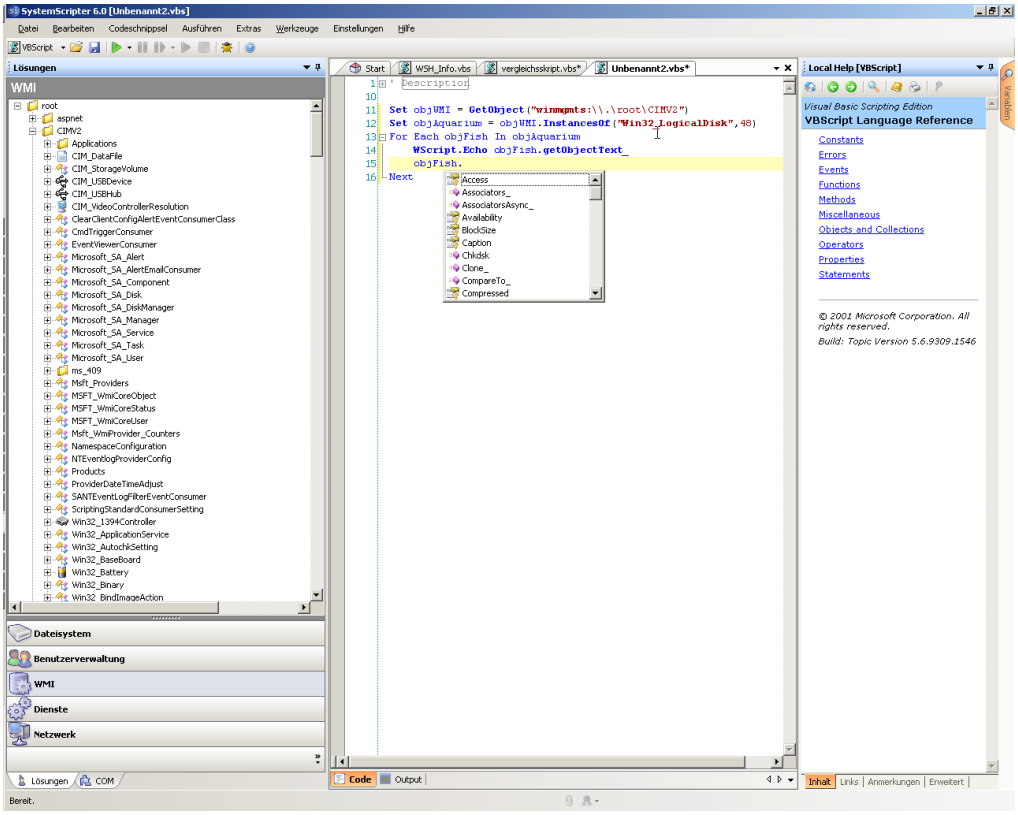

Abbildung 2.6: Der SystemScripter erzeugt Codefragmente durch Ziehen & Fallenlassen (Drag & Drop) aus dem Lösungsfenster.

Durch einen Klick auf den Start-Pfeil in der Symbolleiste oder die Registerkarte Skript ausführen wird ein Skript innerhalb des SystemScripters gestartet. Ausgaben, die eigentlich ins Kommandozeilenfenster gehen würden, landen in der Registerkarte Skript ausführen. Fehler zeigt der SystemScripter direkt im Codefenster an – sehr elegant mithilfe von Tooltipps (siehe folgende Abbildung).

Dokumentation

Eine Schwachstelle des SystemScripters ist die fehlende Dokumentation: Hier wird nur die WSH-Dokumentation von Microsoft, nicht aber eine Hilfedatei zum Editor selbst mitgeliefert. Lediglich auf der Website gibt es einige Hinweise.

Das Debugging, also das Finden und Entfernen von Bugs (Programmfehlern) in Skripten, ist eine zentrale Aufgabe für den Skriptprogrammierer.

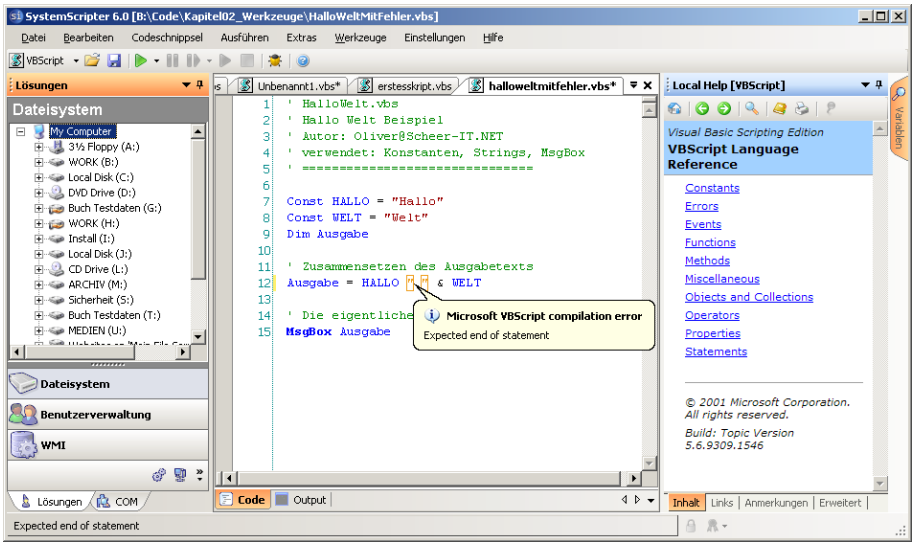

Abbildung 2.7: Anzeige einer Fehlermeldung im SystemScripter 6.0Microsoft Script Debugger

Der Microsoft Script Debugger (msscrdbg.exe) ist ein Debugging-Werkzeug, das zusammen mit dem Internet Explorer installiert wird. Mit dem Script Debugger können aber nicht nur Internet-Explorer-Skripte, sondern auch WSH-Skripte auf Fehler untersucht (auf Neudeutsch: debugt) werden.

Auf der Buch-Website finden Sie den Script Debugger auch als Einzel-Setup im Verzeichnis \Install\Werkzeuge\Debugger\Microsoft Script Debugger.

TIPP: Die Editoren PrimalScript und SystemScripter enthalten auch jeweils einen eigenen Skript-Debugger, der aufgrund der Integration in den Editor komfortabler zu bedienen ist. Allerdings sind diese Produkte im Gegensatz zum Microsoft Script Debugger auch kostenpflichtig.

2.3.1 Fehlerarten

Man muss beim Debugging drei Arten von Fehlern unterscheiden.

Kompilierungsfehler

Kompilierungsfehler: Der Begriff Kompilierungsfehler scheint in einem Buch über Scripting auf den ersten Blick verwunderlich zu sein, da Skriptsprachen ja interpretiert werden. Eine Kompilierung im engeren Sinne findet beim WSH auch nicht statt, doch wird das Skript vor dem Start durch den WSH auf die syntaktische Gültigkeit hin überprüft. Bereits zu diesem Zeitpunkt kann die Vollständigkeit von Sprachkonstrukten kontrolliert werden. Ein Kompilierungsfehler entsteht beispielsweise, wenn Sie im nachfolgenden Listing die Zeile 5 (End If) weglassen. Sofern ein Kompilierungsfehler festgestellt

wird, wird die Ausführung des Skripts gar nicht erst begonnen, selbst wenn die ersten Befehle des Skripts fehlerfrei sind.

Listing 2.2: Demo-Skript

```
a = msgbox("Fehler oder kein Fehler?",vbYesNo)
If a = vbYes then
   u = 0
   x = 7 / u ' Laufzeitfehler: Division durch 0
End If  ' ohne diese Zeile -> Kompilierungsfehler
Msgbox "Ergebnis: " & X
```

Abbildung 2.8:
Anzeige eines Kompilierungsfehlers im WSH

Laufzeitfehler

Laufzeitfehler: Der Parser findet jedoch nicht alle Fehler (z. B. nicht initialisierte Variablen oder nicht definierte Unterroutinen) und kann auch zahlreiche Fehler gar nicht finden (z. B. Division durch null). Diese Fehler werden erst zur Laufzeit festgestellt, d. h., die Skriptausführung beginnt und wird beim Auftreten des Fehlers angehalten. Sofern die Programmzeile, in der sich der Fehler befindet, nicht durchlaufen wird, tritt der Fehler auch nicht auf. Wenn Sie im obigen Listing mit Nein antworten, tritt der „Division durch null"-Fehler also nicht auf.

Abbildung 2.9:
Laufzeitfehler im WSH

Logische Fehler

Logische Fehler: Das größte Problem sind logische Fehler: Sie werden nicht vom WSH bemerkt, sondern führen zu unerwarteten Ergebnissen bei der Ausführung.

2.3.2 Start des Debuggers

Für den Windows Script Host (WSH) muss das Debugging grundsätzlich mit dem Schlüssel HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Settings\ActiveDebugging = 1 aktiviert sein.

Wenn obige Voraussetzung erfüllt ist, gibt es drei Möglichkeiten, den Microsoft Script Debugger für ein WSH-Skript zu starten:

- Wenn das Skript mit der WSH-Kommandozeilenoption //D aufgerufen wurde, startet der WSH den Debugger bei einem Kompilierungs- oder Laufzeitfehler.
- Wenn das Skript mit der WSH-Kommandozeilenoption //X aufgerufen wurde, startet der WSH den Debugger schon beim ersten Befehl unabhängig davon, ob ein Fehler passiert ist. Diese Option benötigt man zum Finden logischer Fehler.
- In VBScript gibt es einen speziellen Befehl, der den Debugger startet: Stop. Zusätzlich muss der WSH aber auch mit der Kommandozeilenoption //D gestartet werden.

Die folgende Programmzeile startet das Skript MonitorServices.vbs von der ersten Programmzeile an im Debugger: Beispiel

```
cscript //X H:\CD\Skripte\10_Services\MonitorServices.vbs
```

In der nächsten Grafik sehen Sie das Skript innerhalb des Microsoft Script Debuggers. Die erste Zeile, die einen Befehl enthält (Dim-Anweisungen werden ignoriert), wird vom Debugger gelb unterlegt.

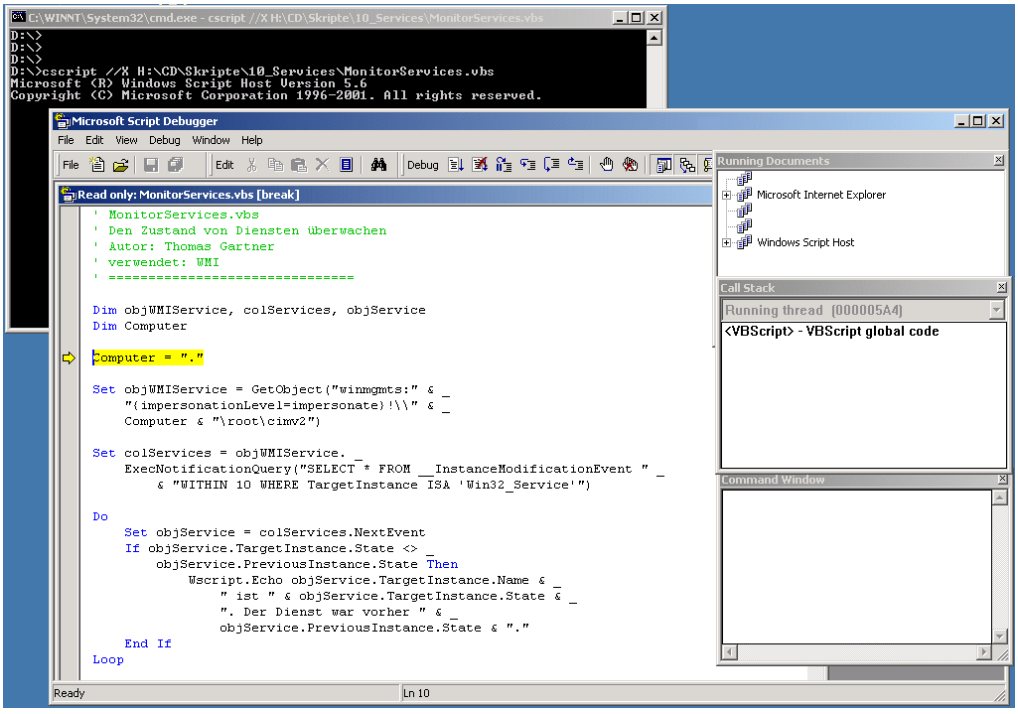

Abbildung 2.10: Start eines Skripts im Microsoft Script Debugger

HINWEIS: Neben dem kostenlosen Microsoft Script Debugger gibt es noch einen WSH-Debugger im kostenpflichtigen Microsoft-Produkt Visual InterDev 6.0. Sind beide Debugger installiert, bietet Windows vor dem Aufruf des Debuggers einen Auswahldialog.

2.3.3 Funktionen des Microsoft Script Debuggers

Innerhalb des Microsoft Script Debuggers stehen folgende Funktionen zur Verfügung:

Funktionen im Debug-Modus

- Durch Drücken der Taste **F8** gelangen Sie zum nächsten Befehl (Einzelschrittmodus). Die jeweils aktuelle Zeile ist gelb markiert.
- Die Taste **F5** setzt das Programm fort. Es läuft weiter, bis es auf einen Fehler, einen Stop-Befehl oder einen zuvor im Debugger gesetzten Haltepunkt trifft.
- Über die Taste **F9** können Haltepunkte gesetzt werden, an denen das Skript stoppen soll (Zeilen mit Haltepunkt werden vom Debugger rot markiert und mit einem Punkt vor der Zeile versehen).
- In einem sogenannten Direktfenster (Command Window) kann man mit dem Fragezeichen (**?**) Werte von Variablen abfragen und Unterroutinen direkt aufrufen.
- Die Aufrufreihenfolge der Unterroutinen kann man im Fenster Call Stack betrachten.

Editor

Der Script Debugger besitzt zwar einen eingebauten einfachen Skripteditor (Menü Datei/Neu), jedoch können auch darin erstellte Skripte nicht innerhalb des Debuggers gestartet werden, sondern müssen extern aufgerufen werden und unterliegen damit den gleichen Beschränkungen wie andere Skripte auch.

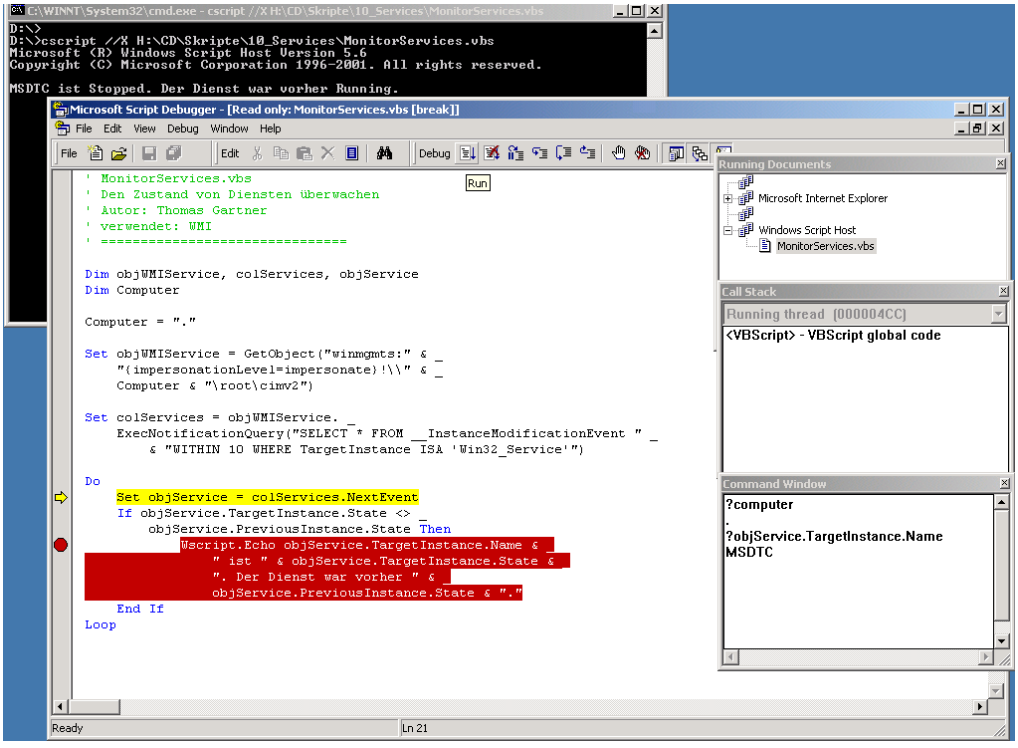

Abbildung 2.11: Analyse eines Skripts im Script Debugger

2.4 Fragen und Aufgaben

1. Welches ist das einfachste Mittel, um eine VBScript-Anwendung zu erstellen?
2. Was versteht man unter dem Begriff Debugging?
3. Welche Arten von Fehlern können innerhalb eines Skripts auftreten?

3 Scripting und die Benutzerkontensteuerung

Dem Problem, dass ein fortgeschrittener Benutzer, Administrator oder Entwickler in bisherigen Windows-Versionen nur reibungslos arbeiten konnte, wenn er immer als Administrator an seinem Rechner angemeldet war, begegnet Microsoft seit Windows Vista mit einer neuen Funktion, die „Benutzerkontensteuerung“ heißt (engl. User Account Control (UAC), zuvor auch User Account Protection (UAP) genannt).

3.1 Benutzerkontensteuerung

UAC

Benutzerkontensteuerung (User Account Control, UAC) bedeutet, dass alle Anwendungen seit Windows Vista immer unter normalen Benutzerrechten laufen, auch wenn ein Administrator angemeldet ist. Wenn eine Anwendung höhere Rechte benötigt (z. B. administrative Aktionen, die zu Veränderungen am System führen), fragt Windows explizit in Form eines sogenannten Consent Interface beim Benutzer nach, ob der Anwendung diese Rechte gewährt werden sollen.

Bei Administratoren reicht zur Bestätigung ein Mausklick („Consent Prompt“), normale Benutzer müssen Name und Kennwort eines administrativen Kontos eingeben („Credential Prompt“). Erst nach der Bestätigung wird die Anwendung mit administrativen Rechten ausgestattet (engl. „elevated“).

Der einfache Wechsel aus dem normalen Benutzerkontext heraus soll Administratoren und Softwareentwickler dazu motivieren, im Standardfall immer als Normalbenutzer zu arbeiten mit der Gewissheit, doch schnell zu mehr Macht zu kommen. Während der Anzeige des sogenannten „Consent Interface“ graut der Rest von Windows aus und steht aus Sicherheitsgründen nicht zur Verfügung.