Windows PowerShell 逆引きハンドブック

蒲生 睦男◆著

目的からPowerShellのコマンドレットが引ける逆引きリファレンスの決定版!

Windowsシステム/サーバー管理者

必携書!

バージョン 5.0/4.0/3.0/2.0 対応!

24時間無料でサンプルデータをダウンロードできます。

C&R研究所

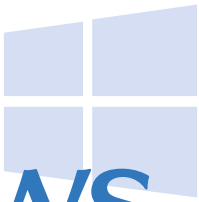
C&R研究所について

C&R研究所は新潟市にある出版社です。ユニークな社風や教育方針は新聞やテレビなどで紹介されたりします。詳細については、次のWebサイトでご覧いただくことができます。

www.c-r.com

また、新潟本社には2代目会社犬「ラッキー」がいます。名刺を持つ正式な社員として広報部に勤務しつつ、セラピードッグとして社内のメンタルヘルスにも貢献しています。

◉2代目会社犬「ラッキー」



Windows PowerShell 逆引きハンドブック

蒲生 睦男◆著

C&R研究所

■権利について

- 本書に記述されている社名・製品名などは、一般に各社の商標または登録商標です。
- 本書では™、©、®は割愛しています。

■本書の内容について

- 本書は著者・編集者が実際に操作した結果を慎重に検討し、著述・編集しています。ただし、本書の記述内容に関わる運用結果にまつわるあらゆる損害・障害につきましては、責任を負いませんのであらかじめご了承ください。
- 本書で紹介している操作の画面は、Windows 10を基本にしています。他の環境では、画面のデザインや操作が異なる場合がございますので、あらかじめご了承ください。
- 本書は2016年2月現在の情報で記述しています。

■サンプルについて

- 本書で紹介しているサンプルは、C&R研究所のホームページ（http：//www.c-r.com）からダウンロードすることができます。ダウンロード方法については、5ページを参照してください。
- サンプルデータの動作などについては、著者・編集者が慎重に確認しております。ただし、サンプルデータの運用結果にまつわるあらゆる損害・障害につきましては、責任を負いませんのであらかじめご了承ください。
- サンプルデータの著作権は、著者及びC&R研究所が所有します。許可なく配布・販売することは堅く禁止します。

●本書の内容についてのお問い合わせについて

この度はC&R研究所の書籍をお買いあげいただきましてありがとうございます。本書の内容に関するお問い合わせは、「書名」「該当するページ番号」「返信先」を必ず明記の上、C&R研究所のホームページ（http://www.c-r.com/）の右上の「お問い合わせ」をクリックし、専用フォームからお送りいただくか、FAXまたは郵送で次の宛先までお送りください。お電話でのお問い合わせや本書の内容とは直接的に関係のない事柄に関するご質問にはお答えできませんので、あらかじめご了承ください。

〒950-3122 新潟県新潟市北区西名目所4083-6　株式会社 C&R研究所　編集部
FAX 025-258-2801
「Windows PowerShell逆引きハンドブック」サポート係

PROLOGUE

PowerShell（Windows PowerShell）は、Windows Server 2008 R2、Windows 7から標準で搭載されているシステム管理用のコマンドシェルです。シェルとしてだけでなく、スクリプト言語としても利用できる、とてもパワフルで便利なツールです。

本書は、PowerShellを使ってシステム管理を行う人のためのリファレンスです。PowerShellのコマンドレットを目的別に分類し、やりたいことから探していただけるように、逆引き形式を採用しています。手元に置いていつでもお使いいただけるようにA5サイズを採用していますので、何度でも繰り返しページをめくってみてください。

本書は、PowerShellのバージョン2.0からバージョン5.0の基本的なコマンドレットの実行例を、初心者にもわかりやすいシンプルな事例で紹介しています。さらに、ONEPOINTではポイントとなるパラメータ（引数）についてわかりやすい解説するとともに、COLUMNには、読者のみなさんの理解を深める手助けとなるように、補足情報を豊富に盛り込んでいます。また、コマンドレットの構文も掲載しています。

PowerShellは、日本語のドキュメントが少なく、バージョン3.0以降ではヘルプも日本語化されていません。このため、英語が苦手な人にとっては、習得が困難です。また、公開されているドキュメントの記述も、間違っていたり、情報が不足していたりと、十分ではありません。

本書は、各コマンドレットを実際に実行した結果と、ドキュメントやヘルプの記述を比較し、相違点や矛盾点を精査して記述していますので、これからPowerShellを学ぶ初心者はもちろん、PowerShellのパワーユーザーの参考書としても、最適な内容となっています。

最後に、本書を執筆にするにあたってお世話になったスタッフの皆さまに心から感謝を申し上げます。そして、本書がPowerShellでシステム管理を行う読者の皆様に少しでもお役に立てれば、これ以上の幸せはありません。

2016年2月

蒲生 睦男

本書について

本書の表記方法

本書の表記についての注意点は、次のようになります。

▶コマンドの実行例について

本書のコマンドの実行例については、Windows 10のバージョン5.0を基本にしています。カレントディレクトは「C:¥work」を基本にしています。他のOSやバージョンによって実行結果などが異なる場合があります。また、誌面の都合上、折り返し位置などが実際の画面と異なる場合があります。あらかじめ、ご了承ください。

なお、白地に色付きの文字については、コマンドや実行結果についての解説になります。実際の実行例では表示されません。

▶バージョン5.0の64bit版について

執筆時点では、バージョン5.0の64bit版では、パスに日本語が含まれている、または日本語を入力する場合、フォントをMSゴシックに変更しないと、カーソルの位置などがずれるという問題があります。

この問題は、フォントをMSゴシックに設定(タイトルバーのアイコンをクリックしてメニューから[プロパティ(P)]を選択し、「フォント」タブの[フォント(F)]で「MS ゴシック」を選択して[OK]ボタンをクリック)すると解決します。

▶バージョンの表記について

本文中では、バージョン2.0/3.0/4.0/5.0を、それぞれV2/V3/V4/V5と表記しています。

▶コマンドの中の↵について

本書に記載したコマンドの実行例の中の↵は、コマンドの入力を確定するためなどの[Enter]キーを押すことを表しています。

▶行継続のプロンプトについて

本書に記載したコマンドの実行例の中の行頭の「>>」は、行継続のプロンプトを表しています。なお、バージョン2.0 ～ 4.0や5.0の32bit版は、最後の行を入力した後に、さらに「>>」が表示されるので[Enter]キーを押してください。

▶コマンドやコードの中の▼について

本書に記載したコマンドの実行結果やコードは、誌面の都合上、ページをまたがって記載されていることがあります。その場合は▼の記号で、1つの実行結果・コードであることを表しています。

▶各項目の対応バージョンについて

各項目の先頭にはその項目に対応しているバージョンを記載しています。薄く表示されているマークは非対応を表しています。

サンプルファイルのダウンロードについて

本書のサンプルデータは、C&R研究所のホームページからダウンロードすることができます。本書のサンプルを入手するには、次のように操作します。

❶「http://www.c-r.com/」にアクセスします。

❷トップページ左上の「商品検索」欄に「174-0」と入力し、[検索]ボタンをクリックします。

❸検索結果が表示されるので、本書の書名のリンクをクリックします。

❹書籍詳細ページが表示されるので、[サンプルデータダウンロード]ボタンをクリックします。

❺下記の「ユーザー名」と「パスワード」を入力し、ダウンロードページにアクセスします。

❻「サンプルデータ」のリンク先のファイルをダウンロードし、保存します。

サンプルのダウンロードに必要な
ユーザー名とパスワード

ユーザー名 **psgr**
パスワード **5xf7w**

※ユーザー名・パスワードは、半角英数字で入力してください。また、「J」と「j」や「K」と「k」などの大文字と小文字の違いもありますので、よく確認して入力してください。

サンプルはZIP形式で圧縮してありますので、解凍してお使いください。なお、サンプルは、コマンドを実行する際にデータなどが必要となる場合のファイルになります。使い方などについては、サンプルが必要な項目のHINTをご確認ください。

CONTENTS

CHAPTER 01 PowerShellの基礎知識

CHAPTER 02 PowerShellの基本文法

CHAPTER 03 ヘルプ・コマンド・エイリアス

CHAPTER 12 リモート・セッション

CHAPTER 13 WSMan・WMI

CHAPTER 16 トランザクション

CHAPTER 01

PowerShellの基礎知識

PowerShellとは

PowerShellの概要

PowerShell（正式には「Windows PowerShell」）とは、Microsoftが提供するシステム管理用のコマンドラインシェル（対話的にOSを操作するためのソフトウェア）およびスクリプト言語です。「コマンドレット」と呼ばれるコマンド（単機能の命令）を使用して対話的に作業を行うことができます。また、コマンドレットを組み合わせて、複雑な処理をスクリプトとして記述して実行することができます。

▶標準装備だから気軽に始められる

PowerShellは、Windows Server 2008 R2、Windows 7から標準装備されているので、気軽に始めることができます。それ以前のバージョンでは、インストールすることで利用できます（32ページ参照）。

▶一貫したコマンドレット体型

PowerShellのコマンドレットは、標準的な名前付けパターン（[動詞]-[名詞]）を使用しているので、コマンドレットの動作を容易に推測できます。また、コマンドレットには、豊富なオプションが容易されおり、オプションパラメータを指定することで、コマンドレットの動作を変更することができます（39ページ参照）。

▶.NET Frameworkとの連携

PowerShellは、CLR（.NET Frameworkアプリケーションを実行するための仮想マシン）上で動作します。また、コマンドレットの戻り値（結果）は常に「オブジェクト」で、.NET Frameworkのクラスを呼び出すことができます。

▶データへのアクセスの共通化

PowerShellでは、ファイルシステム、レジストリ、デジタル署名、環境変数、エイリアス、スクリプト変数、関数、WS-Management構成情報（2.0以降）をドライブ（PSドライブ）として扱うことができます。これらのドライブには、ドライブレターだけでなく、文字列を使用してアクセスすることができます（53ページ参照）。

▶統合スクリプティング開発環境によるスクリプトの作成

PowerShell 2.0以降には、Windows PowerShell Integrated Scripting Environment（以降「ISE」）が標準装備されています。ISEは、スクリプトおよびモジュールを記述、実行、およびテストすることのできる統合開発環境です（31ページ参照）。

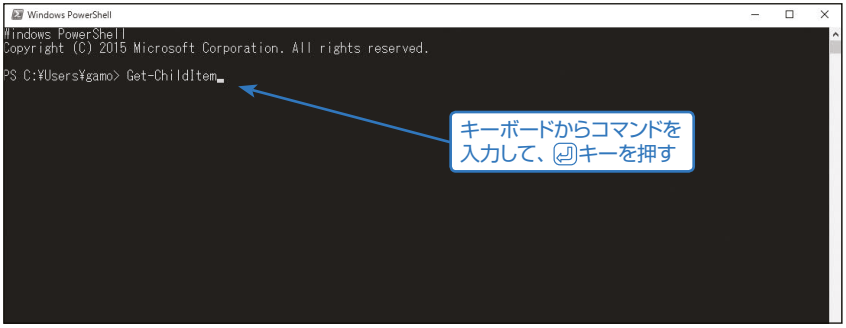
PowerShellコンソール/ISEについて

PowerShellには、コンソール版（PowerShellコンソール）とGUI版（ISE）があります。

▶コンソール版

コンソール版のPowerShellは、従来のコマンドプロンプトと同様のコマンドラインのシェルです。「PowerShellコンソール」と呼ぶこともあります。キーボードからコマンドを入力して、↵キーを押すことで、結果を得ることができます。

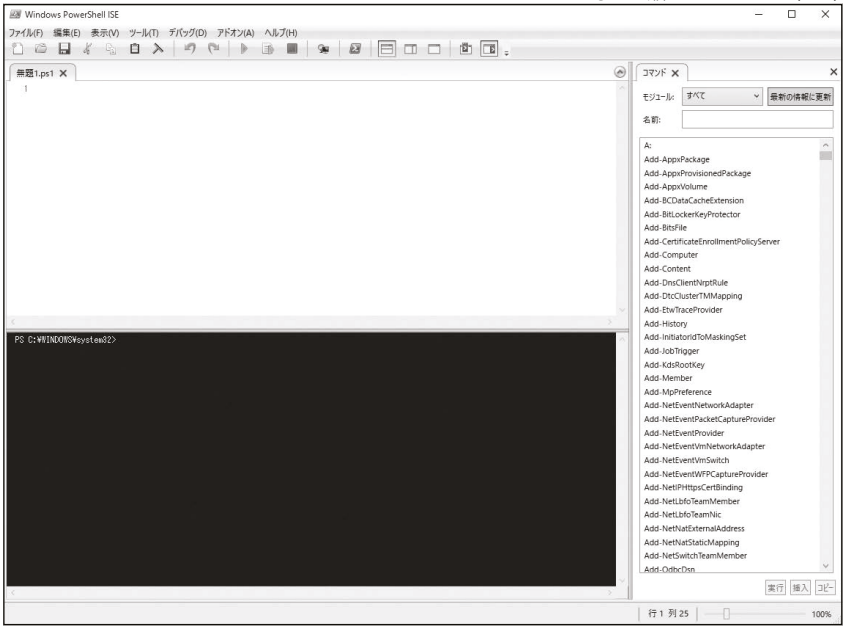
◉コンソール版のPowerShell



▶GUI版（ISE）

GUI版のPowerShellは、「ISE」（Windows PowerShell Integrated Scripting Environment）と呼ばれ、PowerShell2.0から搭載された統合開発環境です。スクリプトおよびモジュールを記述、実行、およびテスト（デバッグ）を行うとこができます。

◉GUI版のPowerShell（ISE）



PowerShellのインストール

PowerShellのインストールについて

PowerShellは、Windowsのバージョンによっては、標準ではインストールされていません。その場合は、PowerShell（Windows Management Framework）と対応する.NET Frameworkのパッケージをダウンロードしてインストールする必要があります。また、標準でインストールされている場合でも、上位バージョンをインストールしてアップグレードすることができます。

PowerShellをインストールする場合は、最新のサービスパック（SP）を適用し、インストール可能な最新バージョンのPowerShellをインストールすることをお勧めします。なお、環境によっては、古いバージョンのPowerShellをインストールしても、自動的に最新のPowerShellにアップグレードされる場合があります。

▶PowerShellとWindowsのバージョンの対応

PowerShellとWindowsのバージョンの対応は、次の表のとおりです。

OS	標準装備のPowerShellのバージョン	インストール可能なPowerShellのバージョン
Windows Server 2003/2003 R2	なし	1.0（SP1）、2.0（SP2）
Windows Server 2008	1.0	2.0（SP1）、3.0（SP2）
Windows Server 2008 R2	2.0	3.0/4.0（SP1）/5.0（SP1）
Windows Server 2012	3.0	4.0/5.0
Windows Server 2012 R2	4.0	5.0
Windows XP x64	なし	1.0
Windows XP	なし	1.0（SP2）、2.0（SP3）
Windows Vista	なし	1.0、2.0（SP1）
Windows 7	2.0	3.0/4.0（SP1）/5.0（SP1）
Windows 8	3.0	なし
Windows 8.1	4.0	5.0
Windows 10	5.0	なし

※（）内は、インストールする際に適応する必要があるサービスパック

Windows Server 2008では、サーバーマネージャーの「機能の追加」でPowerShellを有効化する必要があります。また、Windows Server 2008 R2で、PowerShellのGUI環境（ISEなど）を利用するには、サーバーマネージャーの「機能の追加」で、「.NET Framework 3.5.1」と「Windows PowerShell Integrated Scripting Environment（ISE）」を有効にする必要があります。

各バージョンのPowerShell（Windows Management Framework）の入手先（ダウンロード先）のURLは、次の表のとおりです。

バージョン	URL
1.0（Vista用）	http://support.microsoft.com/kb/928439
2.0	http://support.microsoft.com/kb/968929
3.0	https://www.microsoft.com/en-us/download/details.aspx?id=34595
4.0	https://www.microsoft.com/ja-jp/download/details.aspx?id=40855
5.0	https://www.microsoft.com/en-us/download/details.aspx?id=46889

※2015年8月現在、XPおよびServer 2003用のPowerShell 1.0は提供されていません。

▶ PowerShellと.NET Frameworkの対応

PowerShellを動作させるには、各バージョンに対応する.NET Frameworkをあらかじめインストール必要があります。PowerShellと.NET Frameworkのバージョンは、次の表のとおりです。

PowerShellのバージョン	.NET Framework
1.0	2.0以上
2.0	2.0以上（GUI環境の動作には3.5以上）
3.0	4.0以上
4.0	4.5以上
5.0	4.5以上

各バージョンの.NET Frameworkの入手先（ダウンロード先）のURLは、次の表のとおりです。

バージョン	URL
2.0	http://www.microsoft.com/ja-jp/download/details.aspx?id=6523
3.5	http://www.microsoft.com/ja-jp/download/details.aspx?id=21
4.0	http://www.microsoft.com/ja-jp/download/details.aspx?id=17718
4.5	http://www.microsoft.com/ja-jp/download/details.aspx?id=30653

PowerShellの起動

[スタート]メニューからの起動(Windows Vista/7/10/Server 2008)

[スタート]メニューからPowerShellを起動するには、次のように操作します。

❶ [スタート]ボタンをクリックし、[すべてのアプリ]→[Windows PowerShell]→[Windows PowerShell](Vista/7/Server 2008は[すべてのプログラム]→[アクセサリ]→[Windows PowerShell])を選択します。

❷ コンソール版の場合は[Windows PowerShell]を、ISE(GUI版)の場合は[Windows Power Shell ISE]をクリックします(「(x86)」と付いているメニューをクリックすると、32bit版が起動)。

◉Windows 10の例

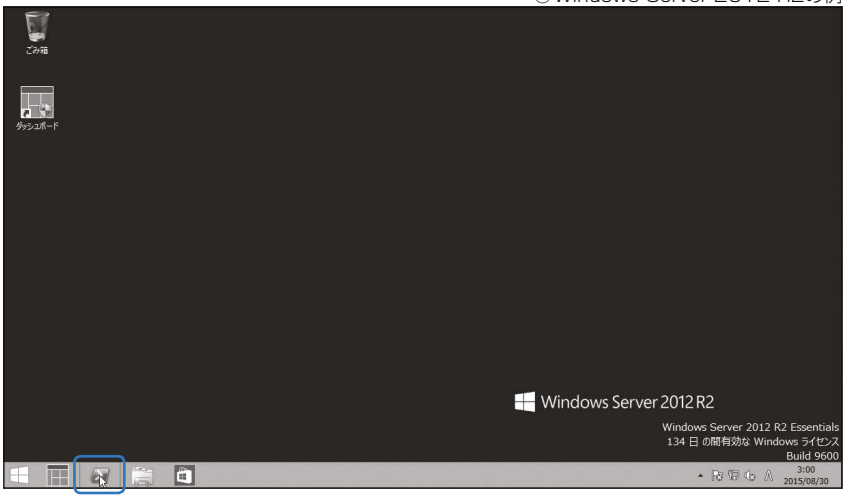
なお、PowerShellを管理者として起動するには、操作❷のタイミングで[Windows Power Shell](ISEは[Windows PowerShell ISE)を右クリックし、[管理者として実行](Vista/7/Server 2008は、[管理者として実行(A)])を選択します。

タスクバーからの起動(Windows 7/Server 2008 R2以降)

タスクバーから、PowerShell(コンソール版)を起動するには、タスクバーの[Windows PowerShell]アイコンをクリックします。ISE(GUI版)を起動するには、タスクバーの[Windows PowerShell]アイコンを右クリックし、[Windows PowerShell ISE]をクリックします。また、管理者として実行するには、タスクバーの[Windows PowerShell]アイコンを右クリックし、[管理者として実行する](ISEの場合は、[ISEを管理者として実行する])をクリックします。

なお、Windows 7/8/8.1では、環境によってはタスクバーに[Windows PowerShell]アイコンがない場合があります。

◉Windows Server 2012 R2の例



スタート画面からの起動(Windows 8/Server 2012以降)

スタート画面からPowerShell(コンソール版)を起動するには、アプリ一覧を表示(Windows 8/Server 2012では、スタート画面の空欄を右クリック、Windows 8.1ではマウスを動かたときに画面左下に表示される[↓]をクリック)し、[Windows PowerShell]をクリックします。Server 2012では、スタート画面の[Windows PowerShell]タイルをクリックして、PowerShell(コンソール版)を起動することもできます。

Windows 8でアプリ一覧にISEや32bit版のPowerShellを起動するためのアイコン(タイル)を常に表示するには、スタート画面もしくはアプリ一覧のチャームから[設定]→[タイル]を選択し、[管理ツールを表示する]を[はい]に設定します。

◉Windows 8.1の例

なお、管理者として実行するには、[Windows PowerShell]（ISEは[Windows Power Shell ISE]）を右クリックします。Windows 8/Server 2012の場合は、画面下に表示されるアプリバーの[管理者として実行]をクリックします。Windows 8.1の場合は、表示されるショートカットメニューから[管理者として実行（A）]を選択します。

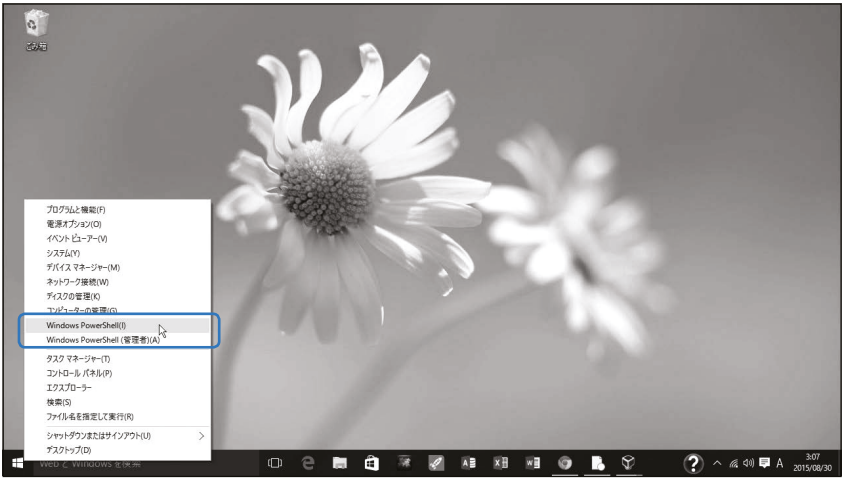
クイックアクセスメニューから起動する（Windows 8.1/10）

クイックアクセスメニューからPowerShell（コンソール版）を起動するには、次のように操作して、クイックアクセスメニューのコマンドプロンプトをPowerShellに置き換えます。

❶ タスクバーを右クリックし、[プロパティ(R)]を選択する

❷「ナビゲーション」タブを選択し、[右下隅を右クリックするか、Windowsキー ＋ Xキーを押したときに表示されるメニューで、コマンドプロンプトをWindows PowerShellに置き換える(C)]のチェックボックスをONにし、[OK]ボタンをクリックします。

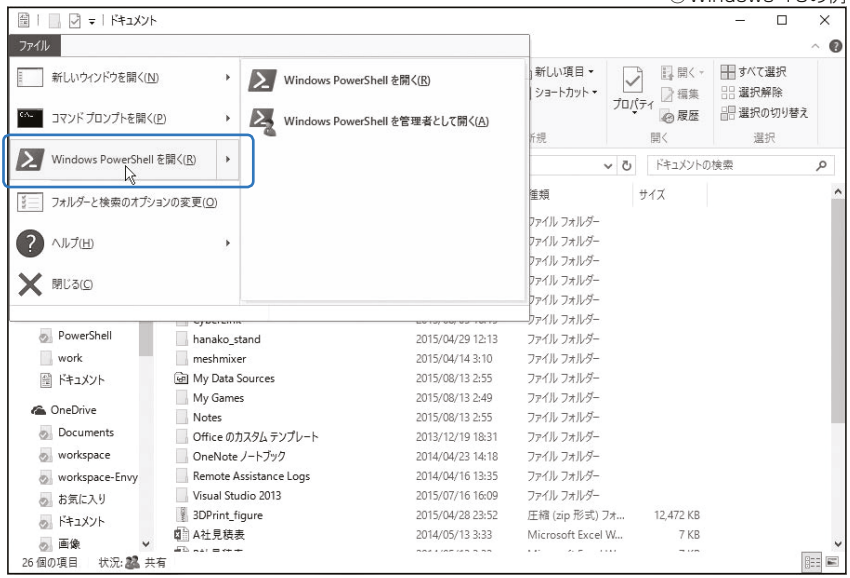
◉コマンドプロンプトをPowerShellに置き換えたクイックアクセスメニュー



エクスプローラから起動する（Windows 8/Server 2012以降）

エクスプローラからPowerShell（コンソール版）を起動するには、任意のドライブ、フォルダを開いて、[ファイル]→[Windows PowerShellを開く(R)]（または[ファイル]→[Windows PowerShellを開く(R)]→[Windows PowerShellを開く(R)]）をクリックします。管理者としてWindows PowerShellを開く場合は、[ファイル]→[Windows PowerShellを開く(R)]→[管理者としてWindows PowerShellを開く(A)]を選択します。

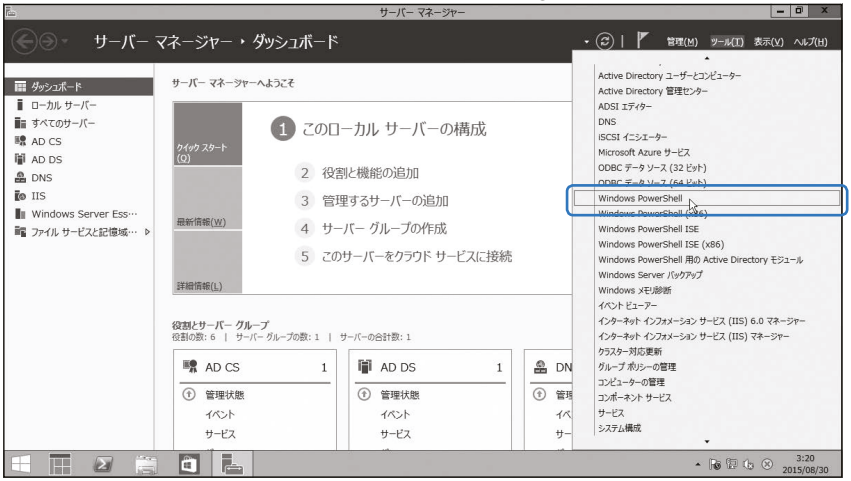
◉Windows 10の例



サーバーマネージャーから起動する(Windows Server 2012以降)

サーバーマネージャーから、[ツール(T)]メニューから[Windows PowerShell](32bit版は[Windows PowerShell (x86)])を選択します。ISE(GUI版)を起動するには、[ツール(T)]メニューから[Windows PowerShell ISE](32bit版は[Windows PowerShell ISE(x86)])を選択します。サーバーマネージャーからは管理したいサーバーを選択し、そのサーバーにリモート接続した状態でPowerShellを起動することができます。

◉Windows Server 2012 R2の例



コマンドレットについて

コマンドレットとは

コマンドレットとは、PowerShellで利用可能なコマンド（命令）のことです。コマンドレットは単独で実行することも、パイプラインで複数を組み合わせて実行することができます（49ページ参照）。また、複数のコマンドレットの組み合わせをスクリプトとして記述することができます。各バージョンのPowerShellに標準で用意されているコマンドレット（PowerShellのコアモジュールに含まれるコマンドレット）の数は、次の表のとおりです。なお、3.0、4.0、5.0は「Core」「Diagnostic」「Host」「Management」「Security」「Utility」「WS-Management」モジュールのコマンドレットの総数になります。

バージョン	コマンドレット数
1.0	129
2.0	236
3.0	259
4.0	262
5.0	275

これ以外にもモジュールにて、多数のコマンドレットが提供されます。利用可能なコマンドレットを調べるには、「Get-Command」コマンドレットを使います（129ページ参照）。

コマンドレットの命名規則

PowerShellのコマンドレットは、動作を容易に推測できるように、次のような標準的な名前付けパターンを使用しています。

```
[動詞]-[名詞]
```

たとえば、「Get-Location」は、「Location（ロケーション）をGet（取得）する」という意味で、文字通り、ロケーション（カレントディレクトリ）を取得するコマンドレットになります。なお、PowerShellでは、大文字・小文字を区別しないので、「get-location」や「Get-location」と入力、または記述しても構いません。

コマンドレットのパラメータについて

コマンドレットのパラメータ

コマンドレットには、複数のパラメータが指定される場合があります。「Get-Help」コマンドレットを実行すると、コマンドレットのすべてのパラメータが表示されます。

パラメータに関する追加情報は、パラメータの説明およびパラメータ属性テーブルにあります。パラメータの詳細な情報を表示するには、「Get-Help」の「Full」パラメータまたは「Parameter」パラメータを使用します（120ページ参照）。

ヘルプの構文

PowerShellのコマンドレットヘルプ、ヘルプトピック、およびその他のドキュメントでは、構文の説明において、コマンドレットで次の表記法が使用されます。

```
<コマンドレット名> -<必須パラメータ名> <必須パラメータ値>
            [-<オプションパラメータ名> <オプションパラメータ値>]
            [-<オプションスイッチパラメータ>]
            [-<オプションパラメータ名>] <必須パラメータ値>
```

たとえば、「New-Alias」コマンドレットの構文は次のとおりです。

```
New-Alias [-Force] [-PassThru] [-Scope <string>]
   [-Option {None | ReadOnly | Constant | Private | AllScope}]
   [-Description <string>] [-Name] <string> [-Value] <string>
   [-confirm] [-whatif] [<CommonParameters>]
```

なお、この構文では、読みやすさのために大文字が使用されていますが、PowerShellでは大文字と小文字は区別されません。

パラメータは順に表示されます。パラメータの順序は、そのパラメータ名がオプションである場合にのみ重要です。コマンドレットの使用時にパラメータ名を指定しない場合、PowerShellでは位置および型に基づいてパラメータに値が割り当てられます。

パラメータ名の前には「-」（ハイフン）が付きます。スイッチパラメータは、値の型を伴わずに表示されます。その他のパラメータは、必須引数のMicrosoft .NET Framework型または使用可能な値の列挙とともに表示されます。

「<>」（山カッコ）はプレースホルダーテキストを示します。このテキストでは、項目の種類（文字列やオブジェクトなど）を記述することができます。このテキストは1つまたは複数の共通パラメータのプレースホルダーである場合もあります。

「[]」（角カッコ）はオプションの項目を示します。パラメータが省略可能であるか、必須パラメータの名前が省略可能であることを示します。スイッチパラメータは、常に省略可能です。

「｛｝」（波カッコ）は、列挙型の値の前後に表示されます。「New-Alias」コマンドレットの例では、「Option」パラメータには、一覧表示されているいずれの値（「None」「ReadOnly」「Constant」「Private」「AllScope」のいずれか）を指定することもできます。

オプションの項目

オプションの項目は、「[]」（角カッコ）で囲まれています。たとえば、「New-Alias」コマンドレットの構文では、「Scope」パラメータはオプションです。これはパラメータ名と型を「[]」が囲む構文で示されます。

```
    [-Scope <string>]
```

たとえば、次の2つの「New-Alias」コマンドレットの例は、どちらも正しい使用方法です。

```
New-Alias -Name utd -Value Update-TypeData
New-Alias -Name utd -Value Update-TypeData -Scope global
```

パラメータの値が必須であっても、パラメータ名は省略可能の場合があります。これは、「New-Alias」コマンドレットの次の例のように、パラメータ名を「[]」で囲み、パラメータの型は「[]」で囲まない構文で示します。

```
    [-Name] <string> [-Value] <string>
```

たとえば、次の4つのコマンドの結果は同じになります。

```
New-Alias -Name utd -Value Update-TypeData
New-Alias -Name utd Update-TypeData
New-Alias utd -ValueUpdate-TypeData
New-Alias utd Update-TypeData
```

入力時にパラメータ名がステートメントに含まれていない場合、PowerShellは、その引数の位置を使用して値をパラメータに割り当てようとします。たとえば、次の例は、「Value」パラメータが指定されていないので不完全です。

```
New-Alias utd
```

「New-Alias」コマンドレットでは、「Name」および「Value」パラメータの両方に値を指定する必要があります。

構文の例では、「[]」は.NET Framework型に対する名前付けおよびキャストにも使用されています。このような場合、「[]」は要素が省略可能であることを示すわけではありません。

配列引数

「[]」(角カッコ)は配列を表すためにも使用されています。たとえば、「Restart-Service」コマンドレットの構文は、次のようになります。

```
Restart-Service [-Name] <string[]> [-Include <string[]>]
      [-Exclude <string[]>] [-Force] [-PassThru] [-Confirm] [-WhatIf]
      [<CommonParameters>]
```

「Name」パラメータには、再起動するサービスの名前が引数として必要です。この構文で引数は「string[]」と、配列で指定できることを表しているので、次のようにサービスの名前をコンマ区切り一覧で指定することができます。

```
Restart-Service RasAuto, RasMan, RemoteAccess
```

SECTION-006

共通パラメータ/リスク軽減パラメータについて

共通パラメータ

共通パラメータ(CommonParameters)は、すべてのコマンドレットで使用できるパラメータです。共通パラメータはどのコマンドレットでも使用できますが、一部のコマンドレットに対しては効力がない場合があります。共通パラメータの詳細は、次のとおりです(()内はエイリアス)。

▶Verbose(vb)

「Verbose(vb)」は、コマンドによって実行される操作の詳細メッセージを表示するかどうかを指定します。このパラメータはコマンドに「Write-Verbose」コマンドレットが含まれている場合に有効になります。「$true」、または値を指定しない場合は、詳細メッセージを表示します。「$false」(既定値)の場合は、詳細メッセージを表示しません。

構文は次のとおりです。

```
-Verbose[:{$true | $false}]
```

▶Debug(db)

「Debug(db)」は、コマンドによって実行される操作のデバッグメッセージ(プログラマレベルの詳細情報)を表示するかどうかを指定します。このパラメータはコマンドに「Write-Debug」コマンドレットが含まれている場合に有効になります。「$true」または値を指定しない場合は、デバッグメッセージを表示します。「$false」の場合は、「$DebugPreference」の値が既定値(SilentlyContinue)でない場合にデバッグメッセージを表示しません。

構文は次のとおりです。

```
-Debug[:{$true | $false}]
```

▶WarningAction(wa)

「WarningAction(wa)」は、警告が発生したときの処理方法を指定します。このパラメータはコマンドに「Write-Warning」コマンドレットが含まれている場合に有効になります。

構文は次のとおりです。

```
-WarningAction[:{SilentlyContinue | Continue | Inquire | Stop}]
```

設定可能な値は、次のとおりです。

値	説明
SilentlyContinue	警告メッセージが表示されないようにし、コマンドの実行を継続する
Continue	警告メッセージを表示し、コマンドの実行を継続する(既定値)
Inquire	警告メッセージを表示し、実行を継続する前に確認メッセージを表示する
Stop	警告メッセージを表示し、コマンドの実行を停止する

▶ WarningVariable(wv)

「WarningVariable(wv)」は、コマンドに関する警告を変数に格納します。変数の内容に警告を追加するには、変数名の前に「+」(プラス記号)を入力します。

構文は次のとおりです。

```
-WarningVariable [+]<変数名>
```

▶ ErrorAction(ea)

「ErrorAction(ea)」は、エラーが発生したときの処理方法を指定します。このパラメータはコマンドに「Write-Error」コマンドレットが含まれている場合に有効になります。

構文は次のとおりです。

```
-ErrorAction[:{Continue | Ignore | Inquire | SilentlyContinue | Stop | Suspend }]
```

設定可能な値は、次のとおりです。

値	説明
Continue	エラーメッセージを表示し、コマンドの実行を継続する(既定値)
Ignore(V3〜)	エラーメッセージを表示せずに、コマンドの実行を継続する
Inquire	エラーメッセージを表示し、実行を継続する前に確認メッセージを表示する
SilentlyContinue	エラーメッセージが表示されないようにし、コマンドの実行を継続する
Stop	エラーメッセージを表示し、コマンドの実行を停止する
Suspend(V4〜)	エラーメッセージを表示し、コマンドを中断する(ワークフローのみサポート)

▶ ErrorVariable(ev)

「ErrorVariable(ev)」は、コマンドに関するエラーメッセージを指定された変数および「$Error」自動変数に格納します。変数の内容にエラーメッセージを追加するには、変数名の前に「+」(プラス記号)を入力します。

構文は次のとおりです。

```
-ErrorVariable [+]<変数名>
```

▶ OutVariable(ov)

「OutVariable(ov)」は、コマンドからの出力オブジェクトを指定された変数に格納し、コマンドラインに表示します。変数に出力を追加するには、変数名の前に「+」(プラス記号)を入力します。

構文は次のとおりです。

```
-OutVariable [+]<変数名>
```

▶ OutBuffer(ov)

「OutBuffer(ov)」は、パイプライン経由でオブジェクトが送信されるまでに、バッファーに蓄積するオブジェクト数を決定します。

構文は次のとおりです。

```
-OutBuffer <整数値>
```

▶ PipelineVariable(pv)

「PipelineVariable(pv)」は、現在のパイプラインの要素の値を指定された変数に格納します。

構文は次のとおりです。

```
-PipelineVariable <変数名>
```

▶ InformationAction(infa)

「InformationAction(infa)」は、情報メッセージの処理方法を指定します(V5のみ)。このパラメータはコマンドに「Write-Information」コマンドレットが含まれている場合に有効になります。

構文は次のとおりです。

```
-InformationAction [:{SilentlyContinue | Continue | Inquire | Stop | Ignore}]
```

設定可能な値は、次のとおりです。

値	説明
Continue	情報メッセージを表示し、コマンドの実行を継続する(既定値)
Ignore	情報メッセージを表示せずに、コマンドの実行を継続する
Inquire	情報メッセージを表示せずに、実行を継続する前に確認メッセージを表示する
SilentlyContinue	情報メッセージを表示されないようにし、コマンドの実行を継続する
Stop	情報メッセージを表示し、コマンドの実行を停止する

▶ InformationVariable(iv)

「InformationVariable(iv)」は、コマンドに関する情報メッセージを指定された変数に格納します。変数の内容に情報メッセージを追加するには、変数名の前に「+」(プラス記号)を入力します(V5のみ)。

構文は次のとおりです。

```
-InformationVariable [+]<変数名>
```

リスク軽減パラメータ

リスク軽減パラメータは、コマンド実行時のリスクを回避または軽減するためのパラメータです。リスク軽減パラメータは、次のとおりです（（）内はエイリアス）。

▶ WhatIf（wi）

「WhatIf（wi）」は、コマンドを実行する代わりにコマンドの結果を説明するメッセージを表示します。コマンドを実際に実行することなく、コマンドの動作を確認できます。「$true」または値を指定しない場合は「WhatIf」の動作を有効にし、「$false」の場合は、「$WhatIf Preference」変数の値が「1」であった場合に発生する自動的な「WhatIf」の動作を抑止します。

構文は次のとおりです。

```
-WhatIf[:{$true | $false}]
```

▶ Confirm（cf）

「Confirm（cf）」は、コマンドを実行する前に確認メッセージを表示します。「$true」または値を指定しない場合は、「Confirm」の動作を有効にし、「$false」の場合は、「$Confirm Preference」の値がコマンドレットの推定リスク以下であった場合に表示される自動確認メッセージを抑止します。

構文は次のとおりです。

```
-Confirm[:{$true | $false}]
```

エイリアスについて

エイリアスとは

エイリアスとは、コマンドレットやコマンドの要素（関数、スクリプト、ファイル、実行可能ファイルなど）に使用する代替名です。PowerShellのすべてのコマンドでは、エイリアスをコマンド名の代わりに使用できます。エイリアスを作成するには、「New-Alias」コマンドレットを使用します（136ページ参照）。

また、PowerShellには、「Set-Location」コマンドレットに関連付けられた「cd」と「chdir」や、「Get-ChildItem」コマンドレットに関連付けられた「ls」と「dir」などの一連の組み込みエイリアスが用意されています。定義されているすべてのエイリアス（組み込みエイリアスを含む）を取得するには、「Get-Alias」を使用します（134ページ参照）。

組み込みエイリアスの例（抜粋）を下表に示します。

コマンドレット	エイリアス
Get-Content	cat、gc
Set-Location	cd、chdir、sl
Copy-Item	copy、cp、cpi
Get-ChildItem	dir、gci、ls
Get-Location	gl、pwd
Move-Item	mi、move、mv
Remove-Item	rd、ri、rm、rmdir
Rename-Item	ren、rni

1つのコマンドレットには、複数の組み込みエイリアスが関連付けされている場合があります。組み込みエイリアスには、そのコマンドレットの同等の機能を持つコマンドプロンプトのコマンド（「cd」「dir」など）やUNIXのシェルコマンド（「cat」「ls」など）、またはコマンドレットの動詞、名詞部分から一部の文字列を抜粋（多くの場合、それぞれの頭文字）した名前（「gc」「sl」「gci」など）が使用されています。

リダイレクトについて

リダイレクトとは

リダイレクトとは、コマンド（コマンドレット）の出力先を別の場所に変更することです。既定では、PowerShellのコマンド出力は、PowerShellコンソールに送信されます。ただし、出力をテキストファイルに送信することも、エラー出力を通常の出力ストリームにリダイレクトすることもできます。

リダイレクトの方法

PowerShellでは、次の方法で出力をリダイレクトできます。

▶「Out-File」コマンドレットを使用する

「Out-File」コマンドレットを使用し、コマンド出力をテキストファイルに送信します。通常、「Out-File」コマンドレットは、そのパラメータ（「Encoding」「Force」「Width」「NoClobber」など）を使用する必要がある場合に使用します（272ページ参照）。

▶「Tee-Object」コマンドレットを使用する

「Tee-Object」コマンドレットは、コマンド出力をテキストファイルに送信し、その後でパイプラインに送信します（334ページ参照）。

▶リダイレクト演算子を使う

PowerShellのリダイレクト演算子は、次のとおりです。

演算子	説明	例
>	指定したファイルに出力を送信する	Get-Process > process.txt
>>	指定したファイルの内容に出力を追加する	dir *.ps1 >> scripts.txt
n>	指定したファイルにエラーを送信する	Get-Process none 2> errors.txt
n>>	指定したファイルの内容にエラーを追加する	GetProcess none 2>> save-errors.txt
n>&1	正常な出力ストリームにエラーを送信する	Get-Process none, powershell 2>&1

上記の表中の「n」には、次のストリーム番号を指定できます。

ストリーム番号	ストリーム
2	エラー
3（V3～）	警告
4（V3～）	詳細
5（V3～）	デバッグ
*（V3～）	すべての出力

リダイレクト演算子の構文は次のとおりです。

```
<入力コマンド> <演算子> [<パス>¥]<ファイル名>
```

指定したファイルがすでに存在する場合、データを追加しないリダイレクト演算子（「>」および「2>」）を使用すると、ファイルの現在の内容が警告なしに上書きされます。ただし、ファイルが読み取り専用ファイル、非表示のファイル、システムファイルのいずれかの場合、リダイレクトは失敗します。追加リダイレクト演算子（「>>」および「2>>」）を使用した場合、読み取り専用ファイルには書き込みは行われませんが、システムファイルと非表示のファイルには内容が追加されます。

読み取り専用ファイル、非表示のファイル、またはシステムファイルに対し、内容のリダイレクトを強制的に行うには、「Out-File」コマンドレットとその「Force」パラメータを使用します。リダイレクト演算子は、ファイルへの書き込み時にUnicodeエンコードを使用します。ファイルのエンコードが異なる場合、出力が適切にフォーマットされないことがあります。内容を非Unicodeファイルにリダイレクトするには、「Out-File」コマンドレットとその「Encoding」パラメータを使用します（272ページ参照）。

パイプラインについて

パイプラインとは

パイプラインとは、「|」(パイプライン演算子)を使用して連結された一連のコマンドです。各パイプライン演算子は、直前のコマンドの結果を次のコマンドに渡します。

パイプラインを使用すると、一方のコマンドにより出力されたオブジェクトを他方のコマンドの入力として使用できます。さらに、そのコマンドの出力をさらに別のコマンドに渡すこともできます。その結果、一連の単純なコマンドから、非常に強力なコマンドチェーンを作成できます。

パイプラインの例は、次のようになります。

```
コマンド1 | コマンド2 | コマンド3
```

パイプラインでは、左から右に、表示された順序でコマンドが処理されます。上の例では、コマンド1により出力されるオブジェクトがコマンド2に渡されます。コマンド2は、これらのオブジェクトを処理し、コマンド3に渡します。コマンド3は、これらのオブジェクトを処理し、パイプラインでの次の処理に渡します。このパイプラインにはこれ以上コマンドが存在しないので、結果がコンソールに表示されます。

なお、パイプラインによる処理(一方のコマンドの出力に含まれるオブジェクトが他方のコマンドに渡される処理)を、「パイプ処理」といいます。

PowerShellでのパイプラインの使用

PowerShellのコマンドレットは、パイプラインでの使用を想定して設計されています。たとえば、一般的な使用例では、「Get」コマンドレットで取得した結果をパイプ処理し、同じ名詞の処理コマンドレット(「Set」「Start」「Stop」「Rename」コマンドレットなど)に渡すことができます。

たとえば、「Get-Service」コマンドレットからすべてのサービスをパイプ処理して、「Start-Service」、または「Stop-Service」コマンドレットに渡すことができます。次のパイプラインを実行すると、コンピュータ上でWMIサービスが起動されます。

```
Get-Service wmi | Start-Service
```

PowerShellプロバイダーのオブジェクトの取得または設定を行うコマンドレット(「Item」および「ItemProperty」コマンドレットなど)も、パイプラインで使用できるように設計されています。たとえば、PowerShellレジストリプロバイダーの「Get-Item」または「Get-ChildItem」コマンドレットの結果をパイプ処理して、「New-ItemProperty」コマンドレットに渡すことができます。

たとえば、「HKLM:¥Software¥MyCompany」レジストリキーに「8124」という値を持つ新しいレジストリエントリ「NoOfEmployees」を追加するには、次のように入力します。

```
Get-Item -Path HKLM:¥Software¥MyCompany | New-Itemproperty -Name NoOfEmployees -Value 8124
```

「Get-Member」「Where-Object」「Sort-Object」「Group-Object」および「Measure-Object」などの多くのユーティリティコマンドレットは、ほぼパイプラインにのみ使用されます。これらのコマンドレットには、あらゆるオブジェクトをパイプ処理で渡すことができます。たとえば、コンピュータ上のすべてのプロセスをパイプ処理して、「Sort-Object」コマンドレットに渡し、それらをハンドル数に基づいて並べ替えることができます。

```
Get-Process | Sort-Object -Property handles
```

また、「Format-List」および「Format-Table」などの書式設定用コマンドレット、「Export-Clixml」および「Export-CSV」などの「Export」コマンドレット、「Out-Printer」などの「Out」コマンドレットにも、任意のオブジェクトをパイプ処理で渡すことができます。たとえば、「Winlogon」プロセスを「Format-List」コマンドレットにパイプ処理して渡し、プロセスのすべてのプロパティを一覧表示できます。

```
Get-Process winlogon | Format-List -Property *
```

PowerShellでのパイプラインの動作

オブジェクトがパイプ処理される（一方のコマンドの出力に含まれるオブジェクトが他方のコマンドに渡される）と、PowerShellは、パイプ処理されたオブジェクトを受け取り側のコマンドレットのいずれかのパラメータに関連付けます。このため、PowerShellには入力オブジェクトをコマンドレットパラメータに関連付ける（パラメータをバインドする）コンポーネントが存在し、このコンポーネントは、次の条件を満たすパラメータを検索します。

- パイプラインからの入力を許可するパラメータ（すべてのパラメータが許可するわけではないため）
- 渡されるオブジェクトの型またはオブジェクトの変換が可能な目的の型を受け入れるパラメータ
- コマンドでまだ使用されていないパラメータ

たとえば、「Start-Service」コマンドレットには多くのパラメータが存在しますが、パイプラインの入力を許可するのは「Name」および「InputObject」の2つのパラメータのみです。「Name」パラメータは文字列を受け取り、「InputObject」パラメータはサービスオブジェクトを受け取ります。このため、文字列およびサービスオブジェクト（またはサービスオブジェクトに変換可能なプロパティを持つオブジェクト）をパイプ処理して、「Start-Service」に渡すことができます。

PowerShellのパラメータバインドコンポーネントが、パイプ処理されたオブジェクトを受け取り側のコマンドレットパラメータに関連付けることができない場合、コマンドは失敗し、見つからなかったパラメータ値の入力を求めるメッセージがPowerShellよって表示されます。

なお、「Get-Help」コマンドレットを「Full」または「Parameter」パラメータと一緒に使用すると、パイプライン入力を許可するコマンドレットのパラメータを特定することができます（120ページ参照）。

ワイルドカードについて

ワイルドカードとは

ワイルドカード(ワイルドカード文字)とは、任意の文字を表すための記号です。ワイルドカードを使うと、特定のパターンに一致するファイルだけを検索することができます。PowerShellでサポートされるワイルドカードは、次のとおりです。

ワイルドカード	説明	例	一致する	一致しない
*	0個以上の文字と一致する	a*	A、ag、Apple	banana
?	指定された位置にある1文字と正確に一致する	?n	an、in、on	ran
[ ]	指定された文字と一致する	[bc]ook	book、cook	hook
[-]	ある範囲の文字と一致する	[a-l]ook	book、cook、look	took

大部分のコマンドレットは、一部のパラメータでワイルドカード文字を受け入れます。各コマンドレットのヘルプトピックには、ワイルドカード文字を使用できるパラメータについての説明があります(該当するパラメータがある場合)。ワイルドカード文字を使用できるパラメータでは、大文字と小文字は区別されません。たとえば、「?n」は「An」「an」「In」「in」「On」「on」を返します。

また、1つのパラメータ内で異なるワイルドカード文字を組み合わせることができます。たとえば、「C:¥Techdocs」ディレクトリにあるファイルの中で、「a」から「l」までの文字で始まるすべての「.txt」ファイルを表示するには、次のようにコマンドを入力します。

```
Get-ChildItem c:¥techdocs¥[a-l]*.txt
```

このコマンドは、文字範囲を表すワイルドカード「[a-l]」を使用して、ファイル名が「a」から「l」の範囲内の文字で始まる必要があることを示しています。さらに、ワイルドカード「*」を使用して、最初の文字とファイル拡張子の間には任意の文字列があってよいことを示すプレースホルダーを提供しています。

スクリプトについて

スクリプトとは

スクリプトとは、1つまたは複数のPowerShellコマンド（コマンドレット）を含むテキストファイルです。PowerShellのスクリプトのファイル名拡張子は「.ps1」です。スクリプトを作成すると、スクリプトのパスとファイル名を入力するだけでコマンドを実行でき、コマンドを再入力する手間を省くことができます。また、簡単に他のユーザーとコマンドを共有できます。

スクリプトは、#Requires特殊コメント、パラメータの使用、Dataセクションのサポート、セキュリティを確保するためのデジタル署名など、さまざまな機能を備えています。スクリプトやスクリプト内の任意の関数に対するヘルプトピックを記述することもできます。

スクリプトの作成方法

スクリプトを作成するには、テキストエディタ（メモ帳など）またはスクリプトエディタ（ISEなど）を起動します。コマンドを入力し、有効なファイル名と「.ps1」ファイル名拡張子を付けたファイルに保存します。スクリプトには、単独のコマンドに加えて、パイプライン、関数、制御構造（「If」ステートメントや「For」ループなど）を使用するコマンドを含めることができます。

スクリプトの実行方法

スクリプトを実行する前に、PowerShellの既定の実行ポリシーを変更する必要があります。既定の実行ポリシーである「Restricted」では、ローカルコンピュータ上に作成したスクリプトを含むすべてのスクリプトの実行が禁止されます（442ページ参照）。

PowerShellのセキュリティ機能により、Windowsエクスプローラでスクリプトアイコンをダブルクリックしても、スクリプトは実行されません。スクリプトを実行するには、スクリプトファイルの完全な名前を含む完全なパスを入力します。たとえば、「C:¥Scripts」ディレクトリにある「ServicesLog」スクリプトを実行するには、次のように入力します。

```
PS C:¥work> c:¥scripts¥ServicesLog.ps1
```

現在のディレクトリ（カレントディレクト）にあるスクリプトを実行するには、現在のディレクトリへのパスを入力するか、または現在のディレクトリを表すドットに続けて円マーク（¥）を使用します。たとえば、カレントディレクトにある「ServicesLog.ps1」スクリプトを実行するには、次のように入力します。

```
PS C:¥work> .¥ServicesLog.ps1
```

プロバイダーについて

プロバイダーとは

プロバイダー（PowerShellプロバイダー、PSプロバイダー）とは、特殊なデータストアのデータ（PowerShellのエイリアス、関数、変数など）をPowerShellで利用できるようにし、それらのデータの表示と管理を可能にするためのプログラムです。

プロバイダーが公開するデータはドライブに表示されます。このデータには、ハードディスクドライブ上のデータにアクセスする場合と同じように、パスを使用してアクセスします。プロバイダーがサポートする組み込みコマンドレットを使用して、プロバイダーのドライブにあるデータを管理できます。また、このデータ用に特別に設計されたカスタムコマンドレットも使用できます。

組み込みプロバイダー

PowerShellには、さまざまなデータストアへのアクセスに使用できる、組み込みプロバイダーのセットが用意されています。

プロバイダー	ドライブ	データストア
Alias	Alias:	PowerShellエイリアス
Certificate	Cert:	デジタル署名用のX509証明書
Environment	Env:	Windows環境変数
FileSystem	C:、D:など（システムによって異なる）	ファイルシステムのドライブ、ディレクトリ、およびファイル
Function	Function:	PowerShellの関数
Registry	HKLM:、HKCU:	Windowsレジストリ
Variable	Variable:	PowerShellの変数
WSMan	WSMan:	WS-Management構成情報

また、独自のPowerShellプロバイダーを作成したり、他のユーザーが開発したプロバイダーをインストールしたりすることもできます。セッションで使用可能なプロバイダーの一覧を表示するには、「Get-PSProvider」コマンドレットを使います（240ページ参照）。

「Alias」プロバイダー

「Alias」プロバイダー（エイリアスプロバイダー）は、PowerShellのエイリアスおよびエイリアスで表される値へのアクセスを提供します。「Alias」プロバイダーを使用すると、PowerShellでエイリアスを取得、追加、変更、クリア、および削除できます。「Alias」プロバイダーは、「Alias:」ドライブにデータストアを公開します。

▶コマンドレット

「Alias」プロバイダーは、「Invoke-Item」コマンドレットを除き、名前に「Item」という名詞が含まれるすべてのコマンドレットをサポートします。また、「Get-Content」および「Set-Content」コマンドレットもサポートします。ただし、「ItemProperty」という名詞を含むコマンドレットおよび、あらゆるコマンドレットの「Filter」パラメータはサポートされません。なお、PowerShellには、エ

イリアスの表示や変更を行うために設計された、次のコマンドレットが用意されています。

コマンドレット	解説
Get-Alias	134ページ
New-Alias	136ページ
Set-Alias	138ページ
Export-Alias	139ページ
Import-Alias	141ページ

▶動的パラメータ

動的パラメータとは、「PowerShell」プロバイダーによって追加されるコマンドレットパラメータで、プロバイダーに対応したドライブでコマンドレットが使用されている場合のみ利用できます。「Alias:」ドライブでサポートされる動的パラメータは、次のとおりです。

●Options <スコープ項目オプション>

エイリアスの「Options」プロパティの値を決定します。

値	説明
None	オプションなし(既定値)
Constant	エイリアスは削除できず、プロパティは変更できない(エイリアスを作成するときのみ指定可能)
Private	エイリアスは、現在のスコープでのみ表示される(子スコープでは表示されない)
ReadOnly	「Force」パラメータを使用しない限り、エイリアスのプロパティは変更できない(「Remove-Item」を使用すると、エイリアスを削除はできる)
AllScope	エイリアスは、新たに作成されるすべてのスコープにコピーされる
Unspecified	オプションはユーザーよって指定されていない(V3～)

サポートされているコマンドレットは、「New-Item」「Set-Item」です。

「Certificate」プロバイダー

「Certificate」プロバイダー(証明書プロバイダー)は、PowerShell内からX509証明書ストアおよび証明書へのアクセスを提供します。「Certificate」プロバイダーを使用して、証明書の名前空間内を移動し、証明書ストアおよび証明書を表示することができます。また、証明書と証明書ストアのコピー、移動、および削除を行うこともできます。「Certificate」プロバイダーは、証明書の名前空間をPowerShellの「Cert:」ドライブとして公開します。「Cert:」ドライブには次の3つのレベルが存在します。

●保存場所(Microsoft.PowerShell.Commands.X509StoreLocation)

現在のユーザーおよびすべてのユーザーの証明書をグループ化するための高レベルのコンテナーです。各システムには、「CurrentUser」および「LocalMachine」の保存場所があります。

●証明書ストア(System.Security.Cryptography.X509Certificates.X509Store)

証明書の保存および管理に使用される物理ストアです。

● X509証明書(System.Security.Cryptography.X509Certificates.X509Certificate2)

それぞれがコンピュータのX509証明書を表しています。証明書は拇印によって識別されます。

▶ コマンドレット

「Certificate」プロバイダーは、「Set-Location」「Get-Location」「Get-Item」「Get-ChildItem」「Invoke-Item」」コマンドレットをサポートしています。さらに、PowerShellセキュリティスナップイン(Microsoft.PowerShell.Security)には、証明書プロバイダーに加えて、Authenticode署名の取得と設定、および証明書の取得を行うスナップインも含まれています。セキュリティスナップインに含まれているコマンドレットの一覧を参照するには、次のように入力します。

```
Get-Command -Module *security
```

▶ 動的パラメータ

「Cert:」ドライブでサポートされる動的パラメータは、次のとおりです。

● CodeSigningCert

コード署名機関を持つ証明書のみを取得します。

サポートされているコマンドレットは、「Get-Item」「Get-ChildItem」です。

「Environment」プロバイダー

「Environment」プロバイダー(環境プロバイダー)は、Windows環境変数へのアクセスを提供します。「Environment」プロバイダーを使用すると、PowerShellでWindows環境変数を取得、追加、変更、クリア、および削除できます。「Environment」プロバイダーは、「Env:」ドライブにデータストアを公開します。

▶ コマンドレット

「Environment」プロバイダーは、「Invoke-Item」を除き、名前に「Item」という名詞が含まれるすべてのコマンドレットをサポートします。また、「Get-Content」および「Set-Content」コマンドレットもサポートします。ただし、「ItemProperty」という名詞を含むコマンドレット、およびあらゆるコマンドレットの「Filter」パラメータはサポートされません。

「FileSystem」プロバイダー

「FileSystem」プロバイダーは、ファイルおよびディレクトリ(フォルダ)へのアクセスを提供します。「FileSystem」プロバイダーを使用すると、PowerShellでファイルやディレクトリを取得、追加、変更、クリア、および削除できます。

▶ コマンドレット

「FileSystem」プロバイダーは、名前に「Item」「Content」、および「ItemProperty」という名詞が含まれるすべてのコマンドレットをサポートします。また、あらゆるコマンドレットの「Filter」パラメータをサポートします。

▶動的パラメータ

「FileSystem」プロバイダーのドライブでサポートされる動的パラメータは、次のとおりです。

● Encoding <エンコード>

ファイルを読み書きするときの文字エンコーディングを指定します。「Encoding」パラメータの有効値は、705ページ参照してください。

サポートされているコマンドレットは、「Add-Content」「Get-Content」「Set-Content」です。

● Delimiter <区切り文字>

ファイルを読み取るときに使用する区切り文字を指定します

サポートされているコマンドレットは、「Get-Content」です。

● Wait

ファイルの内容が追加されるときまで待機します。

サポートされているコマンドレットは、「Get-Content」です。

「Function」プロバイダー

「Function」プロバイダー（関数プロバイダー）は、PowerShellで定義されている関数へのアクセスを提供します。「Function」プロバイダーを使用すると、PowerShellで関数やフィルターを取得、追加、変更、クリア、および削除できます。「Function」プロバイダーは、「Function:」ドライブにデータストアを公開します。

▶コマンドレット

「Function」プロバイダーは、「Invoke-Item」を除き、名前に「Item」という名詞が含まれるすべてのコマンドレットをサポートします。また、「Get-Content」および「Set-Content」コマンドレットもサポートします。ただし、「ItemProperty」という名詞を含むコマンドレットおよび、あらゆるコマンドレットの「Filter」パラメータはサポートされません。

▶動的パラメータ

「Function:」ドライブでサポートされる動的パラメータは、「Alias:」ドライブと同じです（54ページ参照）。

「Registry」プロバイダー

「Registry」プロバイダー（レジストリプロバイダー）は、PowerShellからシステムレジストリのキーと値へのアクセスを提供します。「Registry」プロバイダーを使用すると、PowerShellでレジストリのキーと値を取得、追加、変更、クリア、および削除できます。「Registry」プロバイダーは、「HKLM:」および「HKCU:」ドライブにデータストアを公開します。

▶コマンドレット

「Registry」プロバイダーは、名前に「Item」という名詞が含まれるすべてのコマンドレット（「Item」コマンドレット）をサポートします。レジストリのキーやサブキーを操作する場合は、

「Item」コマンドレットを使用します。また、名前に「ItemProperty」という名詞が含まれるコマンドレット(「ItemProperty」コマンドレット)もサポートします。レジストリの値やデータを操作する場合は、「ItemProperty」コマンドレットを使用します。レジストリプロバイダーでは、名前に「Content」という名詞が含まれるコマンドレットは使用できません。

▶動的パラメータ

「Registry」プロバイダーのドライブでサポートされる動的パラメータは次のとおりです。

●Type <レジストリ値の種類>

レジストリに値を保存するときに使用するデータ型を指定するか、レジストリの値のデータ型を識別します。有効値は「PropertyType」パラメータと同じです(712ページ参照)。

サポートされているコマンドレットは、「Set-Item」「Set-ItemProperty」です。

「Variabl」プロバイダー

「Variabl」プロバイダー(変数プロバイダー)は、PowerShellの変数とその値へのアクセスを提供します。「Variabl」プロバイダーを使用すると、現在のコンソールにあるPowerShellの変数を取得、追加、変更、クリア、および削除できます。「Variabl」プロバイダーは、「Variable:」ドライブにデータストアを公開します。

▶コマンドレット

「Variabl」プロバイダーは、「Invoke-Item」を除き、名前に「Item」という名詞が含まれるすべてのコマンドレットをサポートします。また、「Get-Content」および「Set-Content」コマンドレットもサポートします。ただし、「ItemProperty」という名詞を含むコマンドレットおよび、あらゆるコマンドレットの「Filter」パラメータはサポートされません。なお、PowerShellには、特に変数の表示と変更を行うために設計された、次のコマンドレットが用意されています。

コマンドレット	解説
Get-Variable	152ページ
New-Variable	154ページ
Set-Variable	156ページ
Clear-Variable	157ページ
Remove-Variable	158ページ

「WSMan」プロバイダー

「WSMan」プロバイダーは、Web Services for Management(WS-Management)の構成情報へのアクセスを提供します。「WSMan」プロバイダーを使用すると、ローカルまたはリモートコンピュータ上のWS-Management構成データを追加、変更、クリア、および削除できます。「WSMan」プロバイダーは、「WSMan:」ドライブにデータストアを公開します。

「WSMan」プロバイダーは、PowerShellのドライブと、WS-Managementの構成設定の論理グループに相当するディレクトリ構造を表示します。これらのグループはコンテナーと呼ばれています。

● Client

WS-Managementクライアントのさまざまな設定を構成できます。構成情報はレジストリに保存されます。

● Service

WS-Managementサービスのさまざまな設定を構成できます。構成情報はレジストリに保存されます。サービス構成は「サーバー構成」と呼ばれることもあります。

● Shell

WS-Managementシェルのさまざまな設定を構成できます。たとえば、リモートシェルアクセスを有効にしたり(AllowRemoteShellAccess)、同時にアクセス可能な最大ユーザー数を設定したり(MaxConcurrentUsers)できます。

● Listener

リスナーの作成と構成ができます。リスナーは、WS-Managementプロトコルを実装してメッセージを送受信する管理サービスです。

● Plugin

プラグインは、さまざまな機能を提供するためにWS-Managementサービスが読み込んで使用します。既定では、PowerShellはイベント転送プラグイン、PowerShellプラグイン、およびWindows Management Instrumentation(WMI)プロバイダープラグインという3つのプラグインを提供しています。これらのプラグインは、イベント転送、構成、およびWMIアクセスをサポートします。

● ClientCertificate

クライアント証明書を作成できます。クライアント証明書は、WS-Managementクライアントが証明書認証を使用するように構成されているときに使用されます。

▶ コマンドレット

PowerShellでは、次の「WSMan(WS-Management)」コマンドレットを使用できます。

コマンドレット	解説
Connect-WSMan	570ページ
Disconnect-WSMan	572ページ
Get-WSManInstance	573ページ
New-WSManInstance	575ページ
Remove-WSManInstance	578ページ
Set-WSManInstance	580ページ
Invoke-WSManAction	582ページ
New-WSManSessionOption	584ページ
Enable-WSManCredSSP	586ページ

コマンドレット	解説
Get-WSManCredSSP	588ページ
Disable-WSManCredSSP	590ページ
Set-WSManQuickConfig	591ページ
Test-WSMan	592ページ

また、リスナー、プラグイン（IntitializationParameters、Resources、Security）および「ClientCertificate」プロバイダーのパスは、特定の「New-Item」のサポートを提供します。カスタムヘルプを参照するには、適切なパスで「Get-Help New-Item」と入力します。

▶動的パラメータ

「WSMan:」ドライブでサポートされる動的パラメータは次のとおりです。

● Address <アドレス>

このリスナーが作成されたアドレスを指定します。「Address」パラメータの値はリスナーの作成時に設定されます。

値	説明
*	「*」（ワイルドカード文字）を使用すると、ネットワークインターフェイスカード（NIC）上のすべてのIPドレスが結合される
IP:<IPアドレス>	「IP:」の後にIPv4の「.」（ドット）で区切られた10進数形式またはIPv6の「:」（コロン）で区切られた16進数形式で有効なIPアドレスを入力する
MAC:	「MAC:」の後に、NICのMACアドレスを入力する

サポートされているコマンドレットは、「Get-Item」です。

● AllowRemoteShellAccess <ブール値>

リモートシェルへのアクセスを有効にします「False」に設定した場合は、新しいリモートシェル接続がサーバーによって拒否されます。既定値は「True」です。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● AllowUnEncrypted <ブール値>

クライアントコンピュータが暗号化されていないトラフィックを要求することを許可します。既定では、クライアントコンピュータは暗号化されたネットワークトラフィックを要求します。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Arguments <引数>

カスタムシェルに渡す引数文字列とコマンドライン引数を指定します。このパラメータは省略可能です。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Basic <ブール値>

クライアントコンピュータが基本（ベーシック）認証を使用することを許可します。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Capability <操作>

このURIでサポートされる操作を指定します。URIがサポートしている各操作につき1つエントリを作成する必要があります。

値	説明
Create	Create操作がサポートされる。Create操作でこの概念がサポートされている場合は、「SupportFragment」属性が使用される。「SupportFiltering」属性は無効なので、「False」に設定する必要がある。Shell操作もサポートされている場合、この操作はURIに対して無効
Delete	Delete操作がサポートされる。Delete操作でこの概念がサポートされている場合は、「SupportFragment」属性が使用される。「SupportFiltering」属性は無効なので、「False」に設定する必要がある。Shell操作もサポートされている場合、この操作はURIに対して無効
Enumerate	Enumerate操作がサポートされる。「SupportFragment」属性は「Enumerate」操作ではサポートされていないため、「False」に設定する必要がある。「SupportFiltering」属性は有効。プラグインでフィルター処理がサポートされている場合は、この属性を「True」に設定する必要がある。Shell操作もサポートされている場合、この操作はURIに対して無効
Get	Get操作がサポートされる。Get操作でこの概念がサポートされている場合は、「SupportFragment」属性が使用される。「SupportFiltering」属性は無効なので、「False」に設定する必要がある。Shell操作もサポートされている場合、この操作はURIに対して無効
Invoke	Invoke操作がサポートされる。「SupportFragment」属性は「Invoke」操作ではサポートされていないため、「False」に設定する必要がある。「SupportFiltering」属性は無効なので「False」に設定する必要がある。Shell操作もサポートされている場合、この操作はURIに対して無効
Put	Put操作がサポートされる。Put操作でこの概念がサポートされている場合は、「SupportFragment」属性が使用される。「SupportFiltering」属性は無効なので、「False」に設定する必要がある。Shell操作もサポートされている場合、この操作はURIに対して無効
Subscribe	Subscribe操作がサポートされる。「SupportFragment」属性は「Subscribe」操作ではサポートされていないため、「False」に設定する必要がある。「SupportFiltering」属性は無効なので、「False」に設定する必要がある。Shell操作もサポートされている場合、この操作はURIに対して無効
Shell	Shell操作がサポートされる。「SupportFragment」属性はShell操作ではサポートされていないため、「False」に設定する必要がある。「SupportFiltering」属性は無効なので、「False」に設定する必要がある。他の操作もサポートされている場合、この操作はURIに対して無効。URIに対してShell 操作が構成されている場合、「Get」「Put」「Create」「Delete」「Invoke」および「Enumerate」操作は、シェルを管理するため、WS-Management（WinRM）サービス内部で処理される。その結果、プラグインで操作を処理できない

サポートされているコマンドレットは、「Get-Item」「New-Item」「Remove-Item」「Set-Item」です。

● CbtHardeningLevel <ポリシー>

認証要求内のチャネルバインドトークンの必要条件ポリシーを設定します。このパラメータの値はHTTPS接続に対してのみ有効です。

値	説明
Strict	チャネルバインドトークンがない要求は、すべて拒否される。この設定では、チャネルバインドトークンの使用によってすべての接続がセキュリティで保護される
Relaxed	要求にチャネルバインドトークンがある場合に、接続がセキュリティで保護される。チャネルバインドトークンがない場合でも、接続は受け入れられる。ただし、チャネルバインドトークンの使用によって防げる攻撃に対して脆弱になる
None	指定したチャネルバインドトークンはすべて無視される

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● CertificateThumbprint <証明書捺印>

サーバー証明書の捺印を指定します。この値は、証明書の「Thumbprint」フィールドに含まれている2桁の16進数値の文字列を表します。この処理を実行する権限のあるユーザーアカウントのデジタル公開キー証明書(X509)を指定します。証明書は、クライアント証明書ベースの認証で使用されます。これらはローカルユーザーアカウントにのみマッピングでき、ドメインアカウントでは機能しません。証明書の捺印を取得するには、「Cert:」ドライブで「Get-Item」または「Get-ChildItem」コマンドレットを使用します。

サポートされているコマンドレットは、「Clear-Item」「Get-Item」「Set-Item」です。

● Certificate <ブール値>

クライアントを証明書認証に使用できます。WS-Managementクライアントはコンピュータストアで証明書を見つけようとします。コンピュータストアで証明書を見つけることができない場合、クライアントは現在のユーザーストアで見つけようとします。一致する証明書が見つからない場合は、エラーメッセージが表示されます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● CredSSP <ブール値>

クライアントがCredSSP認証を使用することを許可します。CredSSP認証では、ユーザーが資格情報を委任できます。このオプションは、1台のリモートコンピュータで実行していても、データを他のリモートコンピュータから収集したり、追加のコマンドを他のリモートコンピュータで実行したりするコマンド用に設計されています。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● HTTP <ポート番号>

HTTPが使用されるときにクライアントが使用するポートを指定します。既定では、HTTPはポート80を使用します。1～65535の任意の値を指定できます

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● HTTPS <ポート番号>

HTTPSが使用されるときにクライアントが使用するポートを指定します。既定では、HTTPS

はポート443を使用します。1～65535の任意の値を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Digest <ブール値>

クライアントがダイジェスト認証を使用することを許可します。ダイジェスト認証はHTTPとHTTPSでサポートされています。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Enabled <ブール値>

リスナーが有効か無効かを指定します。既定値は「True」です。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● ExactMatch <ブール値>

「Sddl」パラメータで指定されているセキュリティ設定の使用方法を指定します。「True」に設定した場合は、Sddlのセキュリティ設定を使用して、URIで指定されているURIへのアクセス試行のみが承認されます。「False」に設定した場合は、Sddlのセキュリティ設定を使用して、URIで指定されている文字列で始まるURIへのアクセス試行が承認されます

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● FileName <ファイル名>

「ResourceURI」と「SelectorSet」パラメータで指定された管理リソースの更新に使用する入力ファイルを指定します。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● FileName (Plugin) <ファイル名>

操作プラグインのファイル名を指定します。要求を受け取ると、このエントリに入力した環境変数がユーザーのコンテキストで展開されます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● HostName <ホスト名>

WS-Management(WinRM)サービスが実行されているコンピュータのホスト名を指定します。値は完全修飾ドメイン名、IPv4またはIPv6リテラル文字列、またはワイルドカード文字でなければなりません。

サポートされているコマンドレットは、「Clear-Item」「Get-Item」「Set-Item」です。

● IdleTimeOut <アイドル待ち時間>

リモートシェルでユーザー操作がない場合に、リモートシェルが開いたままになる最大時間をミリ秒で指定します。指定した時間が過ぎると、リモートシェルは自動的に削除されます。0～

2147483647の任意の値を指定できます。「0」はタイムアウトが無限であることを示します。既定は「900000」(15分)です。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● IPv4Filter <IPv4アドレス>

リスナーが使用できるIPv4アドレスを指定します。このパラメータを空白にすると、IPv4アドレスは使用できません。「*」(ワイルドカード文字)を入力した場合は、任意のIPv4アドレスを使用できます。IP範囲のリストを入力した場合は、指定した範囲のIPアドレスを使用できます。複数の範囲は「,」(コンマ)で区切り、それぞれの範囲をドット区切りの10進数形式のIPv4アドレスのペアとして「-」(ハイフン)で区切って指定します。

サポートされているコマンドレットは、「Clear-Item」「Get-Item」「Set-Item」です。

● IPv6Filter <IPv6アドレス>

リスナーが使用できるIPv6アドレスを指定します。このパラメータを空白にすると、IPv6アドレスは使用できません。「*」(ワイルドカード文字)を入力した場合は、任意のIPv6アドレスを使用できます。IP範囲のリストを入力した場合は、指定した範囲のIPアドレスを使用できます。複数の範囲は「,」(コンマ)で区切り、それぞれの範囲をドット区切りの16進数形式のIPv6アドレスのペアとして「-」(ハイフン)で区切って指定します。

サポートされているコマンドレットは、「Clear-Item」「Get-Item」「Set-Item」です。

● Issuer <証明機関名>

証明書を発行した証明機関の名前を指定します。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Kerberos <Boolean>

クライアントがKerberos認証を使用することを許可します。「Kerberos」認証は、クライアントとサーバーがKerberos証明書を使用して相互に認証するスキームです。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● ListeningOn <IPアドレス>

サービスが実際にリッスンするIPアドレスまたはすべてのIPアドレスを指定します。IPアドレスの値は、IPv4のドット区切りの10進数表記かIPv6のコロン区切りの16進数表記で指定します。

サポートされているコマンドレットは、「Get-Item」です。

● MaxBatchItems <バッチアイテム最大数>

Pull応答で使用できる要素の最大数を指定します。1～4294967295の任意の値を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● MaxConcurrentUsers <最大同時接続ユーザー数>

リモートシェルから同じコンピュータに同時にリモート操作を実行できるユーザーの最大数を指定します。指定した制限を超えた場合は、新しいシェル接続が拒否されます。1～100の任意の値を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● MaxEnvelopeSizekb <SOAPデータ最大サイズ>

最大SOAPデータをキロバイト単位で指定します。32～4294967295の任意の値を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● MaxMemoryPerShellMB <メモリ最大容量>

アクティブなリモートシェルとそのすべての子プロセスごとに割り当てできるメモリの最大合計容量を指定します。0～2147483647の任意の値を指定できます。「0」は、メモリを割り当てるリモート操作の機能が、使用可能な仮想メモリによってのみ制限されることを示します。既定値は「0」です。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● MaxProcessesPerShell <プロセス最大数>

任意のシェル操作を開始できるプロセスの最大数を指定します。0～2147483647の任意の数値を指定できます。「0」はプロセス数に制限がないことを示します。既定では、シェルごとにプロセスは5つに制限されます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● MaxShellsPerUser <シェルの最大数>

同じシステムでユーザーがリモートから同時に開けるシェルの最大数を指定します。このポリシー設定を有効にすると、指定した限度を超えた場合に、ユーザーは新しいリモートシェルを開けなくなります。このポリシー設定を無効にした場合や構成していない場合、既定では、ユーザーごとのリモートシェル数の制限は「2」に設定されます。0～2147483647の任意の数値を指定できます。「0」はシェル数に制限がないことを示します。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● MaxTimeoutMs <最大タイムアウト時間>

Pull要求以外のすべての要求に使用できる最大タイムアウトをミリ秒単位で指定します。500～4294967295の任意の数値を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Name <セッションの表示名>

WS-Managementセッションの表示名を指定します。「Get-PSSession」や「Enter-PSSession」などの他のコマンドレットを使用する場合は、この名前を使用してセッションを参照できます。この名前は、コンピュータまたは現在のセッションで一意でなくてもかまいません。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Name (Plugin) <プラグインの表示名>

プラグインに使用する表示名を指定します。プラグインからエラーが返された場合、表示名はエラーXMLに挿入され、クライアントアプリケーションに返されます。この名前はロケールに固有ではありません。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Negotiate <ブール値>

クライアントがネゴシエート認証を使用することを許可します。ネゴシエート認証は、クライアントがサーバーに認証要求を送るスキームです。サーバーは、Kerberosプロトコルを使用するかNTLMを使用するかを決定します。Kerberosプロトコルはドメインのアカウント、NTLMはローカルコンピュータのアカウントを認証する場合に選択されます。ドメインユーザーの場合、ユーザー名は「domain¥user_name」の形式で指定する必要があります。サーバーコンピュータのローカルユーザーの場合、ユーザー名は「server_name¥user_name」の形式で指定する必要があります。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● NetworkDelayMs <遅延待機時間>

ネットワーク遅延時間に対応するためにクライアントコンピュータが待機する余分な時間をミリ秒単位で指定します。500～4294967295の任意の値を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Password <パスワード文字列>

ローカルアカウントまたはドメインアカウントのパスワードを指定します。NULLにすることはできません。クライアントコンピュータは、コンピュータでシェルを作成するときに使用する資格情報を指定できます。ドメインユーザーの場合、ユーザー名は「domain¥user_name」の形式で指定する必要があります。サーバーコンピュータのローカルユーザーの場合、ユーザー名は「server_name¥user_name」の形式で指定する必要があります。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Plugin <プラグイン>

WS-Managementの機能を拡張するプラグインを指定します。

サポートされているコマンドレットは、「New-Item」「Remove-Item」です。

● Port <ポート番号>

このリスナーを作成するTCPポートを指定します。1～65535の任意の値を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Resource <リソース>

管理操作の明確な種類または値を表すエンドポイントを指定します。サービスは1つまたは複数のリソースを公開し、リソースによっては複数のインスタンスを所有できます。管理リソースはWMIクラスまたはデータベーステーブルに似ており、インスタンスはクラスのインスタンスまたはテーブル内の行に似ています。たとえば、「Win32_LogicalDisk」クラスはリソースを表します。「Win32_LogicalDisk="C:¥"」はリソースの特定のインスタンスです。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● ResourceURI <リソースURI>

ディスク、プロセスなど、コンピュータのリソースの種類を識別するURIを指定します。

サポートされているコマンドレットは、「Get-Item」です。

● RootSDDL <ルートSDDL>

アクセス制御エントリのセキュリティ記述子定義言語(SDDL)を指定します。これは、指定したリソースURIへのアクセス承認に使用されるセキュリティ設定を識別します。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● SDKVersion <バージョン>

WS-ManagementプラグインSDKのバージョンを指定します。有効な値は「1」のみです。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Shell <プロセス文字列>

カスタムシェルのプロセス文字列を指定します。環境変数を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● ShellTimeout <タイムアウト時間>

非アクティブであったためシェルがタイムアウトになるまでの時間を指定します。タイムアウト値をミリ秒単位で指定します。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Subject <エンティティ>

証明書によって識別されるエンティティを指定します。

サポートされているコマンドレットは、「Get-Item」です。

● SupportsOptions <ブール値>

プラグインがオプションの使用をサポートしているかどうかを指定します。これらは要求メッセージの「wsman:OptionSet」ヘッダー内で渡されます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Transport <トランスポート>

「WS-Management」プロトコルの要求や応答の送受信に使用するトランスポートを指定します。この値は、「HTTP」または「HTTPS」でなければなりません。「Tansport」の値はリスナーの作成時に設定されます。

サポートされているコマンドレットは、「Get-Item」です。

● TrustedHosts <ホスト名のリスト>

信頼されているネットワーク接続でローカルコンピュータに接続しているリモート)コンピュータ(ホスト)の一覧を指定します。HTTPを介した基本認証など、クライアントがサービスに認証できない認証スキームとトランスポートを使用している場合に、この一覧に指定されているコンピュータに要求を送信できます。ホスト名にはドメインネームシステム(DNS)名かIPアドレスを指定できます。空白を指定した場合は、どのホストも信頼しません。「*」(アスタリスク)を指定した場合は、すべてのホストを信頼します。

サポートされているコマンドレットは、「Clear-Item」「Get-Item」「Set-Item」です。

● URLPrefix <URLプレフィックス>

HTTPまたはHTTPS要求を受け入れるURLのプレフィックスを指定します。これは、「a-z」「A-Z」「9-0」「_」(アンダースコア)、および「/」(スラッシュ)の文字のみを含む文字列です。文字列の先頭と末尾に「/」を使うことはできません。たとえば、コンピュータ名が「SamplePc」の場合、WS-Managementクライアントは送信先アドレスに「http://SamplePc/URLPrefix」と指定します。

サポートされているコマンドレットは、「Clear-Item」「Get-Item」「Set-Item」です。

● UserName <String>

ローカルアカウントまたはドメインアカウントのユーザー名を指定します。NULLの値は使用できません。クライアントは、コンピュータでシェルを作成するときに使用する資格情報を指定できます。ドメインアカウントの場合は、ユーザー名は「domain¥user_name」の形式で、サーバーコンピュータのローカルアカウントの場合、ユーザー名は「server_name¥user_name」の形式で指定する必要があります。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● Value <初期化パラメータ>

初期化パラメータを指定します。これは、構成オプションの指定に使用されるプラグインに固有の値です。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● XMLRenderingType <XMLレンダリング形式>

「WSMAN_DATA」オブジェクトを使用してXMLがプラグインに渡される形式を指定します。

値	説明
Text	入力XMLデータは「WSMAN_DATA_TYPE_TEXT」構造体に格納される。この構造体は、XMLをPCWSTRのメモリバッファーとして表す
XMLReader	入力XMLデータは「WSMAN_DATA_TYPE_WS_XML_READER」構造体に格納される。この構造体は、XMLを「WebServices.h」ヘッダーファイルで定義される「XmlReader」オブジェクトとして表す

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● xmlns <XML名前空間>

XML名前空間を指定します。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● lang <言語>

言語(「ja」「en」など)、または言語と地域(「ja-JP」「en-US」など)を表す文字列を指定します。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● HTTP <ポート番号>

HTTPが使用されるときにクライアントが使用するポートを指定します。既定では、HTTPはポート80を使用します。1～65535の任意の値を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

● HTTPS <ポート番号>

HTTPS使用されるときにクライアントが使用するポートを指定します。既定では、HTTPSはポート443を使用します。1～65535の任意の値を指定できます。

サポートされているコマンドレットは、「Get-Item」「Set-Item」です。

リモート機能について

PowerShellのリモート機能

PowerShellでは、一時接続または固定接続を使用して、ネットワークに接続されている別のコンピュータ上でコマンド(リモートコマンド)を実行することができます。リモート処理には、WS-Management(WinRM)が使用されます。

WS-Managementは、さまざまなベンダーのハードウェアとオペレーティングシステムの相互運用を可能にする、SOAPをベースとするファイアウォール互換の標準通信プロトコルです。MicrosoftがWS-Managementは、WinRM(Windows Remote Management)と呼ばれます。

WinRMのリモート機能を使用してリモートコマンドを実行するには、ローカルコンピュータ(クライアントコンピュータ)およびリモートコンピュータ(ホストコンピュータ)をリモート処理用に構成する必要があります(514ページ参照)。

リモート処理用に構成されていない場合、次のコマンドレットを実行することができません。

コマンドレット	解説
New-PSWorkflowSession	506ページ
Invoke-AsWorkflow	510ページ
New-PSSession	539ページ
Get-PSsession	541ページ
Disconnect-WSMan	544ページ
Connect-PSSession	546ページ
Enter-PSSession	549ページ
Receive-PSSession	555ページ
Remove-PSSession	558ページ
Invoke-Command	565ページ
Connect-WSMan	570ページ
Get-WSManInstance	573ページ
New-WSManInstance	575ページ
Set-WSManInstance	580ページ
Test-WSMan	592ページ

「ComuterName」パラメータを持つコマンドレットでのデータの取得

一部のコマンドレットには、リモートコンピュータからオブジェクトを取得するための「ComputerName」パラメータがあります。これらのコマンドレットは、コンピュータをリモート処理用に構成することなく使用できます。ただし、無条件で使用できるのは、「Get-Service」と「Set-Service」コマンドレットの2つです。その他のコマンドレットは、ファイアウォールの受信規則や依存するサービスの設定を行う必要があります。

▶ファイアウォール

ファイアウォール(Windowsファイアウォール)によってブロックされるコマンドレットは、次のとおりです。

コマンドレット	受信規則	解説
Get-WinEvent	リモートイベントログ監視(RPC)	389ページ
Get-HotFix	Windows Management Instrumentation (WMI受信)	402ページ
Restart-Computer	Windows Management Instrumentation (WMI受信)	404ページ
Stop-Computer	Windows Management Instrumentation (WMI受信)	405ページ
Rename-Computer	Windows Management Instrumentation (WMI受信)	459ページ
Get-WmiObject	Windows Management Instrumentation (WMI受信)	593ページ
Remove-WmiObject	Windows Management Instrumentation (WMI受信)	598ページ
Invoke-WmiMethod	Windows Management Instrumentation (WMI受信)	600ページ
Register-WmiEvent	Windows Management Instrumentation (WMI受信)	616ページ

これらのコマンドレットをリモートコンピュータに対して実行できるようにするには、リモートコンピュータ側のファイアウォールの設定で、表中の「受信規則」の項目を有効にします。

▶リモートレジストリサービス

リモートレジストリサービス(Remote Registry)に依存するコマンドレットは、次のとおりです。

コマンドレット	解説
New-EventLog	376ページ
Write-EventLog	378ページ
Get-EventLog	380ページ
Show-EventLog	382ページ
Limit-EventLog	384ページ
Clear-EventLog	385ページ
Remove-EventLog	387ページ
Get-Process	410ページ
Get-Counter	429ページ

これらのコマンドレットをリモートコンピュータに対して実行できるようにするには、リモートコンピュータ側でリモートレジストリサービスを起動します。リモートレジストリサービスを起動するには、次のように操作します。

❶ 次のように入力して、リモートレジストリサービスのスタートアップモードを取得します。

```
PS C:¥work> Get-WmiObject win32_service -filter "name='RemoteRegistry'"↵



ExitCode  : 1077
Name      : RemoteRegistry
ProcessId : 0
StartMode : Disabled ←スタートアップモード
State     : Stopped
Status    : OK
```

❷ スタートアップモードが「Manual」なら操作❸に進みます。スタートアップモードが「Disabled」なら次のように入力して、「Manual」に設定します。

```
PS C:¥work> Set-Service -Name RemoteRegistry -StartupType Manual
```

❸ 次のように入力して、リモートレジストリサービスを起動します。

```
PS C:¥work> Start-Service -Name RemoteRegistry
```

自動変数とユーザー設定変数について

PowerShellの自動変数

自動変数とは、PowerShellの状態情報を格納する変数です。自動変数は、PowerShellによって作成され、保持されます。自動変数の一覧を次の表に示します。

変数	説明
$$	セッションが受け取った最後の行にある最後のトークンを格納
$?	最後の演算が成功した場合は「True」、失敗した場合は「False」を格納
$^	セッションが受け取った最後の行にある最初のトークンを格納
$_	パイプラインオブジェクトの現在のオブジェクトを格納
$Args	宣言されていないパラメータの配列や、関数、スクリプト、またはスクリプトブロックに渡されるパラメータ値を格納
$ConsoleFileName	セッションで最後に使用されたコンソールファイル(.psc1)のパスを格納
$Error	最新のエラーを表すエラーオブジェクトの配列を格納
$Event	処理中のイベントを表す「PSEventArgs」オブジェクトを格納
$EventSubscriber	処理中のイベントのサブスクライバーを表す「PSEventSubscriber」オブジェクトを格納
$ExecutionContext	PowerShellホストの実行コンテキストを表す「EngineIntrinsicsオブジェクトを格納
$False	「False」を格納
$ForEach	「ForEach-Object」ループの列挙子を格納
$HOME	ユーザーのホームディレクトリの完全なパスを格納
$Host	PowerShellの現在のホストアプリケーションを表すオブジェクトを格納
$Input	関数に渡される入力を格納する列挙子
$LastExitCode	最後に実行されたWindowsベースのプログラムの終了コードを格納
$Matches	「-match」および「-not matc」による演算の結果、一致した任意の文字列値のハッシュテーブルが格納される
$MyInvocation	スクリプト、関数、スクリプトブロックなど、現在のコマンドに関する情報を含むオブジェクトを格納
$NestedPromptLevel	現在のプロンプトレベルを格納
$NULL	「Null」または空の値を格納
$PID	現在のPowerShellセッションをホストしているプロセスのプロセス識別子(PID)を格納
$Profile	現在のユーザーおよび現在のホストアプリケーションのPowerShellプロファイルの完全なパスを格納
$PSBoundParameters	有効なパラメータとそのパラメータ現在の値のディクショナリを格納
$PSCommandPath (V3～)	現在実行中のスプリクトのパスを格納
$PsCulture	オペレーティングシステムで現在使用されているカルチャの名前を格納
$PSDebugContext	この変数は、デバッグ中にデバッグ環境に関する情報を格納
$PSHOME	PowerShellのインストールディレクトリの完全なパスを格納
$PSItem(V3～)	変数「$_」と同じ
$PSScriptRoot	スクリプトモジュールが実行されているディレクトリを格納
$PSUICulture	オペレーティングシステムで現在使用されているユーザーインターフェイス(UI)カルチャの名前を格納