Windows PowerShell 超入門 [4.0対応]

新丈 径[著]
Kei Shinjou

キホン操作からスクリプトの作成、文法などをていねいに解説

めざせ上級ユーザー

- 既存の道具やノウハウを活用
- 面倒な繰り返し操作を一括処理
- デスクトップ作業の自動化を実現できる

インプレス

商標

◆ Windows、Windows Server、Windows PowerShell および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

◆その他、本書に登場する会社名、製品名、サービス名は、各社の登録商標または商標です。

◆本文中では、(R)、(C)、TM マークは表記しておりません。

本書の利用について

◆本書の内容に基づく実施・運用において発生したいかなる損害も、株式会社インプレスと著者は一切の責任を負いません。

◆本書の内容は、2014 年 6 月の執筆時点のものです。本書で紹介した製品／サービスなどの名称や内容は変更される可能性があります。あらかじめご注意ください。

◆ Web サイトの画面、URL などは、予告なく変更される場合があります。あらかじめご了承ください。

はじめに

Windows はその名のとおり、画面にウィンドウを表示して、グラフィカルなユーザーインターフェイス（GUI）をマウスなどで操作しながら使うことを基本とした OS です。しかし、GUI による PC の操作は人に優しい半面、操作がまだるっこしいと感じられるときもあります。PC に自分の意思を最速で伝える方法は残念ながら、現時点では、キーボードによるコマンドの入力です（最近のモーションセンサー技術の隆盛と流行は、新たなユーザーエクスペリエンスの到来を示していますし、そこから PC へのもっとダイレクトな意思伝達の方法が生まれるやもしれません）。

その一方で、キーボードをタタタッと叩いて、Enter キーをタンッ！ とするだけで PC を思いのままに操作できるのは実はとっても気持ちがよいことです。そして、本書で取り上げる PowerShell には、Windows をこのようなキーボード操作で自在に操るだけの力が備わっているのです。

本書では、PowerShell を使いこなすために知っておくべき最低限の知識を紹介することにしました。本書を読んだだけでは、PowerShell をしゃぶりつくしたことには到底なりませんが、それでも PowerShell がどんなものかを知り、PowerShell で簡単なスクリプトを書けるようになるでしょう。そこから、Windows をキーボードで使いこなす楽しさを少しでも感じていただければと思います。といっても、そこはあくまでもスタート地点でしかありません。そこからは、より高度な内容を取り扱っている解説書、インターネットで提供されている豊富な情報を参考にしながら、PowerShell の奥深い世界への旅を楽しんでみてください。

本書がその最初の一歩として皆さんのお役に立てば幸いです。

2014 年 6 月

新丈 径

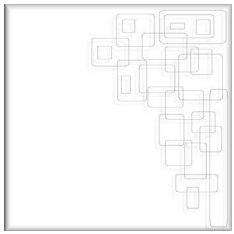
目　次

第 1 章　Windows PowerShell とは

Windows PowerShell（以下、PowerShell と略）がリリースされたのは、.NET Framework 2.0 がリリースされた翌年、2006 年のことです。PowerShell の最初のバージョンがリリースされて以来、Windows システムの拡張やクラウドサービスへの対応などに適応し、Windows におけるオブジェクトベースの管理ツールとして大きな進化を遂げてきました。

PowerShell は、それまで WSH（Windows Script Host）、VBScript に代わる、Windows システムを包括的に管理するコマンドラインツールといえるでしょう。IT 管理者にとっては、Windows システム全体を管理するツールとして、すでになくてはならない存在です。しかし、IT の専門家向けのツールであり、多機能であるがゆえに、一般のユーザーや経験の浅い若手 IT 管理者が利用するには、少々敷居の高いツールであることも事実です。ですが.NET Framework や COM などといったオブジェクトを活用して、ファイルの処理やディスク管理の自動化を図るうえでは、一般ユーザーにとっても非常に有効なツールであることは間違いありません。

本章では、PowerShell を一般ユーザーレベルで利用するための、基本的な知識と環境の準備を行います。



1.1　仮想的な OS 環境の管理ツール

Microsoft Windows（以下、Windows と略）は、1990 年代に PC のデファクトのオペレーティングシステム環境として爆発的に普及したため、当時 Windows が動作している環境であれば、PC やデバイス間のデータ交換や変換に問題が起こることは比較的少なかったといえるでしょう。ところが、1990 年代後半に、インターネットの利用環境が普及したことにより、インターネットを介したデータ交換の必要性や XML Web サービス*1 を利用する Windows アプリケーションの需要が高まってきました。

そして 21 世紀の初頭、それまでの MFC や ATL といった開発環境と決別し、まったく新しいアプリケーションの開発環境であり、実行環境である、.NET Framework の開発を機に、Windows システムのオブジェクト指向に向けた変革が始まりました（Windows Update で.NET Framework の配布が開始されたのは 2002 年 4 月）。オブジェクト指向システムとして開発された Smalltalk-80 のリリースからおよそ 20 年後のことです。

.NET Framework の採用により、OS のアーキテクチャ、開発環境といった、ソフトウェアプラットフォームのオブジェクト指向化という動きを加速させました（**図 1-1**）。それにより、Web アプリケーションやデータベースを利用するソフトウェア開発の効率化はもちろん、システム管理上の課題で

＊1　XML、HTTP、SOAP、REST などのインターネットの標準技術を利用して、異なるプラットフォームのアプリケーションとも統合可能なソフトウェアの概念。

あった、当時の DLL ヘル[*2]の解消や他のハードウェアプラットフォームや OS との相互運用といったことを目指したものではあったことも、間違いないでしょう。

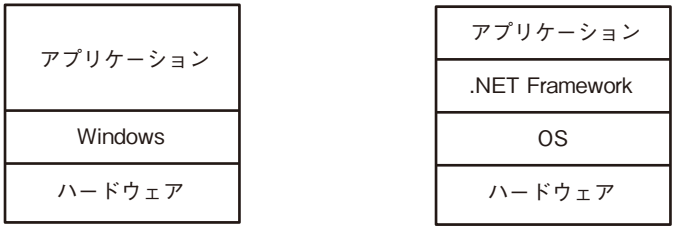


図 1-1　Windows と.NET Framework の概念図

かつては、Windows というオペレーティングシステム自体がアプリケーションの実行環境であり、アプリケーションの開発環境も、Win32 API や MFC（Microsoft Foundation Class）といった、Windows システムを直接アクセスするような開発基盤でした。それが、実際の Windows という OS の上に、.NET Framework という**仮想的な OS** の層を作り（これを「オブジェクト指向 OS」と称する人もいます）、アプリケーションからの Windows へのアクセスを.NET Framework という仮想的な OS を介して行うようになりました。

また、「仮想的な OS」が存在することにより、CPU や OS の種類を問わず、.NET Framework の標準仕様である共通言語基盤（CLI：Common Language Interface）の仕様に従った仮想的な OS 環境を実装することも可能になりました。すでに、マイクロソフト以外のベンダーが、他の OS 環境へ独自に実装しています。Xamarin による Mono プロジェクトや、いくつかのオープンソースによる実装プロジェクトも存在しており、こうした環境を利用することで、.NET Framework で開発したアプリケーションを、Linux や Android、Mac OS X などの OS でも利用できるようになっています（**図 1-2**）。

＊2　DLL（Dynamic Link Library）の仕様上の変更により、仕様変更前の DLL の機能に依存したアプリケーションが動作しなくなること。また、さまざまなバージョンの DLL が多種多様に存在することで、管理が複雑になることを指します。

「仮想的な OS」は、ファイルやメモリなどのリソースの管理、プロセスやスレッドの制御といったシステムへのアクセスを、従来の Win32 API へのアクセスなどよりも抽象化された「オブジェクト」として実装しています。言い方を換えると、.NET Framework とは、ファイルをアクセスするオブジェクト、メモリリソースを管理するオブジェクト、プロセスやスレッドを管理するオブジェクトなどによって構成されています。

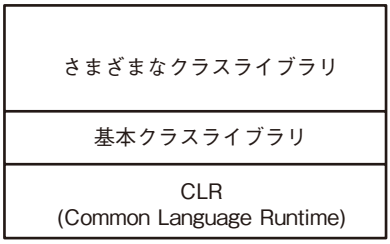
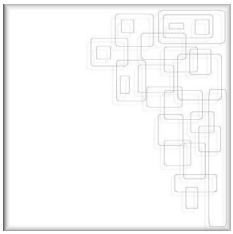


図 1-2　.NET Framework と共通言語基盤

このように、オブジェクトを利用してソフトウェアを開発する、あるいはオブジェクト層（.NET Framework）上でアプリケーションが実行されるということは、OS が管理するコンピュータリソースをオブジェクトを介して操作する、コンピュータの運用管理も自ずからオブジェクトを対象としたものになります。これまで、コマンドプロンプトのバッチ処理やその他の管理ツールが不十分であったことはいうまでもないことですが、オブジェクトを操作し、管理するための新しいツールの登場は必然だったといえるでしょう。



1.2　PowerShell の進化の履歴

先に述べたような市場の背景と Windows システム自体がオブジェクト指向 OS として改良されたことで、従来の管理ツールとはまったく異なる、オブジェクトを操作可能な管理ツールとして PowerShell が開発されました。リリースされた当初は、VBScript と同程度の機能でしたが、2.0（Windows 7 で標準搭載）、3.0（Windows 8 標準搭載）、4.0（Windows 8.1 標準搭載）、2014 年には 5.0 と、マイクロソフトの Windows Server をはじめ、各種のサーバー管理やネットワーク管理の自動化ツールとして発展してきました。OS の機能が拡張や、Microsoft Azure や Office 365 といったクラウドサービスの提供とともに、PowerShell の機能も拡張され、非常に多機能なスクリプト言語となっています。マイクロソフトの OS 環境においては、最も多機能なスクリプト言語であるといってもよいでしょう。以下では簡単に、PowerShell の進化の履歴をたどってみましょう。

1.2.1　PowerShell 1.0

PowerShell は、もともと Microsoft Exchange Server 2007 の管理を自動化するツールとして付属していたものです。2006 年に最初の 1.0 がリリースされました。Windows XP、Windows Server 2003 が市場で主流になっていた頃です。それまでの DOS コマンド群やバッチファイル、VBScript、WHS などによるいろいろな手法を組み合わせ、つぎはぎだらけのシステム管理か

らの脱却を目指して開発されました。

1.2.2　PowerShell 2.0

Windows 7 のリリースとともに、PowerSehll 2.0 が標準で OS にインストールされるようになりました。PowerShell 2.0 では、新たな機能として、リモートコンピュータのコマンドの実行、バックグラウンドジョブ、イベント処理、スクリプトデバッガー、PowerShell ISE（Integrated Scripting Environment）などといった機能が拡張されました。

1.2.3　PowerShell 3.0

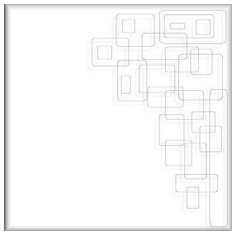
2012 年の Windows 8.0 のリリースでは、PowerShell 3.0 が搭載されました。このバージョンでは、大幅な機能拡張が施され、2000 を超えるコマンドレットが提供されるようになっています。スクリプト言語などの動的言語機能を提供するランタイム“Dynamic Language Runtime（DLR）”に対応し、さらにパーサーの改善や JIT コンパイルの採用などによりスクリプトの実行速度が向上しています。運用管理ツールとしては、前バージョンから飛躍的に機能が向上したバージョンです。

1.2.4　PowerShell 4.0

PowerShell のメジャーバージョンの更新は、これまで 3 年に 1 回のペースで行われてきましたが、2013 年の Windows 8.1 のリリース時に、PowerShell 4.0 がリリースされました。Windows 7 SP1 にも対応していますが、注意が必要なのは、Windows 8.0 ではこのバージョンがサポートされていないことです（つまり、Windows 8.0 のユーザーが PowerShell 4.0 を使いたければ、Windows 8.1 へアップグレードする必要があるということです。とはいえ、使いやすさなどを考えれば、PowerShell 4.0 を使いたいかどうかに関係なく、

Windows 8.1 以降のバージョンにアップグレードするのがよいと筆者は思います)。機能的な拡張の目玉は、DSC（Desired State Configuration）という、サーバー構成をコードで記述することで自動化を実現する、新たな Windows の構成管理ツールです[*3]。

＊3　コンピュータの構成を、設定ファイルに、項目と設定値という形式で並べて記述する。



1.3　Windows ユーザーにとっての用途

これまで、Windows 上の操作やアプリケーションの操作で、同じことを何度も繰り返すことに焦燥感を抱いたことがある人なら、PowerShell はかなり役立つツールになる可能性があります。繰り返しの処理が面倒だと思ったら、ちょっと立ち止まって PowerShell の関数やスクリプトを使うことを検討すれば、意外なほど簡単に問題が解決するかもしれません。おそらく、ものぐさな人にとってこそ、PowerShell は有効なツールです。

確かに、PowerShell を使うのは、難しそうだと感じる人が多いのも事実です。通常はアプリケーションしか利用しないユーザーであれば、PowerShell の存在自体を知らないという方も少なくないでしょう。PowerShell が管理ツールとして発展してきており、IT 管理者が利用することを前提とした書籍や Web の解説が多いため、専門的な内容の書籍が多く、一般ユーザーには扱いにくい印象を与えます。ですが、個人ユーザーにとって役に立たないかといえば、そんなことはありません。Windows 上で Microsoft Office のようなアプリケーションを利用するユーザーや Windows 環境におけるテキストや XHTML、ディスクやファイルの操作などを行うだけのユーザーであっても、PowerShell で一括で処理可能なことが数多くあります。

また、PowerShell という手段を覚えたからといって、Windows アプリケーションが使えなくなるわけではありません。GUI ツールが便利なのであれば、むしろ積極的にそうしたツールは使うべきです。ただし、頻繁に（数十

から数百回）繰り返し行う処理で、しかも実行する頻度の高い処理であれば、PowerShell を使うことで、格段に手間を省くことができる場合もあります。

1.3.1　多様な機能を持つコマンドの組み合わせ

とはいっても、「それではスクリプトを作成しましょう」といわれても、最初は敷居が高いかもしれません。しかし、PowerShell を使うということは、必ずしも何十行、何百行ものスクリプトをゼロから書かなければならないということではありません。

たとえ 1 行のコマンドラインでも、PowerShell に備わった機能を組み合わせるだけで、驚くほど多様なことが実現できます。たとえば、ルートディレクトリから階層的にディレクトリをたどって、拡張子が“.txt”のファイルから、"PowerShell"という文字列を含むファイルを探す処理を考えてみましょう。

Windows エクスプローラーで操作するなら、ルートディレクトリで、検索ボックスに“*.txt”を指定すれば、一括でファイルを一覧することはできます。ただしフォルダーとファイルの検索では、フォルダーオプションを適切に設定しないと、なかなか意図した検索対象となりません。検索しようとしているフォルダーやファイルは、インデックスの作成対象となっているのか、ファイルの内容を常に検索するのか、インデックスを使わないようにするのか、といった設定を確認しておく必要もあります。また、検索されたファイルが指定した文字列を含んでいることはわかっても、Windows エクスプローラーの出力結果では、ファイル名を表示するだけなので、その文字列のファイル内での位置（行数）や前後の文字列などは表示されません。得られる情報も少ないため、出力されたファイル名リストの加工や絞り込みを考えると、Windows エクスプローラーでは、まかなえないことも結構あります。扱うファイルが数個なら、個々のファイルの内容を調べるのもそれほど大変な作業ではありませんが、数百個のファイルを個々に調べるとなると、気が遠くなるような作業です。それを PowerShell を使えば、次の 1 行で処理できます。

```
>get-childitem c:¥* -include *.txt -recurse | select-string -pattern
 "PowerShell" -casesensitive
```

ここでは、詳細は説明しませんが、第 2 章以降でこうした事例を数多く解説しています。

もちろん、Windows アプリケーションにも膨大な資産があり、こうした処理を行う専用のアプリケーションも存在します。ファイルの検索と文字列の検索に目的が絞られていて、そうしたツールを使い慣れており、その特定のツールを使った方が効率がよいのであれば、無理に PowerShell を使う必要はありません。いや、使うべきではないでしょう。

しかし、Windows アプリケーションが専用ツールであるのに対し、PowerShell は各種コマンド機能を組み合わせて使う、汎用的なツールである点が、重要なポイントです。個々のパーツは小さな機能しか持ちませんが、組み合わせることでさまざまな状況に柔軟に対応可能であるおかげで、専用ツールではまかなえないような用途には、PowerShell が大きな効果を発揮します。

1.3.2　既存のスクリプトを利用する

また、PowerShell が登場してからすでに多くの利用者が蓄積したノウハウがインターネットには存在します（PowerShell 関連の情報を提供しているサイトの URL は、本文中の脚注に適宜示しましたので、参照してください）。マイクロソフトが提供している Script Browser を利用するだけでも、かなり多くのスクリプトを検索できます。自分の目的にぴったり合ったもの（スクリプトや関数）がすでに存在するのであれば、それを利用しない理由はありません。たとえば、一般によく使われそうなサンプルとして、Word ファイルから PDF や XPS を作成する PowerShell の関数として、`Convert-Doc2PdfXps` が国内サイトで提供されています[*4]。

次の 1 行で、**C:¥Work** に置かれたすべての Word ファイルを PDF に変換

＊4　http://gallery.technet.microsoft.com/office/0c6c6e6a-5ce0-4c3b-be51-aae2b4ed9fd3

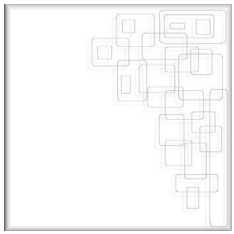
できます。

```
PS C:¥Work>Convert-Doc2PdfXps C:¥Work¥*.docx
```

Word のみを使うのであれば、個々のファイルを開き、［ファイル］－［PDF として保存］という操作を、PDF への変換対象となる Word ファイルの数だけ繰り返さなければなりませんが、カレントディレクトリにたとえ数百個の Word ファイルがあっても、1 行のコマンドを実行するだけです。

また、たとえそのまま使えないスクリプトだとしても、PowerShell の基本的な文法を理解していれば、既存のスクリプトを変更して利用することもできます。ここで紹介した `Convert-Doc2PdfXPS` は、カレントディレクトリの Word ファイルのみを変換対象としていますが、先に示した“`*.txt`”ファイルの検索のように、PowerShell では階層化ディレクトリをすべて検索することができるので、このスクリプトにほんの少し手を加えるだけで、ドライブ内のすべての Word ファイルを対象として、PDF に変換するような処理も実現できます。このように、デスクトップ上の多くの処理は、既存コードの変更でまかなえることも少なくありません。

先にも述べたように、もともと PowerShell は Windows Server や Exchange Server の管理の自動化を目的として作られたものですから、その機能全体を把握するのは、かなり骨が折れます。しかし、ここで目的としている、デスクトップ環境の操作、アプリケーションで作成したファイルの加工の効率化という範囲であれば、比較的容易に PowerShell を活用できます。



1.4　PowerShell の特徴

PowerShell は、オブジェクトに対応した管理ツールとして、コマンドプロンプトのような、これまでの Windows の管理ツールとは異なる点がいくつかありますので、注意すべき点について簡単に説明します。

1.4.1　コマンドレット

コマンドレットは、Windows PowerShell コマンドであり、コマンドプロンプトで実行されるコマンドラインツールに相当するものです。コマンドプロンプトでも、`cls`、`dir`、`cd` などといったあらかじめコマンドプロンプト自体に組み込まれている内部コマンドはありますが、機能の豊富なコマンドの多く、たとえば `chkdsk.exe`、`sort.exe`、`ipconfig.exe` といったコマンドは、それぞれ独立のプログラムとして作成され（外部コマンドとも呼ばれます）、コマンドプロンプトから呼び出されて実行されるコマンドラインツールです。PowerShell にも組み込みのコマンドがいくつかありますが、高度で複雑な処理のほとんどはコマンドレットによって実行されます。

PowerShell さえ起動すれば、すべてのコマンドレットは実行できるように見えますが、コマンドレットの実行には、.NET Framework のクラスライブラリが必要です。なぜなら、コマンドレットの実体は、.NET Framework の基本クラスライブラリ（Base Class Library）の上の `Cmdlet` クラスの「オブジェクト」だからです。つまり、PowerShell のコマンドレットは、.NET

Framework のクラスとして実装されています。

1.4.2 PowerShell とオブジェクト

PowerShell は、「オブジェクトベーススクリプト言語である」といわれます。「オブジェクトベース」というのは、他のスクリプト言語やシェル環境では操作の対象がテキスト（アスキーコードと漢字コード）であるのに対して、PowerShell で操作するのは、オブジェクトであるということです。たとえば、どちらも一見同じ文字列のように見えても、オブジェクトであればプロパティ（性質のようなもの、文字列の長さなど）とメソッド（オブジェクトに対して実行できるアクション、大文字と小文字の変換など）が用意されています。このあたりの詳細は、第 3 章以降で解説します。

また、PowerShell で扱えるオブジェクトは、.NET オブジェクト（.NET Framework クラスライブラリが提供するオブジェクト）だけでなく、COM オブジェクト（Component Object Model Object）、WMI オブジェクト（Windows Management Interface Object）などを操作の対象として利用可能です（**図 1-3**）。

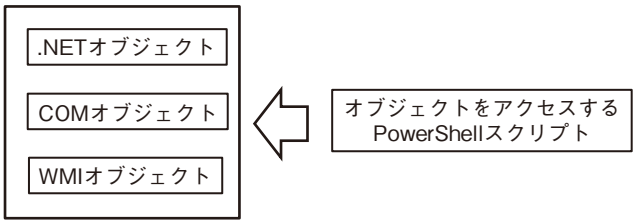


図 1-3　PowerShell からのオブジェクトへのアクセス

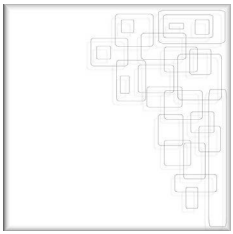
COM オブジェクトと聞いても馴染みがないかもしれませんが、Windows アプリケーションで COM オブジェクトが使われている実際の状況がわかれば、容易に理解できると思います。長年 Windows を使ってきた人であれば、HTML 上でリンクされた Excel ファイルをクリックすると、IE のウィンドウ内で Excel がシートの表示を行ったり、Outlook の編集機能に Word が使わ

れていたりといった、1 つのアプリケーションの機能を他のアプリケーションから利用しているのを体験したことがあると思います。これは、Excel や Word の機能が、COM オブジェクトとして、他のプログラムから操作可能なオブジェクトを提供しているからです。

1.4.3　コマンドレットで実行できること

PowerShell を利用した処理として、先にファイルの検索と PDF への変換の例を挙げましたが、PowerShell のコマンドレットを使用して実行できる作業には、テキストファイルの読み取りと書き込みから、データの並べ替え、デバイスの情報取得、フィルター処理まで、さまざまな種類があります。以下に示したのは、Windows クライアントで行うことを前提に示した事例ですが、ここに挙げた項目以外にも、セキュリティ管理やリモートアクセス、並列処理なども可能です。

●Windows PowerShell のエイリアスの設定
●ヘルプと情報の参照
　コマンドレットの使い方、Windows PowerShell のバージョン情報を表示する
●日付と時刻の取得／設定
　現在の日付と時刻の取得や日付計算を実行する
●ファイルとフォルダーの操作
　ファイルとフォルダーの作成、名前の変更、コピー、削除を実行する
●データの読み書き、変換、保存
　テキストファイルへの読み取りと書き込み、HTML 形式または XML 形式でのデータを保存する
●システム管理タスク
　サービスの開始と停止、WMI を使用したデータの取得、イベントの管理、イベントログの処理を行う



1.5　PowerShell 4.0 のインストール

すでに Windows 8.1 を利用している方は、PowerShell 4.0 の実行環境が用意されているので、追加でインストールするものはありません。Windows 8.0 のユーザーは、8.1 にアップデートすると、PowerShell も 3.0 から 4.0 へアップデートされます。

Windows 7 には、PowerShell 2.0 が標準でバンドルされています。PowerShell 4.0 を利用するには、Windows 7 の SP1（Service Pack 1）でシステムのアップデートを行ったあと、PowerShell 4.0 を実行するのに必要なファイルをマイクロソフトの Web サイトからダウンロードする必要があります。PowerShell は、単体で提供されているものではなく、Windows 環境の管理基盤である WMF（Windows Management Framework）の 1 つのツールとして提供されます。WMF には、Windows 環境を管理するツール群として、PowerShell と WinRM（Windows のリモート管理）、PowerShell Web Services などが含まれます（バージョンによって構成されているツールが異なります）。

本書では、PowerShell 4.0 を対象としていますので、実行可能な OS は、Windows 7 SP1 以降と Windows 8.1 になります[*5]。

＊5　PowerShell 4.0 の実行可能なサーバー OS としては、Windows Server 2008 SP1/2012/2012R2 があります。

1.5.1　PowerShell の入手先

ここでは、Windows 7 SP1 を対象として、PowerShell 4.0 をインストールするために必要なファイルを用意するところから始めましょう。Windows 8.1 のユーザーでも、今後新しいバージョンの PowerShell がリリースされ、導入する際には、ほぼ同じような手順で対応できます。

PowerShell 4.0 を実行するには、.NET Framework 4.5 が必須ですので、WMF 4.0 をインストールする前に以下の URL から、`dotNetFx45_Full_setup.exe` をダウンロードしてください。

■ Download Center　Microsoft .NET Framework 4.5（日本語）

```
http://www.microsoft.com/ja-JP/download/details.aspx?id=30653
```

まず、上の URL で表示されるサイトの、［ダウンロード］ボタンをクリッ

図 1-4　Microsoft .NET Framework 4.5 のダウンロードサイトの画面

クします（**図 1-4**）。「その他の推奨ダウンロード」などが表示されますが、無視して［ダウンロードせずに続けます］ボタンをクリックします。ダウンロードされたファイルを実行すれば、.NET Framework 4.5 のインストールは終了です。

■ Download Center　Windows Management Framework4.0.（日本語）

```
http://www.microsoft.com/ja-JP/download/details.aspx?id=40855
```

次に、上の URL で示されたサイトの、［ダウンロード］ボタンをクリックして、Windows Management Framework 4.0 のダウンロードを行います（**図 1-5**）。

図 1-5　WMF 4.0 のダウンロードサイトの画面

するとダウンロードファイルを選択するウィンドウが現れるので、必要なファイルのボタンをクリックしてダウンロードするファイルを選択し、［次

へ］ボタンをクリックしてください（**図 1-6**）

図 1-6　ダウンロードするファイルの指定画面

ダウンロードファイルの選択で表示されているものは、以下の内容です。

●`Windows Management Framework 4.0 Release Notes.docx`
リリースノート

●`Windows PowerShell Desired State Configuration Quick Reference for Windows Management Framework 4.0.pdf`
DSC のクイックリファレンス（PDF 版）

●Windows PowerShell Desired State Configuration Quick Reference for Windows Management Framework 4.0.pptx
DSC のクイックリファレンス（PowerPoint 版）

●`Windows6.1-KB2819745-x64-MultiPkg.msu`
（WMF 4.0　64 ビット版）

●Windows6.1-KB2819745-x86-MultiPkg.msu
（WMF 4.0　32 ビット版）
●Windows8-RT-KB2799888-x64.msu
（WMF 4.0 64 ビット版）

PowerShell インストールに必要なファイルは、WMF 4.0（Windows6.1-KB*.msu）のみです。ただし、32 ビット OS か 64 ビット OS かによって、インストールするファイルが異なるので注意してください。

WMF 4.0 のファイルを展開すると、以下のツールが Windows にインストールされます。

●Windows PowerShell 4.0
●Windows PowerShell Integrated Scripting Environment（ISE）
●Windows PowerShell Web Services（Management Data IIS Extension）
●Windows Remote Management（WinRM）
●Windows Management Instrumentation（WMI）

本書で扱うのは、PowerShell 4.0 と PowerShell ISE（Integrated Scripting Environment）のみですが、他のツールについても簡単に紹介しておきます。必要のない方は、読み飛ばしていただいてかまいません。

1.5.2　PowerShell 4.0 の起動

Windows 7 で PowerShell を起動する場合は、［スタート］メニューの［プログラムとファイルの検索］に、“powershell”（たいていは、「p」と入力するだけでメニューに表示されます）と入力すると、スタートメニューに「Windows PowerShell」「Windows PowerShell ISE」などが表示されます。Windows 8.1 では［スタート］画面で直接「PowerShell」などと入力すればよいでしょう。表示されている項目をクリックすると、一般ユーザーモードで起動されます。マルウェアなどの実行を防ぐため、PowerShell のコマンドレットの実行権限