Windows と Windows Server の互換性ガイドブック

Windows 8, Windows 8.1, Windows Server 2012 と Windows Server 2012 R2

2013 年 10 月 14 日

要約

このガイドブックでは、Windows® 8/Windows 8.1/Windows Server® 2012 オペレーティング システムの変更点と新機能に関する情報を提供します。また、開発者が既存および開発予定のアプリと新しいオペレーティング システムとの互換性を確認するためのガイドラインも含まれています。読者が以前のバージョンの Windows を使い慣れていることを前提としています。

この内容は、以下に適用されます。

- Windows 8.1
- Windows 8
- Windows Server 2012 R2
- Windows Server 2012
- Windows 7
- Windows Server 2008 R2

この**ガイドブック**は Microsoft Windows 環境で使うように設計されているアプリのサード パーティ開発者向けであり、
http://msdn.microsoft.com/library/hh848074(v=vs.85).aspx で参照したり、
https://technet.microsoft.com/en-us/windows/application-compatibility.aspx でダウンロードしたりすることができます。

Microsoft

内容

2013 年 10 月 14 日

概要

Windows 8、Windows 8.1 と Windows Server 2012 R2 には、世界中のアプリ/ドライバー開発者や企業が使用できる、最新のオペレーティング システム (OS) テクノロジとソフトウェア開発プラットフォームが導入されています。Windows のセキュリティ、信頼性、パフォーマンス、ユーザー エクスペリエンスをさらに向上させる一環として、Microsoft はさまざまな新機能の追加、既存の機能の強化と機能の廃止を行っています。

Windows 8、Windows 8.1、および Windows Server 2012 R2 の目標は、過去にリリースされた OS 向けに作成された多くのアプリとの高い互換性を維持することにあります。しかし、革新、セキュリティの強化、信頼性の向上などから生じる、ある程度の互換性の喪失は避けて通れません。全体としては、新しい OS と既存のアプリ/デバイスとの互換性は、高く保たれています。

このドキュメントでは、アプリやデバイスに新しい OS との互換性があることを確認する方法を説明し、アプリの互換性に関するいくつかの既知の問題の概要を紹介します。

このガイドブックは、2 つの大きなセクションに分かれています。

- Windows 8.1 と Windows Server 2012 R2
- Windows 8 と Windows Server 2012

ガイドブックの今回のリリースで新しく追加されたトピックと更新されたトピックは以下のとおりです。

- Windows 8.1 のオペレーティング システム バージョンの変更点
- オンデマンドでインストールする Windows コンポーネント
- HTML のフォント フォールバック動作
- Windows 8.1 は HTTP URI を不許可で、HTTPS URI のみを許可
- Windows 8.1 のマウスホイール入力
- Windows 高精度タッチパッド デバイス
- Windows 8.1 におけるデスクトップ アプリケーションの高 DPI
- Web アプリケーション プロキシ

- ″内外内″ シナリオや ″慣性″ シナリオで画面のエッジ UI を非表示にする
- Windows ストア アプリからの ImmSetConversionStatus() または ImmGetConversionStatus() の呼び出しはサポートされない
- ユーザーごとからスレッドごとに変化した IME 入力モード モデル
- 入力方式マネージャーの API は簡体字中国語 IME ではサポートされない
- 一部の東アジア言語の IME でソフトウェア入力パネルから vKey が送信される
- 国際的なテキストとフォントのガイダンス
- Windows 8.1 向け Windows アプリ認定キット
- Windows 8.0 向け Windows アプリ認定キット

アプリを最適化して最新バージョンの Windows の機能を活用する方法については、以下のトピックをご覧ください。

Windows OS および Windows Server OS の以前のバージョンを取り上げた *Windows* 互換性ガイドブックは http://msdn.microsoft.com/en-us/library/bb757005.aspx（英語）と http://msdn.microsoft.com/library/dd371778(v=VS.85).aspx をご覧ください。

Windows 8.1 と Windows Server 2012 R2

ガイドブックのこのセクションは、Windows 8.1 固有のトピックに焦点を当てます。一部のトピックは Windows Server 2012 R2 にも適用されます。トピックは次の 4 つに分類されます。

- Windows ストア アプリ
- クライアントとサーバーの互換性
- 新しい機能と機能強化
- ツールとベスト プラクティス

Windows ストア アプリ

以下に示すリソースは、Windows 8 向けに作成されたアプリの Windows 8.1 への移行を支援することを目的としています。

- Windows ストア アプリの Windows 8.1 への移行 (ホワイトペーパー) - Windows 8 から Windows 8.1 へのアプリの移行に関するガイダンスとヒントを掲載したホワイト ペーパーです。ベスト プラクティス、非推奨事項、およびアプリを Windows ストアに再提出するためのガイダンスも含め、最初から最後まで説明されています。
- Windows 8.1: 新しい API と機能 (開発者向け) - 新しい API と新機能を確認し、Windows 8.1 を理解します。
- Windows ストア アプリのターゲットを Windows 8.1 に変更する - Windows 8 から Windows 8.1 にアプリを移行する方法に関するステップ バイ ステップのガイドです。
- Windows ストア アプリのサンプル - 新しい API と機能の使用方法と、廃止予定機能の対策方法を示す Windows 8.1 用のサンプルです。
- DirectX プログラミング - ゲームやグラフィックス アプリ用の新機能と機能強化について説明します。
- JavaScript および Internet Explorer を使用したプログラミングに関連する Windows 8.1 の API の変更点の詳細については、「Windows 8.1 での API の変更点 (HTML)」を参照してください。
- C#/VB/C++ および XAML を使用したプログラミングに関連する Windows 8.1 の API の変更点の詳細については、「Windows 8.1 での API の変更点 (XAML)」を参照してください。

参考資料

Windows ストア アプリの Windows 8.1 への移行 (ホワイトペーパー)
http://go.microsoft.com/fwlink/p/?LinkID=304117

Windows 8.1: 新しい API と機能 (開発者向け)
http://go.microsoft.com/fwlink/p/?LinkID=298951

Windows ストア アプリのターゲットを Windows 8.1 に変更する
https://msdn.microsoft.com/ja-jp/library/dn263114.aspx

Windows ストア アプリのサンプル
http://go.microsoft.com/fwlink/p/?LinkId=309384

DirectX プログラミング
http://go.microsoft.com/fwlink/p/?LinkID=303849

Windows 8.1 での API の変更点 (HTML)
https://msdn.microsoft.com/ja-jp/library/dn263112.aspx

Windows 8.1 での API の変更点 (XAML)
https://msdn.microsoft.com/ja-jp/library/dn263110.aspx

Windows 8.1 および Windows Server 2012 R2 のクライアントとサーバーの互換性

このセクションでは、既存のアプリに影響するおそれがあるために特別な注意を払う必要のあるオペレーティング システムの変更点と、クライアント向け、サーバー向け、またはクライアントとサーバー両方向けの新しいアプリを設計する方法を説明します。このセクションで説明するトピックは以下のとおりです。

- Windows 8.1 のオペレーティング システム バージョンの変更点
- オンデマンドでインストールする Windows コンポーネント
- HTML のフォント フォールバック動作
- Windows 8.1 は HTTP URI を不許可で、HTTPS URI のみを許可
- Windows 8.1 のマウスホイール入力
- Windows 高精度タッチパッド デバイス
- Windows 8.1 におけるデスクトップ アプリケーションの高 DPI
- ″内外内″ シナリオや ″慣性″ シナリオで画面のエッジ UI を非表示にする
- Windows ストア アプリからの ImmSetConversionStatus() または ImmGetConversionStatus() の呼び出しはサポートされない
- 入力方式マネージャーの API は簡体字中国語 IME ではサポートされない
- 一部の東アジア言語の IME でソフトウェア入力パネルから vKey が送信される

Windows 8.1 および Windows Server 2012 R2 のオペレーティング システム バージョンの変更点

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012 R2

説明

以前に GetVersion(Ex) API の使用方法によって望ましくない振る舞いが生じたため、Windows 8.1 における GetVersion(Ex) API の動作にはいくつかの大きな変更が加えられています。

以前の Windows では、GetVersion(Ex) API を呼び出すと、アプリの互換性 shim によって別のバージョンを返すように設定されていない限り、オペレーティング システム (OS) の実際のバージョンが返されました。これは、プロビジョニング ベースで行われており、リリース作業において合理的に shim を適用できるプロセスの数の点において相対的に不完全なものでした。多くのアプリは、バージョン チェックの設計が不完全なために shim が適用されていないことから、互換性 shim 設定が無視されていました。

バージョン チェックを行う第一の理由は、そのアプリのサポート OS を示すメッセージを表示するためです。しかし、チェックが適切でないため、メッセージの内容は多くの場合、「アプリを XP 以降 (当然ながら最新の OS を含む) で実行する必要があります」というものになっています。ほんどの場合において、最新の OS では、バージョン チェックをしなくてもアプリは問題なく実行されます。

影響

Windows 8.1 では、GetVersion(Ex) API は推奨されていません。この API を呼び出すことはできますが、アプリが明確に Windows 8.1 をターゲットにしていない場合は、Windows 8 のバージョン (6.2.0.0) が返されます。

解決策

Windows 8.1 をターゲットにするには、アプリ マニフェストを含めるか、ソース ファイルに _NT_TARGET_VERSION=$ (_NT_TARGET_VERSION_LATEST) を含める必要があります。

アプリ マニフェストは次のように構成します。

```
<exe>.manifest
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-
com:asm.v1" xmlns:asmv3="urn:schemas-microsoft-com:asm.v3">
    <assemblyIdentity
        type="win32"
```

```
        name=SXS_ASSEMBLY_NAME
        version=SXS_ASSEMBLY_VERSION
        processorArchitecture=SXS_PROCESSOR_ARCHITECTURE
    />
    <description> my foo exe </description>
    <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
        <security>
            <requestedPrivileges>
                <requestedExecutionLevel
                    level="asInvoker"
                    uiAccess="false"
                />
            </requestedPrivileges>
        </security>
    </trustInfo>
    <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
        <application>
            <!-- Windows 8.1 -->
            <supportedOS Id="{1f676c76-80e1-4239-95bb-83d0f6d0da78}"/>
            <!-- Windows Vista -->
            <supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"/>
            <!-- Windows 7 -->
            <supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/>
            <!-- Windows 8 -->
            <supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>
        </application>
    </compatibility>
</assembly>
```

次に、以下をソースに追加します。

SXS_MANIFEST_RESOURCE_ID=1

SXS_MANIFEST=foo.manifest

SXS_ASSEMBLY_NAME=Microsoft.Windows.Foo

SXS_ASSEMBLY_VERSION=1.0

SXS_ASSEMBLY_LANGUAGE_INDEPENDENT=1

SXS_MANIFEST_IN_RESOURCES=1

Windows 8.1 の場合、上記の**太字**で示した 2 行が、アプリで Windows 8.1 バージョンの OS を正確にターゲットにする方法を示しています。Windows 8.1 用の .exe の記述は、以前の OS を実行している場合にはまったく影響しません。.rc ファイルを既に定義している場合には、これを .rc ファイルにも追加できます。

trustInfo の追加は必須ではありませんが、追加することを強くお勧めします。追加すると、OS が Windows 8.1 と Windows 8 のどちらの場合でも、.exe が常に適切なバージョンを取得できるようになります。

代替の API は VersionHelpers 関数と呼ばれています。使用するのは非常に簡単で、#include <VersionHelpers.h> を追加するだけです。

例:

VersionHelpers.h ヘッダー ファイルで利用可能なインライン関数を使用すると、Windows のバージョンをテストするときにブール値が返され、OS のバージョンを確認

できます。たとえば、アプリの要件が Windows 8 以降の場合、次のテストを使用します。

C++

```
#include <VersionHelpers.h>

    if (!IsWindows8OrGreater())
    {
       MessageBox(NULL, "You need at least Windows 8", "Version Not
Supported",    MB_OK);
    }
```

利用可能な API は次のとおりです。

```
#define VERSIONHELPERAPI FORCEINLINE BOOL
VERSIONHELPERAPI IsWindowsXPOrGreater()
VERSIONHELPERAPI IsWindowsXPSP1OrGreater()
VERSIONHELPERAPI IsWindowsXPSP2OrGreater()
VERSIONHELPERAPI IsWindowsXPSP3OrGreater()
VERSIONHELPERAPI IsWindowsVistaOrGreater()
VERSIONHELPERAPI IsWindowsVistaSP1OrGreater()
VERSIONHELPERAPI IsWindowsVistaSP2OrGreater()
VERSIONHELPERAPI IsWindows7OrGreater()
VERSIONHELPERAPI IsWindows7SP1OrGreater()
VERSIONHELPERAPI IsWindows8OrGreater()
VERSIONHELPERAPI IsWindows8_1OrGreater()
VERSIONHELPERAPI IsWindowsServer()
```

これらは、質問の内容に応じて TRUE または FALSE を返します。必要なのは、サポートする最低レベルの OS を定義することだけです。

参考資料

Application Compatibility Toolkit のダウンロード (英語)
https://www.microsoft.com/download/details.aspx?id=39982

既知の互換性修正プログラム、互換モード、および AppHelp メッセージ
https://technet.microsoft.com/ja-jp/library/cc765984.aspx

Version Helpers API (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325426

オンデマンドでインストールする Windows コンポーネント

プラットフォーム

クライアント – Windows 8.1

説明

DirectPlay と NTVDM の 2 つの Windows コンポーネントはオンデマンドで Windows 8.1 にインストールされます。この 2 つのコンポーネントは、[レガシ コンポーネント] ノードの下のオプションのコンポーネント内にリストされます。これらのコンポーネントは、OS の一部としてローカルにインストールされ、オプションのコンポーネントとして圧縮されています。アプリでこれらのコンポーネントのうちの 1 つを (通常はアプリの初めての起動時に) 参照または呼び出す場合、自動的にインストールがトリガーされます。ユーザーにはコンポーネントがインストールされることが通知されます。インストールを進めるには、ユーザーが操作を確認する必要があります。ユーザーが管理者でない場合、インストールを成功するには管理者への昇格が必要です。最初のインストールの後は、ユーザーがコンポーネントを再度使用するために何らかの操作をする必要はありません。

影響

アプリの (初めての) 起動時にオプション コンポーネントのレガシ コンポーネントを参照または呼び出す場合、アプリを中断してオンデマンド機能ダイアログが開き、コンポーネントをインストールするユーザーの許可を求めます。ユーザーが [OK] をクリックすると、場合によっては昇格プロンプトも表示されます。その場合は、資格情報を入力する必要があります。その後、コンポーネントがインストールされ、アプリを再開します。

軽減策

オンデマンド機能 UI が開かないようにするために、DISM またはオプションのコンポーネント UI を使用して DirectPlay と NTVDM をプレインストールできます。

解決策

アプリの今後のバージョンでは、Windows のレガシのオプション コンポーネントにリストされているコンポーネントを参照または呼び出さないようにすることを強くお勧めします。レガシのオプション コンポーネントに含まれるのは、古く、あまり使用されないコンポーネントになるため、ユーザーにパフォーマンスとセキュリティの問題を引き起こすおそれがあります。

テスト

互換性を確保するために必要な追加テストはありません。オプションのコンポーネントが呼ばれるか参照されるときにダイアログが表示されること、インストールには昇格が必要なことに対する注意が重要です。

HTML のフォント フォールバック動作

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012

説明

Microsoft では、複数の言語向けに HTML アプリの UI フォント用ロジックを更新しています。すべての言語で単一のフォントを使用するのではなく、言語ごとに、UI フォントが決定されるようになります。たとえば、日本語向け HTML アプリの UI フォントは Meiryo UI となりました。

影響

読み取りやすいフォントのおかげでアプリの見た目は全体的に良くなりますが、以前のフォントを使用していた HTML アプリの見た目は変化する可能性があります。ピクセル単位で完璧なコンテンツ サイズに依存する固定レイアウトでは、問題が発生する可能性があります。たとえば、コンテンツの一部の行が以前よりも大きく表示されて、予期しない改行やコンテナー要素のサイズ変更が発生する可能性があります。とはいえ、実際にはまだ、既存のアプリで問題は発見されていません。

解決策

影響を受けるアプリでは、以前の既定のフォントを使用するのではなく、CSS を使用して特定のフォント (たとえば、” font-family: Meiryo UI”) を指定することで、この動作を軽減できます。以下の表に、スクリプト名ごとの UI フォントを示しています。

スクリプト名	UI フォント
アラビア語	Segoe UI
アルメニア語	Segoe UI
バングラ語	Nirmala UI
注音字母	Microsoft JhengHei UI
ブライユ点字	Segoe UI Symbol
ブギス文字	Leelawadee UI
カナダ音節文字	Gadugi
チェロキー語	Gadugi
コプト文字	Segoe UI Symbol
簡体字	Microsoft YaHei UI
繁体字	Microsoft JhengHei UI
キリル文字	Segoe UI
デバナガリ文字	Nirmala UI
デザレット文字	Segoe UI Symbol
エチオピア文字	Ebrima
ジョージア語	Segoe UI
グラゴル文字	Segoe UI Symbol
ゴート文字	Segoe UI Symbol

スクリプト名	UI フォント
ギリシャ文字	Segoe UI
グジャラート語	Nirmala UI
グルムキー文字	Nirmala UI
ヘブライ語	Segoe UI
古代イタリア文字	Segoe UI Symbol
ジャワ文字	Javanese Text
日本語	Meiryo UI
カンナダ文字	Nirmala UI
クメール語	Leelawadee UI
韓国語	Malgun Gothic
ラオス語	Leelawadee UI
ラテン文字	Segoe UI
マラヤーラム語	Nirmala UI
モンゴル文字	Mongolian Baiti
ミャンマー文字	Myanmar Text
ンコ文字	Ebrima
オガム文字	Segoe UI Symbol
オル チキ文字	Nirmala UI
突厥文字	Segoe UI Symbol
オディア語	Nirmala UI
オスマニア文字	Ebrima
パスパ文字	Microsoft PhagsPa
ルーン文字	Segoe UI Symbol
ソラング ソンペング 文字	Nirmala UI
シンハラ語	Nirmala UI
シリア語	Estrangelo Edessa
タイ ロ文字	Microsoft Tai Le
新タイ ロ文字	Microsoft New Tai Lue
タミール文字	Nirmala UI
テルグ文字	Nirmala UI
ティフィナグ文字	Ebrima
ターナ文字	MV Boli
タイ語	Leelawadee UI
チベット文字	Microsoft Himalaya
ヴァイ文字	Ebrima
イ語	Microsoft Yi Baiti

Windows 8.1 は HTTP URI を不許可で、HTTPS URI のみを許可

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012 R2

説明

Windows 8 向けに作成されたアプリでは、アプリケーション コンテンツ URI に HTTP URI と HTTPS URI を含めることができますが、Windows 8.1 向けに作成されたアプリでは HTTPS URI しか含めることができません。Windows 8.1 のこの新しい制限は、プラットフォームをさらに保護するために強化されたセキュリティ制限の一環です。

影響

Windows 8 向けに作成されたアプリを Windows 8.1 で実行する場合、ApplicationContentUriRules で HTTP URI を使用することは許可されています。

軽減策

WWA 開発者は、<iframe> から WebView コントロール (<x-ms-webview>) に切り替えることをお勧めします。ただし、AppCache、IndexedDB、位置情報、またはプログラムによるクリップボードへのアクセスのサポートが必要な場合、Windows 8.1 でも引き続き <iframe> を使用する必要があります。

引き続き <iframe> をリモート コンテンツに対して使用するには、アプリの ApplicationContentUriRules に新しいエントリが必要です。Web ビューを使用するアプリでは、ScriptNotify イベントを生成する window.external.notify を Web コンテンツで呼び出す必要がある場合、ApplicationContentUriRules 内に新しいエントリが必要です。Visual Studio がない場合は、次のような XML をサブドメインにワイルドカード (https://*.microsoft.com など) を含めて追加して、アプリ マニフェストを更新できます。

```
<Package>
    …
    <Applications>
        <Application>
            …
            <ApplicationContentUriRules>
                <Rule Match="https://www.example.com" Type="include"/>
            </ApplicationContentUriRules>
        </Application>
    </Applications>
</Package>
```

参考資料

ApplicationContentUriRules (英語)
https://msdn.microsoft.com/ja-jp/library/windows/apps/br211416.aspx

<iframe> 要素と <iframe> オブジェクト (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325297

Webview クラス
https://msdn.microsoft.com/ja-jp/library/windows/apps/windows.ui.xaml.controls.webview.aspx

WebView.ScriptNotify イベント
https://msdn.microsoft.com/ja-jp/library/windows/apps/windows.ui.xaml.controls.webview.scriptnotify.aspx

Windows 8.1 のマウスホイール入力

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012

説明

Windows 8.1 のマウスホイール イベントは、以前のバージョンの Windows のようにキーボード フォーカスに基づいて送信されることはありません。Windows 8.1 では、マウスがストア アプリ上にある場合、マウスホイール イベントはそのストア アプリに送信されます。ただし、互換性のために、マウスがデスクトップ アプリの上にある場合は、マウスホイール イベントは引き続きキーボード フォーカスに基づいて送信されます。

影響

マウスがストア アプリの上にある場合、ユーザーがそのストア アプリをクリックしなくても、マウスホイールを動かすことで対応する任意のコンテンツがスクロールされます。これはスタート画面にも適用されます。これにより、Windows 8.1 ではマウスホイールによるスクロールが Windows 8 に比べてシンプルな操作になっています。

軽減策

ほとんどの場合、既存のアプリはこの変更による影響を受けません。ストア アプリで、マウス クリック イベントの登録後にのみマウスホイール イベントをリッスンする場合は、そのアプリはユーザーが能動的にクリックするまでマウスホイールに反応しない可能性があります。したがって、マウスホイール入力で最も起こる可能性が高い問題点は、アプリの動作が Windows 8 のときと今後もまったく変わらないことです。デスクトップ アプリの場合、キーボード フォーカスを取得していてもマウスホイール入力を独占することはもうありませんが、それによりアプリが動作しなくなることも決してありません。そのため、短期的な対策は必要ありません。

解決策

ストア アプリ開発者は、マウス クリック イベントを前処理せずにマウスホイール イベントを受け取ることを想定する必要があります。たとえば、マウス クリックの登録後のみマウスホイール イベントをリッスンする必要はありません。同様に、デスクトップ アプリでは、キーボード フォーカスがあるときにマウスホイール イベントのキャプチャを試みる必要はありません (低レベルのフックを設定するなど)。

テスト

ストア アプリ開発者は、Windows 8.1 上でテストを行って、マウスがアプリの上にあるときにすべてのスクロール機能が動作することを確認する必要があります。デスク

トップ アプリ開発者は、Windows 8.1 上でテストを行って、マウスホイール イベントをキャプチャしていないことを確認します（前述のガイダンスに従います）。

Windows 高精度タッチパッド デバイス

プラットフォーム

クライアント – Windows 8.1

説明

Windows 高精度タッチパッドは、高い精度のポインター入力とジェスチャ機能を備えた新しい種類の入力デバイスです。既定では、これらのデバイスを操作すると超高精度スクロール ホイール メッセージが生成され、デスクトップ アプリで利用できます。

影響

超高精度スクロール ホイール メッセージを処理するように設計されていないアプリでは、スクロールが過剰になるか、適切にスクロールされない場合があります。

軽減策

アプリ開発者は、マウスのスクロール ホイール メッセージのスクロール デルタを確認し、その値に応じてアプリを修正する必要があります。ただし、修正するまでの間は、超高精度スクロール ホイール メッセージ (デルタ = 1) をアプリで除外し、高精度スクロール ホイール メッセージ (デルタ = 40) または標準スクロール ホイール メッセージ (デルタ = 120) を受け取ることができます。

解決策

アプリで超高精度スクロール ホイール メッセージを処理できる場合は、Windows 高精度タッチパッドからこのメッセージが既定のメッセージとして送信されるため、何も行う必要はありません。

アプリで高精度スクロール ホイール メッセージは処理できるが、超高精度ホイール メッセージは処理できない場合は、アプリ マニフェストに以下を追加します。

```
<application xmlns="urn:schemas-microsoft-com:asm.v3">
    <windowsSettings>
      <highResolutionScrollingAware
xmlns="http://schemas.microsoft.com/SMI/2013/WindowsSettings">true</hi
ghResolutionScrollingAware>
  </windowsSettings>
</application>
```

アプリで高精度スクロール ホイール メッセージも超高精度ホイール メッセージも処理できない場合は、標準スクロール ホイール メッセージが必要なことを示すためにアプリ マニフェストに以下を追加します。

```
<application xmlns="urn:schemas-microsoft-com:asm.v3">
    <windowsSettings>
      <ultraHighResolutionScrollingAware
xmlns="http://schemas.microsoft.com/SMI/2013/WindowsSettings">false</u
ltraHighResolutionScrollingAware >
  </windowsSettings>
</application>
```

Windows 8.1 におけるデスクトップ アプリケーションの高 DPI

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012

説明

Windows 8.1 では、Windows 8 でサポートされている 100%、125%、および 150% の倍率に加えて、200% 倍率の表示でデスクトップ アプリケーションが実行されることが想定されます。また、1 つの倍率がすべてのディスプレイに適用されるのではなく、ディスプレイごとにデスクトップ アプリケーションが拡大縮小されます。開発者は、Windows 8.1 の新しい API を呼び出すことで、ディスプレイごとの倍率を取得することもできます。

影響

ユーザーは、Windows 8.1 で新しい高解像度ディスプレイを使用して、高ピクセル データを活用したすばらしい画面表示を体験できます。アプリケーションが高 DPI に対応していない場合でも、Windows によって拡大縮小されます。また、ユーザーは高解像度ディスプレイと低解像度ディスプレイを同じコンピューターで組み合わせて使用でき、コンテンツは Windows によって各ディスプレイに合わせて適切に拡大縮小されます。これは Windows 8 からの強化点です。Windows 8 では、一部のディスプレイでコンテンツが収まりきらないことがあります。

軽減策

アプリケーション自体に拡大縮小する機能がなくても、Windows 側で拡大縮小が行われるので、開発者は通常、追加で作業をする必要がありません。しかし、労力を費やして高 DPI をサポートしたアプリケーションを開発すれば、競争上の優位性を得られます。そのようなアプリケーションは、高 DPI の新しいノート PC やデスクトップ モニターで見栄えが良くなるためです。

解決策

高 DPI を活用できるアプリケーションを開発することは、設計でも実装でも複雑なタスクです。チュートリアル情報、BUILD のプレゼンテーションの内容、同様のサポートについては、下記のリンクを参照してください。

テスト

アプリケーションで高 DPI を利用しない場合にも、100%、125%、150%、および 200% の倍率でアプリケーションをテストして、エンドユーザー エクスペリエンスに満足感と競争力があることを確認することをお勧めします。

参考資料

Windows 8.1 での高 DPI に関するホワイト ペーパーとチュートリアル (英語)
http://go.microsoft.com/fwlink/p/?LinkID=307061

Windows デスクトップ サンプル プログラム
http://go.microsoft.com/fwlink/p/?linkid=310129

BUILD 2013 で開かれた Windows 8.1 における高 DPI に関する分科会セッション (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325300

"内外内" シナリオや "慣性" シナリオで画面のエッジ UI を非表示にする

プラットフォーム

クライアント – Windows 8.1

説明

画面のエッジ UI (チャーム、アプリ バー、アプリの切り替え) をユーザーが意図せず呼び出してしまう場合があります。そのため、開発者がアフォーダンス (境界線など) のない全画面用 UI を設計して、ユーザーが誤って画面のエッジ (端) に触れないようにする機能を導入しました。

不注意でエッジをスワイプすることが最も多い状況は次の 2 つです。

- パン操作をすばやく行うと、その操作が速く、不正確なことが原因で、画面エッジからのスワイプだと認識される可能性があります。
- ペインティング アプリなどでは、手描き入力を指で行っているときにすばやく画面から指先を離したり触れたりすると、内から外に向かってまた内に戻る動作によって、画面エッジの UI が誤ってトリガーされ、ユーザー エクスペリエンスを妨げる可能性があります。

このユーザーと開発者のエクスペリエンスを改善するために、2 つの特定のケースで画面エッジの UI が表示されなくなりました。

- 直接操作の慣性をサポートするアプリをユーザーがパン操作し、慣性が発生しているときに画面の外側からスワイプした場合、タイムアウトするまでは画面エッジの UI が呼び出されることはありません。
- 認定デバイスでは、ユーザーがすばやく画面内から画面外に向かってスワイプしてから、特定の条件の範囲内で画面内に戻ると (内外内の動作)、画面エッジの UI が呼び出されません。

影響

アプリの使用中にユーザーが画面の端に向かってすばやくスワイプしたときに、誤って画面エッジから UI が呼び出されることが減ります。

解決策

開発者は、このような対策を念頭に置くことで、ユーザーがアプリの端に沿って操作しているときに誤って画面エッジの UI をトリガーすることを心配せずに、全画面を利用できます。

Windows ストア アプリからの ImmSetConversionStatus() または ImmGetConversionStatus() の呼び出しはサポートされない

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012 R2

説明

Windows ストア アプリによる ImmSetConversionStatus() または ImmGetConversionStatus() の呼び出しはサポートされておらず、予期しない結果を引き起こすおそれがあります。

影響

アプリの起動時に、IME 入力モードは以下の状態が既定に設定されます。

	ソフトウェア 入力 パネル	ハードウェア キーボード
KOR, JPN	オン	オフ
CHS, CHT	オン	オン

解決策

開発者は、フィールドの入力スコープ値を指定することで、既定の IME 入力モードを制御できます。

指定した入力スコープの IME 入力モードは、IME ごとに判定されます。開発者は IME 入力モードを指定することはできません。

参考資料

InputScope 列挙型 (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325301

ImmSetConversionStatus (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325302

ImmGetConversionStatus (英語)
https://msdn.microsoft.com/en-us/library/windows/desktop/dd318560(v=vs.85).aspx

ユーザーごとからスレッドごとに変化した IME 入力モード モデル

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012 R2

説明

Windows 8 では、IME 変換モード (IME Conversion Mode) と IME センテンス モード (IME Sentence Mode) がユーザー コンテキストに基づいており、1 つのアプリのモードを変えると、他のすべてのアプリにも影響が及んでいました。この動作は、コントロール パネルの地域と言語セクションの詳細設定にある [アプリ ウィンドウごとに異なる入力方式を設定する] という設定を使うことで無効にできます。

互換性を向上するために Windows 8.1 では、[アプリ ウィンドウごとに異なる入力方式を設定する] コントロールの設定にかかわらず、モードが入力コンテキストに保存されます。

Windows 8.1 では、[アプリ ウィンドウごとに異なる入力方式を設定する] 設定は、入力方法そのものの選択のみに影響します。

アプリの起動時に、IME 入力モードは以下の既定に設定されます。

	ソフトウェア 入力 パネル	ハードウェア キーボード
KOR, JPN	オン	オフ
CHS, CHT	オン	オン

影響

ユーザーがアプリの IME 入力モードを変更しても、他のアプリには影響しません。

参考資料

IME 変換モードの値 (英語)
http://go.microsoft.com/fwlink/?LinkId=327523

IME センテンス モードの値 (英語)
http://go.microsoft.com/fwlink/?LinkId=327524

入力方式マネージャーの API は簡体字中国語 IME ではサポートされない

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012 R2

説明

以下の入力方式マネージャーの API は、Windows 8.1 の簡体字中国語 IME ではサポートされていません。

- IFECommon インターフェイス
- IFEDictionary インターフェイス
- IFELanguage インターフェイス
- IImePad インターフェイス
- IImePadApplet インターフェイス
- IImeSpecifyApplets インターフェイス

影響

これらの API を使う機能は、簡体字中国語 IME を使用すると機能しません。

解決策

アプリは、ユーザーが特定の API を使うことなく目的のタスクを完了できるように設計する必要があります。

参考資料

入力方式マネージャーのインターフェイス (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325304

IFECommon (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325305

IFECommon インターフェイス (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325305

IFEDictionary インターフェイス (英語)
http://go.microsoft.com/fwlink/p/?linkid=325308

IFELanguage (英語)
http://go.microsoft.com/fwlink/?linkid=325309

IImePad (英語)
http://go.microsoft.com/fwlink/?linkid=325310

IImePadApplet (英語)
http://go.microsoft.com/fwlink/?linkid=325311

IimePlugInDictDictionaryList (英語)
http://go.microsoft.com/fwlink/?linkid=325312

IImeSpecifyApplets (英語)
http://go.microsoft.com/fwlink/p/?linkid=325313

一部の東アジア言語の IME でソフトウェア入力パネルから vKey が送信されるようになりました

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012 R2

説明

Window 8 では、東アジア言語の IME のどれかが選択されていると、ソフトウェア入力パネル キーを押しても vKey が送信されません。

Windows 8.1 では、以下の構成の場合に vKey が送信されます。

言語	IME	Windows ストア	デスクトップ
簡体字中国語	Microsoft Pinyin, Microsoft Wubi	はい	はい
繁体字中国語	Microsoft Changjie, Microsoft Quick	はい	はい
繁体字中国語	Microsoft Bopomofo	いいえ	いいえ
韓国語	Microsoft IME	はい	いいえ
日本語	Microsoft IME	いいえ	いいえ

影響

ユーザーが vKey を送信しない IME を選択する場合、vKey によってトリガーされる機能は動作しません。

解決策

上記の IME 構成を使用しないアプリは、ユーザーが vKey を必要とする機能を使わずに目的なタスクを完了できる設計にする必要があります。

ユーザーが vKey を送信する IME を選択するが、アプリの設計が vKey を予期していない場合、実行中の Windows のバージョンを認識してそれに応じて応答するようにアプリを変更する必要があります。

新しい機能と機能強化

このセクションでは、ユーザーと開発者の両方のエクスペリエンスを向上する機能強化と新機能について説明します。ただし、一部の機能強化と新機能には既存の製品や新しい製品に影響するものがあるため、開発者は注意する必要があります。以下に、新しい機能と拡張機能の一覧を示します。

- Web アプリケーション プロキシ
- ファイルの変更範囲の追跡
- プレースホルダー ファイル

Web アプリケーション プロキシ

プラットフォーム

サーバー – Windows Server 2012 R2

説明

Windows Server 2012 R2 には、リモート アクセスの役割に Web アプリケーション プロキシという新しいサービスを追加しました。この Web アプリケーション プロキシを使用すると、管理者は外部からアクセスできるようにアプリを公開できます。このサービスは、リバース プロキシとしてや、Active Directory フェデレーション サービス (AD FS) プロキシとして機能します。実際、このサービスは、Windows Server 2012 で利用されていた AD FS プロキシ サービスに置き換わるものです。

Web アプリケーション プロキシを使用すると、組織はオンプレミスの Web リソースを外部からアクセス可能にできます。同時に、認証ポリシーと承認ポリシーを AD FS で制御して、外部からのアクセスのリスクを管理できます。

影響

Active Directory フェデレーション サービス (AD FS) の役割内に AD FS プロキシ サービスはもうありませんが、代わりにリモート アクセスの役割内に Web アプリケーション プロキシが追加されています。これは、アプリ公開用のリバース プロキシ機能を組み込んで、AD FS プロキシ サービスを拡張したものです。

解決策

Web アプリケーション プロキシにアクセスするには、管理者は [サーバー マネージャー] を表示し、新しい役割や役割サービスを追加します。[リモート アクセス] の役割の下に、[Web アプリケーション プロキシ] があります。

参考資料

ソリューション ガイド
https://technet.microsoft.com/ja-jp/library/dn280942.aspx

ファイルの変更範囲の追跡

プラットフォーム

クライアント – Windows 8.1 (すべての SKU)

サーバー – Windows Server 2012 R2 (すべての SKU)

説明

NTFS ファイル システム チームが、新機能を Windows に追加しました。USN ジャーナルは、ファイルが閉じられると、ファイルの変更範囲が格納された更新シーケンス番号 (USN) レコードを出力します。新しいレコードの種類である USN_RECORD_V4 は、このようなファイルの変更範囲を記録するために導入されました。

この機能は既定では有効化されていません。有効にするには、ユーザーが、ファイル システム制御 (FSCTL) コマンドを呼び出す必要があります。ただし、他の Windows コンポーネントによって範囲の追跡が有効化される場合があるため、ユーザーと開発者はこの機能が常に有効になっていると認識している可能性があります。Windows では、開発者が USN ジャーナルにクエリを実行して、範囲の追跡が有効かどうかを調べることができます。

開発者向けに USN_RECORD_V4 へのアクセス方法を説明する MSDN ドキュメントが後日提供されます。

影響

USN ジャーナルを使用する既存のアプリはすべて今までどおり適切に動作し、互換性の問題が生じることはありません。

プレースホルダー ファイル

プラットフォーム

クライアント – Windows 8.1

サーバー – Windows Server 2012 R2

説明

プレースホルダー ファイルによって、ユーザーは接続の有無に関係なく OneDrive のファイルを表示および管理できるようになります。プレースホルダー ファイルは、ファイルがローカルにキャッシュされていない場合でも OneDrive 名前空間を表します。このファイルは、ファイルのメタデータと、写真のサムネイル画像を含みます。

影響

プレースホルダー ファイルはローカル ファイルと表示も動作もほとんど変わりません。

- アプリで共通のファイル ダイアログ ボックスを使用してファイル システムを列挙しても、アプリへの影響はありません。ユーザーが共通のファイル ダイアログ ボックスからファイルを開こうとすると、ファイルのコンテンツがダウンロードされ、アプリに渡されます。
- アプリでこのファイル システムに直接アクセスする場合、下記のガイドラインを利用して、プレースホルダー ファイルを活用できます。

解決策

- 属性に基づく規則によってプレースホルダーが非表示になります。
- プレースホルダーを特定するには、再解析タグ ID の IO_REPARSE_TAG_FILE_PLACEHOLDER を使用します。

シームレスな動作を実現するには次のシェル データ モデルを使用します。

- シェル アイテムの作成には SHCreateItemFromParsingName() を使用します。
- ストリーミングへのバインドには IShellItem::BindToHandler(BHID_Stream) を使用します。
- プロパティにアクセスするためのユーザー プロパティ ハンドラー (IShellItem2::GetPropertyHandler)
- プレースホルダーのサムネイル イメージを取得するためのユーザー シェル サムネイル ハンドラー
- BindToHandler 経由で Verb がストリームにバインドされる場合、Verb 実装で SupportedProtocols=* を指定します。

```
#include <winnth> //for IO_REPARSE_TAG_FILE_PLACEHOLDER
//     Helper that indicates this is a
bool IsFilePlaceholder(WIN32_FIND_DATA const &findData)
{
  return (findData.dwFileAttributes & FILE_ATTRIBUTE_REPARSE_POINT) &&
```

```
    (findData.dwReserved0 == IO_REPARSE_TAG_FILE_PLACEHOLDER);
}
bool IsFilePlaceholder(_In_PCWSTR filePath)
{
  bool isPlaceholder = false;
  WIN32_FIND_DATA findData;
  HANDLE hFind = FindFirstFile(filePath, &findData);
  if (hFind ! = INVALID_HANDLE_VALUE)
  {
    isPlaceholder = (findData.dwFileAttributes &
FILE_ATTRIBUTE_REPARSE_POINT) &&
    (findData.dwReserved0 == IO_REPARSE_TAG_FILE_PLACEHOLDER);
    FindClose(hFind);
  }
  Return isPlaceholder;
}
```

ツール、ベスト プラクティス、およびガイダンス

このセクションでは、既存のアプリに継続して互換性があることを確認したり、設計している新しいアプリに最適な品質と互換性を確実に持たせるうえで役立つヒントを紹介します。このセクションでは、以下のツール、ベスト プラクティス、およびガイダンスについて説明します。

- Microsoft Platform Ready テスト ツール
- 国際的なテキストとフォントのガイダンス
- Windows 8.1 向け Windows アプリ認定キット

Microsoft Platform Ready テスト ツール

プラットフォーム

クライアント – Windows 8.1 | Windows Server 2012 R2

説明

Microsoft Platform Ready (MPR) テスト ツールは、プラットフォーム アプリケーションの準備や、Windows Server 2012 アプリケーションおよび Windows Server 2012 R2 アプリケーションの認定要件が順守されていることの検証に使用されます。Microsoft は、MPR テスト ツールをソフトウェア開発やテスト プロセスに取り入れて、プラットフォーム対応を確実にしてからアプリケーションを市場にリリースすることを強く推奨しています。MPR テスト ツールは、基幹業務アプリケーションのテストや、サーバー製品の購入を決める前にプラットフォーム互換性を評価するため推奨されているツールでもあります。

要件

MPR テスト ツールには、Windows インストーラーに対する以下の自動テストが含まれます。

- セットアップと主要機能
- ドライバーとセキュリティ
- ベスト プラクティス

ほとんどのサーバー アプリの自己テストと結果の送信は 2 時間以下で完了し、複雑なアプリの場合は 4 時間前後かかる可能性があります。すべてのドライバーは Windows ハードウェア認定テストに個別に合格し、Microsoft によって Windows Server 2012 または Windows Server 2012 R2 に関して署名されている必要があります。

使用方法

アプリをテストするには、以下を行います。

1. アプリの要件に応じて、最新の Windows Server 2012 または Windows Server 2012 R2 の最小構成 (フル サーバー GUI、最小サーバー インターフェイス、または Server Core) をインストールします。
2. 認定要件を確認します。
3. クリーンなシステムで、フル UI モードまたはコマンド ライン モードから MPR テスト ツールを実行します。
4. 自己ガイド テスト プロセスを完了します。
5. 結果を確認し、必要に応じてアプリを修正します。
6. Windows Server Catalog とロゴの使用法を一覧で示している Microsoft Platform Ready ポータルにサインインして、成功した結果をアップロードします。

注 2016 年時点では、Microsoft Platform Ready ポータルの提供を停止しています。Microsoft Platform Ready (MPR) テスト ツールは、現在もダウンロード提供しています。

参考資料

Windows Server アプリ認定プログラムの要件
http://go.microsoft.com/fwlink/?LinkID=203153

Microsoft Platform Ready テスト ツール (英語)
http://www.microsoft.com/en-us/download/details.aspx?id=41676

Windows Server 2012 アプリ認定プログラム (英語)
http://go.microsoft.com/fwlink/?LinkID=241671

Windows Server ロゴに関するフィードバック
WSLogoFB@microsoft.com

国際的なテキストとフォントのガイダンス

プラットフォーム

クライアント – Windows 8 | Windows 8.1

サーバー – Windows Server 2012 |Windows Server 2012 R2

説明

Windows 2000 以前から、Windows の各メジャー リリースでは、新しいスクリプトに関するテキスト表示サポートが追加されてきました。各メジャー リリースで行われた変更についての説明は、グローバライゼーション デベロッパー センターの「Windows のスクリプトとフォントのサポート」(英語) の記事で確認できます。

スクリプトのサポートには、テキスト スタック コンポーネントへの特定の変更およびフォントへの変更が必要となる可能性があります。Windows OS には、DirectWrite、GDI、Uniscribe、GDI+、WPF、RichEdit、ComCtl32 など、多くのテキスト スタック コンポーネントがあります。提供される情報は、主に GDI および DirectWrite に関連します。また、一般的にこの情報は、RichEdit、または Windows 8 ストア アプリや Web コンテンツの表示に使用される MSHTML レンダリング エージェントなどの UI フレームワークにも適用できます (ただし、コンポーネントごとに多少の違がある可能性があります)。

ベスト プラクティス

開発者向けのタイポグラフィ ガイダンス

タイポグラフィは、Microsoft デザイン言語で最も重要な部分です。Microsoft デザイン原則それぞれで、タイポグラフィの重要性が強調されています。初めての場合、アプリ開発者には、高度なタイポグラフィ機能をサポートする一連のフレームワークがあります。JavaScript と HTML を使用して Windows ストア アプリのテキストを表示および編集する方法に関する情報については、「テキストの表示と編集」を参照してください。

フォント メトリックス

フォント メトリックスは、アプリ開発者にとって特に重要な領域です。フォント ファイルには、垂直メトリックスと水平メトリックスに関係したさまざまな値が含まれています。これらの値は、OpenType の仕様で文書化されており、さまざまな API (DWRITE_FONT_METRICS 構造体および TEXTMETRIC 構造体を参照) を通じて公開されます。

参考資料

Windows のスクリプトとフォントのサポート (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325315

グローバライゼーション デベロッパー センター (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325339

テキストの表示と編集
https://msdn.microsoft.com/ja-jp/library/windows/apps/hh465442.aspx

OpenType の仕様 (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325342

DWRITE_FONT_METRICS 構造体
https://msdn.microsoft.com/ja-jp/library/windows/desktop/dd368074(v=vs.85)

TEXTMETRIC 構造体 (英語)
http://go.microsoft.com/fwlink/p/?LinkId=325344

Windows 8.1 向け Windows アプリ認定キット

プラットフォーム

クライアント – Windows 7 | Windows 8 | Windows 8.1

説明

Windows アプリ認定キット 3.1 は、Windows SDK for Windows 8.1 に含まれています。このキットを使用すると、Windows 8、Windows 8.1 向けの Windows ストア アプリを事前認定してから配布準備ができます。また、Windows 7、Windows 8、Windows 8.1 のデスクトップ アプリ認定プログラムに関する認定もできます。Windows アプリ認定キット 3.1 は更新され、シームレスなユーザー エクスペリエンスが開発者に提供されます。強化点を 2 ～ 3 挙げると、テストの並列実行による全体的な時間の短縮、テストの精選、レポート機能の強化などが行われています。

注: アップグレードを求めるメッセージが表示された場合は、お使いの Windows アプリ認定キットのバージョンをアップグレードして、キットに含まれる最新の機能を利用できるようにしてください。最新バージョンの Windows アプリ認定キットを使用してテストすると、アプリが Windows ストアのポリシーおよびデスクトップ認定プログラムに対してより適切に準拠していることを確実にできます。

参考資料

Windows アプリ認定キット
Windows アプリ認定キット 3.0 の概要、バージョン 3.0 の新機能、および Windows RT 向け Windows アプリ認定キットを入手するためのリンクを提供します。
http://go.microsoft.com/fwlink/p/?linkid=506773

Windows アプリ認定キットの使用
ストア アプリの開発に関する前提条件および Windows アプリ認定キットを使用してストア アプリを検証する方法の一覧を示します。
http://go.microsoft.com/fwlink/p/?LinkId=267228

認定 (英語)
デスクトップ アプリの認定に役立つさまざまなリソースを提供します。
http://go.microsoft.com/fwlink/p/?LinkId=227193

アプリの認定
Windows ストアに関する認定プロセスを説明します。
https://developer.microsoft.com/ja-jp/store/publish-apps

Windows RT 搭載 PC でのデバッグとテスト
Windows RT 向け Windows ストア アプリ開発のデバッグとテストを取り上げます。
https://msdn.microsoft.com/ja-jp/library/windows/apps/xaml/bg126235.aspx

Windows 8

本セクションでは、Windows 8 固有のトピックを解説します。トピックのいくつかは、Windows Server 2012 にも適用できます。トピックは、次の 3 種類に分類して扱います。

- クライアントとサーバーの互換性
- 新機能と機能拡張
- ツールとベスト プラクティス

クライアントとサーバーの互換性

本セクションでは、既存のアプリに影響を及ぼす可能性があるため、特に注意する必要のあるオペレーティング システムの変更点を示します。また、クライアント、サーバー、またはクライアントとサーバーの両方に関する、新規アプリの設計上の注意点を説明します。本セクションで扱うトピックの機能領域別リストを以下に示します。各機能領域のトピックは影響力の大きい順に記載しています。

アプリの互換性

- セキュリティ アプリの検出ルールの更新
- Shim のステータスの確認
- サーバー アプリにはサーバー グラフィック シェルなしでの実行機能が必要
- Windows 8 から削除された RDS サーバーのコンポーネント
- ファイルの種類とプロトコルの関連付けモデル
- Desktop Activity Moderator
- 既定の .NET Framework 4.5 とオプションの .NET Framework 3.5
- 既定の Web ブラウザーまたは Windows 8 アプリを起動するとデスクトップ アプリが隠れて見えなくなる
- ハイ コントラスト モード
- アプリ (実行可能ファイル) マニフェスト
- キューを使った表示モデルの廃止
- Windows 8 のプログラム互換性アシスタント シナリオ
- デスクトップ ガジェットの削除

記憶域とファイル システム

- Advanced Format (4K) ディスクの互換性の更新
- 論理単位の仮想プロビジョニング
- WinPE とサーバー SKU でオプションとなった拡張記憶域
- VDS から WMIv2 ベースの Windows 記憶域管理 API への移行
- ローカル ボリュームの [以前のバージョン] UI の削除
- MSAHCI から StorAHCI への移行
- Windows 7 の "バックアップと復元" の廃止
- オフロード データ転送

その他

- デスクトップ ウィンドウ マネージャーの常時有効化
- Internet Explorer 9 では Direct2D による "リッチ" メタファイルのレンダリングがサポートされない
- DX9 レガシ ハードウェアのサポートに関する変更点
- Windows ストア アプリでは MSAA を利用できない
- NDIS 6.30 ドライバーでのポート 3 の廃止

セキュリティ アプリの検出ルールの更新

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows 8 に追加される Windows ストア アプリはすべて、Program Files (%programfiles%) の下位にある共通ディレクトリ \program files\WindowsApps にインストールされます。

影響

既存の構成と競合を起こす可能性があるほか、一部のウイルス/マルウェア検出ツールでは、このディレクトリが問題の発生源になる可能性の高い場所と見なされます。

軽減策

現在の推奨策は次のとおりです。

- カスタム アプリのインストール先を \program files\WindowsApps 以外の場所にする
- ウイルス/マルウェア対策ツールのベンダーがヒューリスティックを更新し、上記の場所をマルウェアの検出場所に指定しないようにする

Shim のステータスの確認

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

さまざまな理由から shim を介してアプリを実行する場合、Windows 8 によってイベントがログに記録されます。アプリが shim を介して実行されているかどうかを確かめるには、イベント ビューアーを確認するのが最も簡単な方法です。この確認には、Applications and Services Logs\Microsoft Windows Application-Experience Program\Telemetry にアクセスします。

軽減策

shim を介さずにアプリを実行するには、実行可能ファイルの名前を変更するか、メジャー バージョン番号を 1 つ増やす (実行可能ファイルを再コンパイルする) のが最も良い方法です。

サーバー アプリにはサーバー グラフィック シェルなしでの実行機能が必要

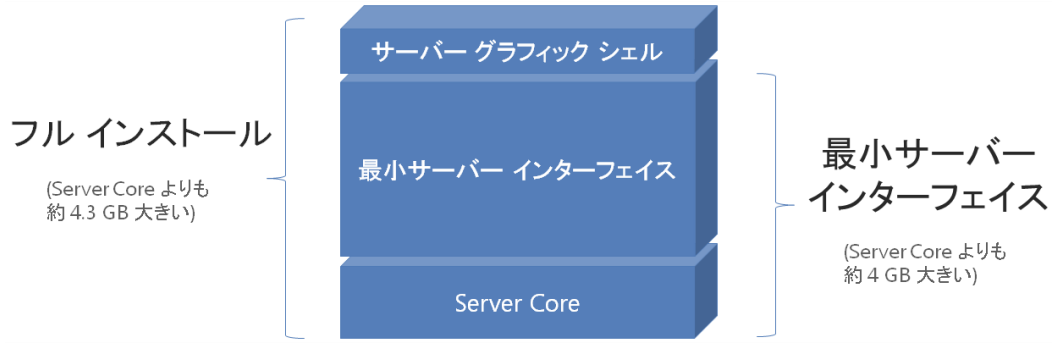
プラットフォーム

サーバー – Windows Server 2012

説明

エクスプローラーと Internet Explorer を含むサーバー グラフィック シェル機能は、Windows Server 2012 の "フル インストール" 時に既定でインストールされます。このサーバー グラフィック シェル機能をアンインストールすると、サービスやパフォーマンスの潜在的な負荷を軽減できます。その結果、管理ツールをサーバー上でローカルに実行しながら、サーバーの再起動の回数を抑えられます。

管理者がサーバー グラフィック シェルをアンインストールすると、サーバーが "最小サーバー インターフェイス" 構成になります。



管理者は、フル インストール構成の代わりに、この最小サーバー インターフェイス構成 (ローカル管理ツール セットを含む) を既定として使用することも可能です。この最小構成では、ローカルの監視および管理が可能で、同時にリソース使用率とサービスの提供頻度の両方を低減させることができます。

後からサーバー グラフィック シェルの機能が必要になった場合は、管理者はサーバー グラフィック シェル機能を再インストールできます (また、Server Core としてインストールしてから、Features On Demand 機能を使用して最小サーバー インターフェイス構成に "ビルドアップ" することも可能です)。

リソース使用率とサービス負荷の削減というメリットを得るには、サーバー用アプリを、この最小サーバー インターフェイス構成で実行できる必要があります。これは、アプリのうちの、サーバー グラフィック シェルを必要とする部分をインストールしないように管理者が選択できるようにするか、サーバー グラフィック シェルがインストールされていることを検知したら、アプリの一部の要素を無効化することで達成されます。

最小サーバー インターフェイスによってリソース使用率とサービス負荷が減少する理由は、この構成のサーバー グラフィック シェルに含まれる多くの API とバイナリが、この最小構成ではサポートされないためです。

場合によっては、サーバー アプリを、Windows Server または Windows クライアントをインストールした他のマシンからのリモート管理 (可能であれば Windows PowerShell リモート処理経由) に対応

させる必要があります。そうすることにより、1 台または複数台の最小サーバー インターフェイス構成のマシンや、Server Core などのより小さな構成のマシンを一元的に管理しやすくなります。

影響

最小サーバー インターフェイス構成では利用できない API やバイナリを必要とする場合、アプリは画面に正しく表示されなかったり、使用できなくなる可能性があります。

軽減策

サーバー アプリ開発者は、アプリのどの部分が削除された API またはバイナリを必要とするかを特定し、サーバー管理者向けの情報として、最小サーバー インターフェイスを使った際にアプリの所定の部分が正しく機能しないことを知らせる必要があります。アプリの該当部分のインストールが任意の場合や、製品の動作上必ずしも必要ない場合は、引き続き最小サーバー インターフェイス構成でアプリをインストールし、実行できます。

サーバー グラフィック シェルがないとアプリをまったく使用できない場合は、その旨を文書に記載し、サーバー管理者にサーバー グラフィック シェルをインストールするように指示する必要があります(Server Core 構成へのインストールの場合は、Features On Demand を使用して機能を追加する必要があります)。さらに、アプリをインストールする前や後に、随時サーバー グラフィック シェルをアンインストールできるよう、必要なファイルがすべて利用可能な状態かどうかを、起動時にアプリがチェックする必要があります。

解決策

依存関係を最小限に抑え、アプリをモジュール化することで、それ以上インストール済みの負荷の高いユーザー インターフェイス コンポーネントを使用しなくても、アプリのコア機能が動作するようにします。削除された API またはバイナリを一切必要とせず、代わりに最小サーバー インターフェイスまたはサーバー コア内に含まれる機能のみで動作するアプリを開発します。これにより、メンテナンス要件が減り、パフォーマンスとユーザーの満足度が向上します。

サーバー グラフィック シェルを利用すると機能を大幅に追加可能なアプリの部分について、アプリケーション開発者は以下の対応が可能です。

- サーバー グラフィック シェルを利用したこれらの追加の機能をオプションでインストールできるようにし、最小サーバー インターフェイス構成ではインストールを省略可能にする
- サーバー グラフィック シェルがインストールされていることを検知し、アプリの動作を適応させる

また、アプリケーション開発者は、可能かつ適切な場合にサーバー アプリをリモート管理できるようにする必要があります。

最小サーバー インターフェイスおよびサーバー コアの検出

Windows Server は、インストール先の各サーバー レベルで、対応するレジストリ値をインストールします。こうしたキーの有無を照会することで、サーバー グラフィック シェルや最小サーバー インターフェイス機能がインストールされ、有効化されているかどうかを確認できます。

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Server\ServerLevels で以下のキーを照会します。

	Server Core	Minimal Server Interface	Server Graphical Shell
ServerCore=1	X	X	X
Server-Gui-Mgmt=1		X	X
Server-Gui-Shell=1			X

上の表の "X" は、対応する機能がインストール済みの場合に、そのレジストリ キーが存在することを意味します。

これらのサーバー レベルは付加的な関係にある点に注意してください。つまり、サーバー グラフィック シェルがインストールされていれば、最小サーバー インターフェイスおよびサーバー コアもインストールされています。その場合は、"両方の" レジストリ キーが存在します。

テスト

アプリのコードを検証し、削除された API とバイナリを必要とする部分を探します。"コア アプリケーション" バイナリから対象のインスタンスをすべて削除した後、サーバー グラフィック シェルを含まない環境でアプリをテストします。この作業には、プロセス モニターなどのツールが役立ちます。

対象の API およびバイナリの使用を完全に中止できない場合は、最小サーバー インターフェイスまたはサーバー コアで実行中にアプリが適切に終了するようにします。

参考資料

Server Core に関する既存の MSDN ドキュメント
http://go.microsoft.com/fwlink/p/?linkid=325406

Windows 8 から削除されたリモート データ サービス サーバーのコンポーネント

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

リモート データ サービス (RDS) サーバーは、Windows 8 では利用できません。

- 既定の "ビジネス オブジェクト" である RDSServer.DataFactory をホストする msadcf.dll ファイルが削除され、その関連ファイル (msadcfr.dll、msadcfr.dll.mui、handler.reg、handsafe.reg) が削除されました
- 既定のハンドラーである msdfmap.dll ファイルが削除され、msdfmap.ini ファイルが削除されました
- ISAPI である msadcs.dll ファイルが削除されました

影響

ユーザーが Windows 8 上で RDS サーバーをホストできません。

軽減策

ユーザーは RDS サーバーを Windows 7 またはその他のダウンレベルのオペレーティング システムのコンピューター上で維持できます。

解決策

ユーザーは RDS サーバーを Windows 7 またはその他のダウンレベルのオペレーティング システムのコンピューター上で維持できます。

参考資料

データ アクセス テクノロジのロードマップ
http://msdn.microsoft.com/ja-jp/library/ms810810.aspx

ファイルの種類とプロトコルの関連付けモデル

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

ファイルの種類とプロトコルの関連付けモデルは、Windows 8 で変更されました。プログラムでファイルの種類またはプロトコルの既定のアプリを設定することはできなくなりました。代わりに、ユーザーは常にファイルの種類またはプロトコルと既定のアプリの関連付けを制御する必要があります。

影響

この変更をユーザーに提供する方法は、アプリの設計に依存します。次に例を示します。

- 多くのアプリは動作のたびに、自身が既定のアプリであるかどうかをチェックします。既定のアプリでない場合は、ユーザーに自身を既定のアプリとして設定するように促します。しかし、アプリはファイルの種類またはプロトコルと関連付ける既定のアプリがどのアプリなのかを正確に問い合わせることができないため、これらはもはや動作しません。
- 多くのアプリは自身が既定のアプリとして設定されるための組み込みのダイアログ ボックスまたはメニューもしくはインストーラーを備えています。しかし、プログラムでファイルの種類またはプロトコルと既定のアプリを関連付けることができないため、これらはもはや動作しません。

軽減策

ユーザーがこれらの変更に対応するためのいくつかの方法を次に示します。

- 既定のアプリが指定されていない場合、ファイルの種類、プロトコル、またはファイルの種類とプロトコルの両方を処理する既定のアプリを選択するようにユーザーを誘導します。
- 新しいアプリをインストールした後に、ファイルの種類またはプロトコルと既定のアプリの関連付けを変更するためのオプションがユーザーに提供されます。
- 既定のプログラム コントロール パネルは、ファイルの種類、プロトコル、またはファイルの種類とプロトコルの両方の既定のアプリへの関連付けを提供します。アプリはコントロール パネルに接続することができます。
- 既定のアプリは、エクスプローラーから変更することができます。

解決策

これらの変更の結果として、以下の API ガイダンスが提供されます。

- IApplicationAssociationRegistration API 内のいくつかのメソッドの呼び出しの機能が変更されているため、**使わない**ようにしてください。
    - アプリが既定であるかどうかを確認するために、QueryAppIsDefault/QueryAppIsDefaultAll を呼び出さ**ないでください**。
    - 現在の既定のアプリ (設定されていれば) を確認するために、QueryCurrentDefault を呼び出さ**ないでください**。
    - 既定のアプリを設定するために、SetAppIsDefault/SetAppIsDefaultAll を呼び出さ**ないでください**。

- 進行中のガイダンスは、以下のとおりです。
    - ファイルの種類またはプロトコルと既定のアプリの関連付けを確認するために照会**しないでください**。
    - ファイルの種類またはプロトコルと既定のアプリの関連付けの変更を監視しようと**しないでください**。
    - ファイルの種類またはプロトコルと既定のアプリの関連付けを設定しようと**しないでください**。
    - アプリでファイルの種類またはプロトコルと既定のアプリの関連付けを管理しようと**しないでください**。
    - アプリのユーザーに default management UI へのアクセスを許可する場合は、既定のプログラム コントロール パネルと統合してください (アプリの management UI はサポートされていません)。
        - 呼び出し IApplicationAssociationRegistrationUI::LaunchAdvancedAssociationUI は、ユーザーが指定されたアプリの既定のプログラム コントロール パネル ページにアクセスすることを許可します。

テスト

- アプリが既定のプログラム コントロール パネルで適切に登録されていることを確認します。
- アプリが操作できるようにファイルの種類、プロトコル、またはファイルの種類とプロトコルの両方がファイルを開く既定のプログラムのリストで適切に登録されていることを確認します。
- アプリをインストール後、新しいアプリの通知が表示され、ユーザーがファイルの種類、プロトコル、またはファイルの種類とプロトコルの両方をアプリと関連付けられることを確認します。

参考資料

Windows Developer 8.1 のデスクトップ アプリにおけるファイルの種類とプロトコルの関連付けのためのベスト プラクティス
http://go.microsoft.com/fwlink/?LinkId=228165

ファイル タイプの関連付けと AutoPlay についての //build/ でのプレゼンテーション
http://go.microsoft.com/fwlink/p/?linkid=325427

Desktop Activity Moderator

プラットフォーム

クライアント – Windows 8

注: DAM はコネクト スタンバイをサポートする Windows 8 クライアント マシンだけに存在します。DAM はサーバー SKU には存在しません。

注: Windows ストア アプリは、DAM に影響を受けません。

説明

お客様のコンピューターに対するニーズは、より軽くて小さい、より多くのモバイル プラットフォームへとシフトしています。モバイル デバイスへの移行の一環として、ユーザーはそのデバイスのバッテリ寿命についてますます懸念しています。Desktop Activity Moderator (DAM) は、Windows 8 の新機能の 1 つであり、コネクト スタンバイをサポートするデバイスで、一貫した長いバッテリ寿命を確保します。

デバイスの電源を入れるとコネクト スタンバイの状態になります。しかし、画面はオフになります。この電源状態では、システムは常に技術的に "オン" です (Windows ストア アプリでメール、VoIP、ソーシャル ネットワーク、インスタント メッセージのような重要なシナリオをサポートするため)。この状態は、ユーザーがスマート フォンの電源ボタンを押したときの状態に似ています。

このように、ソフトウェア (アプリやオペレーティング システム ソフトウェアを含む) はコネクト スタンバイで正常に動作する必要があります。DAM はデスクトップ アプリをスリープ状態 (ACPI デバイスの S3) と同様の方法で制御するために作成されました。コネクト スタンバイ エントリ上のシステムを横断してデスクトップ アプリのプロセスを中断または抑制することで、これを実現しています。このコネクト スタンバイは、Windows ストア アプリが実行されている間、最小限の電力消費と長く一貫したバッテリ寿命を保証します。

詳細

DAM は、システムがコネクト スタンバイをサポートする場合、システム起動時に読み込まれて初期化されるカーネル モード ドライバーです (これは `CallNtPowerInformation` によって返される `SYSTEM_POWER_CAPABILITIES` 構造体の AOAC フィールドが `TRUE` に設定されるかどうかを評価することで決定されます)。

DAM が有効になっている際に、デスクトップのプロセスが作成されたとき、DAM はそのプロセスをシステム管理のジョブ オブジェクトに追加します。

- プロセスがセッション 0 で作成された場合、DAM はプロセスを**抑制する**ためにジョブ オブジェクトに追加します。
- プロセスがインタラクティブ セッション (セッション 1 以上) で作成された場合、DAM はプロセスを**中断する**ためにジョブ オブジェクトに追加します。

注: Windows 8 において、ジョブ オブジェクトは入れ子にすることができます。つまり、ジョブ オブジェクトの DAM の使用は、アプリの既存のジョブ オブジェクトの使用を妨げることはありません。

画面がオンの場合、DAM は解除され、システム上のどのプロセスにも影響を与えることはありません。システムがコネクト スタンバイのとき、システムのアクティビティによっては DAM はプロセスを抑制または中断するかもしれません。

- 中断の対象となるプロセスは、すべてのスレッドが中断され (いかなる状況でも実行することはできません)、アプリの状態 (プロセス メモリ) が維持されます。
- 中断から中断解除されるまでの間に抑制の対象となるプロセス (これらの時間の大半は中断状態において費やされます)。
    - Windows が、重要なアクティビティが発生したことを検知し、このアクティビティが実行されている長時間の間に抑制されたサービスの中断を解除する可能性がある点に注意してください。
    - また、コネクト スタンバイの間、センサーとネットワークが利用できない可能性がある点に注意してください。抑制されたプロセスはネットワーク状況の悪化に対応した設計でなくてはいけません (ほとんどのプロセスにおいては、この変更の必要はありません)。

DAM 中断の予約が行われたか、予約が解除された際に、DAM はメッセージ配信を選択した中断プロセスに WM_POWERBROADCAST メッセージを配信します (API 呼び出しまたは互換性 Shim は後述します)。数秒の遅延後に DAM はプロセスを中断します。

DAM 抑制の予約が行われた、あるいは予約が解除された際に通知は行われません。プロセスは、遅い速度ではありますが機能し続けます。

影響

プロセスは、コネクト スタンバイの間にしばしば中断または抑制されます。中断されたアプリの大半にとって、これは S3 の中断 / 再開、または S4 の休止 / 再開と非常に類似しています。また、uptime/runtime と wall clock time の不整合や、タイマー動作の不整合、または完全に中断する前後のオペレーティング システムの状態における劇的な変化の不整合などの症状が含まれる場合がありますが、この限りではありません。

中断と抑制は単独で行われます (すべての中断可能なプロセスは単独で中断と中断解除が行われます。また抑制可能なプロセスは、単独で抑制と抑制解除が行われます)。このため、2 つの中断されたプロセスまたは 2 つの抑制されたプロセスの間の通信が問題を引き起こすことはありません。

プロセス間通信に依存するソフトウェアは、次のような特別な考慮が必要になる場合があります。

- **セッション 0 (抑制) のプロセスとセッション 1+ (中断) のプロセス間通信** – たとえばトレイ アイコンまたは現在のサービスの状態を表す UI コンポーネント
- **ユーザー モード (セッション 0 または 1) とドライバー (抑制も中断もしていない) のコンポーネント間通信** – たとえばドライバーに代わって動作するサービス

プロセス間通信が正しく処理されていない場合は、アプリはハングアップまたは無反応であるように見えることがあります (コネクト スタンバイ時に画面がオフになっているのと同様に、ユーザーは直接影響を受けないかもしれません)。ただし、ほとんどの場合、サービスおよびドライバーはプロセス間通信の問題に対して堅牢に開発されるべきです。

ソフトウェアを作成するベンダー、あるいはそれに依存するベンダーにおいては、Web のプロセス中断が接続ライフタイムとハンドシェイクにどのような影響を及ぼすかについて検討しなければいけません。さらに、セッション 0 で作成されるプロセスの開発者は、コネクト スタンバイ モードの際にネットワーク接続を使えないため、断続的なネットワーク接続がどのようにプロセスに影響を及ぼすかについて、特に注意する必要があります。

解決策

Windows ストア アプリは、DAM に影響を受けません。デスクトップ アプリが DAM に影響を受ける場合は、以下のいずれかの方式に従って、中断される (たとえば、状態の保存やネットワーク通信が閉じられる) 前に通知を要求することができます。

- アプリがウィンドウ (HWND) を保持しており、ウィンドウ プロシージャをとおして通知を扱いたい場合は、メッセージを登録するために、RegisterSuspendResumeNotification を呼び出してください (また、登録を解除する場合は UnregisterSuspendResumeNotification を呼び出してください)。フラグ パラメーターの DEVICE_NOTIFY_WINDOW_HANDLE を使用することができ、受け取りパラメーターとしてウィンドウの HWND を渡すことができます。受け取られたメッセージは、WM_POWERBROADCAST メッセージです。
- アプリがウィンドウ (HWND) を保持していない、または直接コールバックをしたい場合は、メッセージを登録するために、PowerRegisterSuspendResumeNotification を呼び出してください (また、登録を解除する場合は PowerUnregisterSuspendResumeNotification を呼び出してください)。フラグ パラメーターの DEVICE_NOTIFY_CALLBACK を使用する必要があり、受け取りパラメーターの PDEVICE_NOTIFY_SUBSCRIBE_PARAMETERS 型の値を渡す必要があります。
- アプリを再コンパイルできない場合は、アプリ互換性ツールキット (PromoteDAM shim を使う) をとおして WM_POWERBROADCAST メッセージを受信することができます。

中断メッセージは、`wParam=`**`PBT_APMSUSPEND`** である `WM_POWERBROADCAST` です。このメッセージは、システム上のすべてのプロセスに同時にブロードキャストされます。アプリは迅速かつ効率よく中断処理を実行する必要があります。ブロードキャスト通知後のタイムアウトはプロセスごとにグローバルであるため、この中断処理が広範囲にわたる場合は、プロセスはシステム リソースの競合に見舞われる可能性があります。

中断解除メッセージは、`wParam=`**`PBT_APMRESUME`** である `WM_POWERBROADCAST` です。このメッセージは、システム上のすべてのプロセスに同時にブロードキャストされます。コネクト スタンバイからシステム終了するまでの相対時間は保証されません。

テスト

コネクト スタンバイの遷移をまたいでソフトウェアをテストします。

参考資料

Windows 7 用モバイル バッテリの寿命に関するソリューション
http://go.microsoft.com/fwlink/p/?linkid=325428

SYSTEM_POWER_CAPABILITIES
http://go.microsoft.com/fwlink/p/?LinkId=325429

CallNtPowerInformation 関数
http://go.microsoft.com/fwlink/p/?linkid=325430

ジョブ オブジェクト
http://go.microsoft.com/fwlink/p/?LinkId=325431

RegisterSuspendResumeNotification
http://go.microsoft.com/fwlink/?LinkId=239911

UnregisterSuspendResumeNotification
http://go.microsoft.com/fwlink/?LinkID=239912

PowerRegisterSuspendResumeNotification
http://go.microsoft.com/fwlink/?LinkId=239913

PowerUnregisterSuspendResumeNotification
http://go.microsoft.com/fwlink/?LinkID=239914

アプリ互換性ツールキット
https://www.microsoft.com/download/details.aspx?id=30652

既定の .NET Framework 4.5 とオプションの .NET Framework 3.5

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows 8 では、.NET Framework 4.5 が既定で有効化されます。Windows 8 は .NET 3.5 を既定では含んでいませんが、オプションの機能として、.NET 3.5 用のファイルが Windows 8 インストール メディアに収録されています。

Windows 7 から Windows® 8 にアップグレードする場合は、コンピューター上のすべてのアプリが引き続き正常に動作するよう、.NET Framework 3.5 が完全に有効化されます。

影響

Windows® 8 をクリーン インストールし、その後 .NET Framework 3.5 (または 2.0) に依存するアプリをインストールすると、必要な .NET 3.5 ファイルが要求されます。この不足ファイルは通常、Windows Update を通じて (ユーザーの許可を得たうえで) ダウンロードされますが、Windows Update にアクセスできない場合、不足ファイルの代わりとなるソースが特定されなければ .NET Framework 3.5 の有効化に失敗します。

軽減策

Windows® 8 をクリーン インストールしたテスト マシンのみで .NET Framework 3.5 を有効にするには、以下の手順を実行します。

1. マウントされたオペレーティング システムのビルド ISO イメージから、\sources\sxs\ をコピーして dotnet35 または同様のフォルダーに格納します。以下に例を示します。

```
xcopy e:\sources\sxs\*.* c:\dotnet35 /s
```

2. 管理者権限で、次のコマンド ラインを実行します。

```
Dism.exe /online /enable-feature /featurename:NetFX3 /All /Source:c:\dotnet35
/LimitAccess
```

注: **sources\SxS** フォルダーを再配布メカニズムとして使用しないでください。このメカニズムはサポートされません。

解決策

コンシューマー向け:

Windows 8 には、再頒布可能パッケージをインストールしようとした場合や、.NET 3.5 を必要とするアプリケーション インストーラーによって再頒布可能パッケージが起動された場合に、.NET Framework 3.5 を自動的に有効化するメカニズムが含まれます。

アプリ開発者 (および IT 管理者) 向け:

IT 管理者は、(既存のインストール状況に応じて) .NET 3.5 と .NET 4.5 のいずれかを使用して .NET 3.5 アプリを実行するように設定できます。3.5 と 4.5 のどちらか一方でマネージ アプリを実行するには、アプリケーション構成ファイル内にセクションを追加します。これにより、.NET 3.5 がインストールされている場合は .NET 3.5 を使用してアプリを実行し、それ以外の場合は .NET 4.5 が使用されるようになります。次に、構成ファイルに追加するセクションの例を示します。

```
<configuration>
   <startup>
      <supportedRuntime version="v2.0.50727"/>
      <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5"/>
   </startup>
</configuration>
```

エンタープライズまたは OEM 向け:

EEAP ビルド、および Windows Update にアクセスできないアプリケーションで .NET Framework 3.5 を有効にするには、次の手順を実行します。

1. マウントされた OS のビルド ISO イメージから、\sources\sxs\ をコピーして dotnet35 または同様のフォルダーに格納します。以下に例を示します。

```
xcopy e:\sources\sxs\*.* c:\dotnet35 /s
```

2. レジストリ キーを設定します。

```
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Servicing]
"LocalSourcePath"="C:\dotnet35"
```

エンタープライズ向け:

WSUS を使用してサービスを提供するようにマシンが構成されている場合は、レジストリ エントリを設定することで、Windows Update を使用して、WSUS の代わりに .NET 3.5 をマシンで有効化できるようになります (この設定後も、サービスは引き続き WSUS を通じて提供されます)。

1. レジストリ キーを設定します。

```
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Servicing]
"RepairContentServerSource"=DWORD(2)
```

このレジストリ エントリをグループ ポリシー経由で設定し、.NET 3.5 を有効化する際に、WSUS サーバーをバイパスすることも可能です。このポリシーには、[ローカル コンピューター ポリシー] -> [コンピューターの構成] -> [管理用テンプレート] -> [システム] からアクセスできます。[Specify settings for optional component installation and component repair (オプション コンポーネントのインストールおよびコンポーネントの修復のための設定を指定する)] 設定をオンにします。

[Contact Windows Update directly to download repair content instead of Windows Server Update Services (WSUS) (Windows Update と直接通信し、Windows Server Update Services (WSUS) の代わりに修復コンテンツをダウンロードする)] をオンにすると、Windows の機能 (.NET Framework 3.5 など) や修復機能を追加しようとするたびに、Windows Update を通じてファイルがダウンロードされます。このオプションを使用するには、対象のコンピューターがインターネットおよび WU にアクセスできる必要があります。WSUS がソースとして設定されている場合、通常のサービスの処理には、引き続き WSUS が使用されます。

レジストリ エントリによるローカル ソースの場所の設定

レジストリ エントリを通じて .NET 3.5 ファイル用のローカル ソースの場所を設定すると、IT 管理者はソースの場所を指定することなく [Add/Remove Windows Features (Windows の機能の有効化または無効化)] ダイアログを使用して、ペイロードのない各種機能を有効化できるようになります。レジストリ エントリの値は、グループ ポリシーで制御できます。

サポートされるレジストリ エントリを以下に示します。

エントリ	型	説明
LocalSourcePath	REG_EXPAND_SZ	既定で使用されるローカル ソースのパス。コンマ (,) で区切ると、複数のパスを指定できます。場所が複数の場合は、指定した順番で検索が行われます。 DISM コマンド ラインで指定したローカル ソースの場所は、このレジストリ エントリで指定した場所よりも優先されます。このレジストリ エントリでは、フォルダーの場所も指定できます。 WIM も使用できますが、WIM ファイルへのパスを指定する必要があります。マウントする必要はありません。次に例を示します。 wim:\\machine\share\file.wim:1 末尾の "1" に注意してください。WIM ファイル内のイメージを使用するには、数値インデックスで指定する必要があります。 マウントされた WIM のソース パスについては、マウント ポイントではなく、マウントされたイメージの Windows ディレクトリを参照する必要があります (たとえば、/source:<mount_point> ではなく /source:<mount_point>\windows)。

参考資料

レジストリベースのポリシーの実装
http://go.microsoft.com/fwlink/p/?linkid=325432

既定の Web ブラウザーまたは Windows ストア アプリを起動するとデスクトップ アプリが隠れて見えなくなる

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows ストア のユーザー エクスペリエンスでは、一度に 1 つだけアプリを使うことを前提に、マルチタスク、アプリの切り替え、Windows UI を使った通知の機能を提供します。Windows ストア アプリは、画面上で利用可能なスペースをすべて使うようにサイズ調整されており、最も一般的なビューが全画面表示されます。そのため、システム上の別のアプリを起動した場合、使用中のアプリと新たに開くアプリが、デスクトップ ウィンドウで一緒に表示されるとは限りません。これは Windows ストア アプリと、新しい Windows エクスペリエンスを採用したブラウザーについて常に当てはまります。特定の状況では、引き続きその他のアプリを同時に表示できることもあります。たとえば、デスクトップから別のデスクトップ アプリを起動する場合や、アプリをスナップしている場合などです。

影響

デスクトップ アプリに一般的なアプリのアクティブ化手法が使われている場合 (ファイルやプロトコルに関する ShellExecute API の使用など)、Windows は対象の登録に関連付けられたアプリを起動します。このアプリは Windows ストア アプリの場合もあれば、ユーザーの既定の Web ブラウザーの場合もあります (既定の Web ブラウザーでは、デスクトップ インターフェイスまたは新しい Windows UI を使っている可能性があります)。Windows ストア アプリは全画面表示で起動されるため、デスクトップおよび起動元のデスクトップ アプリは見えなくなります。

注: Windows 8 では、Internet Explorer 10 がユーザーの既定のブラウザーに設定されていますが、ユーザーが別のブラウザーをインストールし、既定のブラウザーとすることも可能です (Windows 7 の場合と同じ)。Internet Explorer 10 では、デスクトップ アプリケーションからリンクを開くと、デスクトップに Internet Explorer が表示されます。ただしユーザーはこの設定を変更し、Internet Explorer を Windows ストア アプリとして開くことができます。ブラウザーのベンダーには、同様の "コンテキストに基づく起動" エクスペリエンスを採用するように推奨していますが、開発者は、すべてのブラウザーが同様に動作するわけではないことを想定しておく必要があります。

軽減策

- この動作を軽減するために、アプリに対して開発者ができることは特にありませんが、エンドユーザーによる対応策を伝える必要があります。これには、拡張された ShellExecuteEx() API によって収集された情報を利用し、コンテキストに応じて適切なダイアログを表示すると効果的です。このダイアログでは、どのアプリを起動するかと、そのアプリが Windows ストア アプリなのか、それともデスクトップ アプリなのかがユーザーに提示されます。Windows ストア アプリと

デスクトップ アプリを見分けるには、CLSID を使うことができます。ユーザーは以下の対応が可能です。

- 大画面のモニターを使っている場合は、Windows ストア アプリを画面の端にスナップしてデスクトップが見えるようにすることで、Windows ストア アプリとデスクトップ アプリの両方を同時に表示できます。
- Internet Explorer がユーザーの既定のブラウザーに設定されている場合は、[コントロール パネル] で [インターネット オプション] を変更することで、ブラウザーの動作を変更できます。[プログラム] タブでは、Internet Explorer によるリンクの取り扱い方法を変更できます。その他のブラウザーでも、同様の設定が可能な場合があります。

ハイ コントラスト モード

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

以前の Windows オペレーティング システムでは、ハイ コントラスト モードは視覚的なスタイルが設定されていないクラシック テーマの下で実行しているテーマに限定されていました。Windows 8 および Windows Server 2012 ではクラシック モードが削除され、視覚的なスタイルが設定されたハイ コントラスト テーマに置き換えられました。この変更の主な利点の 1 つは、クラシック モードで実行されているアプリごとに個別のコード パスを除去できることです。

開発者は、ハイ コントラスト モードがアプリに与える影響と、スタイルを選ばないアプリを開発する方法を学習する必要があります。テーマ カラーの誤った利用は Aero などの視覚スタイルでは正しく動作する可能性がありますが、同じアプリでもハイ コントラストの場合は誤った応答をするため、これらの理解が重要となります。たとえば Aero において、テキストは常に黒になりますし、ハイライト カラーは淡い青になります。しかし、ハイ コントラストの黒の場合は、ハイライト カラーも黒になります。開発者が Windows 8 より前の多くの付属アプリと同じように黒いテキストを想定し、ハイライトにシステムの既定を使う場合は、ユーザーには黒い背景の上に黒いテキストが表示されます。アプリ全体のスタイルが正しく見えるようにテーマやシステム メトリックスを正しく使用する方法を理解する必要があります。

影響

- アプリ マニフェストにおいて、Windows® 8 の <supportedOS> タグを含まないアプリのクライアント領域では、テーマ設定は許可されないため、アプリはクラシック テーマのハイ コントラスト モードで表示するのに必要なコード パスを使用して、クライアント領域を描画する必要があります。
- テーマ設定はハイ コントラスト テーマにおいてアプリの非クライアント領域とクライアント領域の両方で無効になります。また、それらのアプリ マニフェストにおいて Windows 8 <supportedOS> タグが含まれていない場合は、DwnIsCompositionEnabled () API を使用したウィンドウの非クライアント領域の描画は、アプリで無効になり、アプリ全体がクラシック テーマのハイ コントラスト モードで描画されます。
- Windows 8 のためのサポートをマニフェストに追加したアプリにおいて、描画用の視覚スタイルを使用しない場合や、アプリで色や画像をハードコーディングしている場合は、ハイ コントラスト テーマで正しく表示されない場合があります。ハイ コントラスト モードであったとしても、文字が読みにくいか画像が正しく表示されない場合があります。

軽減策

ハイ コントラスト テーマでテキストの色は、マイクロソフト アクセシビリティ ガイドラインに準拠するように作成されています。また、前景と背景のハイ コントラストの比率は 14:1 で維持しています。

既定で有効になっている色が特定のエンド ユーザーに適さない場合は、エンド ユーザーはコントロール パネルでハイ コントラスト テーマの [ウィンドウの色] の設定を簡単にカスタマイズすることができます。

ハイ コントラスト テーマの以下の UI コンポーネントをカスタマイズすることができます。

- ウィンドウの背景色
- テキストの色
- ハイパーリンクの色
- 非活性のテキスト
- 選択されたテキストの前景色と背景色
- アクティブなウィンドウ タイトルの前景色と背景色
- 非アクティブなウィンドウ タイトルの前景色と背景色
- ボタンの前景色と背景色

解決策

ハイ コントラスト テーマのアプリで予期しない動作が発生した場合は、以下のいずれかの解決策を試みてください。

- **Windows 8 用のアプリ マニフェスト:**
  Windows 8 の <supportedOS> タグをアプリ マニフェストに含まないアプリは、クライアント領域がテーマなしで描画されます。付属のアプリはアプリ マニフェストにこのエントリをすべて含める必要があります。以下の Windows 8 の GUID の値を追加してください。{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}
- **Owner-drawn UI の視覚スタイルの使用:**
  Owner-drawn コントロールは、テキストを含むコントロール パーツと状態を正しく描画するために、MSDN の指示に従ってください。開発者は、UxTheme 以外の方法で描画するためにデバイス コンテキストで指定したテキストや背景色に依存すべきではありません。問題のコントロールのためのテーマ パーツに関する記述がない場合は、GetThemeSysColor を適切なメトリックで使って、標準の GDI 関数によってテキストを描画します。UxTheme 呼び出しが適切でない場合は、適切なメトリックを取得するために GetSysColor 関数を使用します。
- **テキストの色の選択:**
  一般的なシナリオで正常に見えると仮定されていたとしても、ハードコードされた文字色を使用しないでください。出荷時のテーマは、関連するメトリックと高い可視性をサポートする方法で作成されています。たとえば、COLOR_HIGHLIGHTTEXT は背景として COLOR_HIGHLIGHT を使用し、COLOR_WINDOWTEXT は背景として COLOR_WINDOW を使用します。これらの関連に例外がある場合は、コード内ではなくテーマ パーツや状態の定義そのもので操作するようにします。ハイ コントラストの UI を設計する際には、ハイ コントラスト ユーザーが色をカスタマイズすることができるように UI が現在適用されているハイ コントラスト テーマにとらわれないことが重要です。

- **WM_ThemeChange イベントへの応答:**
  アプリがテーマから取得した色をキャッシュするか、標準ではない色を適用する場合は、保存された色の値を再計算して UI を再描画する WM_THEMECHANGE のメッセージ ハンドラーを追加します。
- **ハイ コントラスト WWA アプリの記述:**
  Web アプリは、UxTheme API へのアクセスを持ちませんが、UI の基礎として現在のシステム メトリックスで記述されなければいけません。WWA 開発者のために、アプリのハイ コントラスト準拠を確実にするために活用できるいくつかのリソースがあります。
  - W3C CSS Color 仕様は、システム メトリックスの代わりに特定の色を使うための構文を指定します。
  - Internet Explorer 10 には、ハイ コントラスト メディア クエリのサポートが追加されています。
  - WWA は IAccessibilityCapabilities::get_HighContrast() メソッドを使って、ハイ コントラストの状態を確認できます。

  Windows ストア アプリでは、従来のアプリが持つテーマ パーツと同じ問題の多くを持ってはいませんが、ハイ コントラストのコンプライアンスを確認する必要があります。既定では Internet Explorer は、特定のユーザー定義のスタイルを無視し、ハイ コントラストに準拠した値に置き換えます。たとえば、**background-image**、**background**、**color** の CSS プロパティは無視されます。

  Internet Explorer にプロパティの設定を無視させたくない場合で、UI がハイ コントラストに準拠していることを確認した場合は、親要素で新しい M3 CSS のプロパティである **-ms-high-contrast: off** を設定することができます。
- **ハイ コントラスト Windows ストア アプリの記述:**
  Windows ストア アプリは、特定のシステム メトリックスの色は組み合わせて使うように設計されていることを念頭において (SystemColors.WindowColor と SystemColors.WindowTextColor など)、適切な UI 要素の色を決定するために SystemColors クラスを使う必要があります。これによって優れたハイ コントラスト エクスペリエンスが促進されます。
- **以前のバージョンの Windows でのハイ コントラストの正しい検出:**
  以前のバージョンの Windows で実行されているアプリは、マニフェストに Windows のバージョンとの互換性の問題を明記していたとしても、新しいハイ コントラスト テーマにアクセスすることはできません。そのため、以前のバージョンの Windows で使用されるクラシック環境でのレンダリングを処理するには、追加のコード パスを挿入する必要があるかもしれません。このケースにおいては、SPI_GETHIGHCONTRAST フラグを使って SystemParametersInfo 関数を呼び出すことでハイ コントラストの存在をチェックする必要があります。これはハイ コントラストの存在をチェックするためのサポートされている唯一の方法となります。

テスト

アプリのテストにおいて、Windows 8 で提供されているすべての付属のテーマである Aero、ベーシック、ハイ コントラスト 1、ハイ コントラスト 2、ハイ コントラスト黒、ハイ コントラスト白で正しく描画されていることを確認してください。テキストはハイ コントラストのテーマにおいてはっきりと見え、読み取りやすいことを確認してください。

参考資料

Aero スタイルのクラス、パーツ、状態 (新しいベーシック テーマとハイ コントラスト テーマもこれらの状態を使用)
http://go.microsoft.com/fwlink/p/?linkid=325433

パーツと状態で共通するすべてのビジュアル スタイル
http://go.microsoft.com/fwlink/p/?linkid=325434

カスタムのビジュアル スタイルと Owner-Drawn コントロールを使用する
http://go.microsoft.com/fwlink/p/?linkid=325435

GetSysColor 関数
http://go.microsoft.com/fwlink/p/?linkid=325436

W3C CSS Color Module Level 3
http://go.microsoft.com/fwlink/p/?linkid=203761

SystemColors クラス
http://go.microsoft.com/fwlink/p/?linkid=325437

SystemParametersInfo 関数
http://go.microsoft.com/fwlink/p/?LinkId=325438

マイクロソフト アクセシビリティ
http://go.microsoft.com/fwlink/p/?linkid=202480

アプリ (実行可能ファイル) マニフェスト

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows に導入された、アプリ (実行可能ファイル) マニフェストの Compatibility セクションは、オペレーティング システムが、アプリ設計時に想定した Windows のバージョンを判断するために使われます。さらに、アプリ マニフェストを使用すると、そのアプリが想定する Windows のバージョンに基づいて、アプリの期待するモードで Windows が動作するようになります。

マニフェストの Compatibility セクションにより、Windows は新たに作成されたソフトウェアに新しい動作を提供する一方で、既存のソフトウェアとの互換性を維持できます。また、このセクションは、今後リリースされる Windows の互換性向上にも役立ちます。たとえば、Compatibility セクションで、アプリが Windows® 8 のみをサポートするように宣言している場合、今後リリースされる Windows では引き続き、Windows® 8 モードの動作が提供されます。

影響

アプリ マニフェストに Compatibility セクションがない場合、Windows 7、Windows® 8、および今後リリースされる Windows は、既定で Windows Vista モードの動作を提供します。Windows XP と Windows Vista はマニフェストのこのセクションを無視するため、影響を受けない点に注意してください。

次に示す Windows コンポーネントは、Compatibility セクションに基づいてさまざまな動作をします。

リモート プロシージャ コール (RPC) の既定のスレッド プール

- Windows 8 および Windows 7: スケーラビリティ向上とスレッド数の削減のため、RPC に NT スレッド プール (既定のプール) を使用するように変更されました。Windows Vista では、RPC にプライベート スレッド プールを使用していました。
    - Windows 7 以降のバージョンの Windows 用にコンパイルされたバイナリには、既定のプールを使用します。
    - いずれかの RPC API を呼び出す前に I_RpcMgmtEnableDedicatedThreadPool が呼び出されている場合には、プライベート スレッド プールを使用します (Windows Vista の動作)。
    - RPC の呼び出しの後に I_RpcMgmtEnableDedicatedThreadPool が呼び出された場合は、既定のプールを使用します。I_RpcMgmtEnableDedicatedThreadPool がエラー 1764 を返し、要求された処理は実行されません。
- Windows Vista: Windows Vista 以前のバージョンの Windows 用にコンパイルされたバイナリには、プライベート プールを使用します。

DirectDraw のロック

- Windows® 8 および Windows 7: Windows 7 以降のバージョンのオペレーティング システムをサポートするようにマニフェストで宣言しているアプリは、DDRAW の Lock API を呼び出して、プライマリ デスクトップ ビデオ バッファーをロックすることができません。この操作を行うと、エラーが発生し、プライマリ ビデオ バッファーへの NULL ポインターが返されます。
  この動作は、デスクトップ ウィンドウ マネージャーによるコンポジションが有効化されていない場合でも常に適用されます。Windows 7 以降との互換性を宣言したアプリでは、プライマリ ビデオ バッファーをロックしてレンダリングを実行することはできません。
- Windows Vista (既定): アプリがプライマリ ビデオ バッファーをロックできます (レガシ アプリはこの動作に依存しているため)。対象のアプリを実行すると、デスクトップ ウィンドウ マネージャーが無効化されます。

クリッピング ウィンドウなしでの DirectDraw によるプライマリ ビデオ バッファーへのビットブロック転送 (bitblt)

- Windows 8 および Windows 7: Windows 7 以降のバージョンの Windows をサポートするようにマニフェストで宣言しているアプリは、クリッピング ウィンドウなしでプライマリ デスクトップ ビデオ バッファーへの bitblt を実行することはできません。この操作を行うと、エラーが発生し、bitblt 領域はレンダリングされません。
  この動作は、デスクトップ ウィンドウ マネージャーによるコンポジションが有効化されていない場合でも、Windows によって常に適用されます。Windows 7 以降との互換性を宣言したアプリでは、クリッピング ウィンドウに対して bitblt を実行する必要があります。
- Windows Vista (既定): アプリは、クリッピング ウィンドウなしでもプライマリ ビデオ バッファーに bitblt を実行できる必要があります (レガシ アプリはこの動作に依存しているため)。対象のアプリを実行すると、デスクトップ ウィンドウ マネージャーが無効化されます。

GetOverlappedResult API

- Windows 8 および Windows 7: **GetOverlappedResult** を使用するマルチスレッド アプリから制御が戻る際、OVERLAPPED 構造体内のイベントがリセットされないことがあるために、次回この関数を呼び出すと、競合状態となって呼び出し途中で制御が戻る問題を解決します。
- Windows Vista (既定): 上記の競合状態での動作を示し、アプリが影響を受ける可能性があります。Windows 7 での動作が発生する前にこの競合状態を回避する必要があるアプリでは、オーバーラップしたイベントを待機し、シグナルがオンになってから、bWait == FALSE を指定して GetOverlappedResult を呼び出します。

ハイ コントラスト モードでのシェル テーマのステータス

- Windows 8: ハイ コントラスト モードでは、実際のテーマ設定状況を返します。
- Windows 7: ハイ コントラスト モードでは DWM が有効なため、テーマ設定は利用できないことを通知します。
- Windows Vista (既定): ハイ コントラスト モードでは DWM が有効なため、テーマ設定は利用できないことを通知します。

シェルの IPersistFile::Save メソッド

- Windows 8: CShellLink::Save が IPersistFile ハンドラーの呼び出しに相対パス引数が使用されているかどうかを確認し、該当する場合は呼び出しに失敗するようになりました。
  この動作について記載した公開ドキュメント (http://go.microsoft.com/fwlink/?LinkId=228369、英語) では、パス引数には絶対パスを使用する必要があると説明しています。

- Windows 7 以前 (既定): CShellLink::Save は、iPersistFile ハンドラーが相対パス チェックを送信するかどうかを確認せず、絶対パスか相対パスかに関係なく、引き続きアプリの動作を許可します。

プログラム互換性アシスタント (PCA)

- Windows 8: Compatibility セクションを使うアプリには、PCA による軽減策が適用されません。
- Windows 7: Compatibility セクションを使うアプリについては、Windows 8 での変更点 (このドキュメントで説明) に関する潜在的な互換性問題が追跡されます。
- Windows Vista (既定): 正常にインストールできないアプリや、ある特定の状況で実行するとクラッシュするアプリには、PCA による軽減策が適用されます。 詳しくは、参考資料のセクションをご覧ください。

重要機能の活用

アプリ マニフェストに、オペレーティング システムのサポートに関する最新の互換性情報を反映します。ここでは、マニフェストへの追加事項を示します。

名前空間: Compatibility.v1 (xmlns="urn:schemas-microsoft-com:compatibility.v1">)

セクション名: Compatibility (新セクション)

SupportedOS: サポート対象のオペレーティング システムの GUID。サポート対象の各オペレーティング システムに対応する GUID は次のとおりです。

- {e2011457-1546-43c5-a5fe-008deee3d3f0}
  Windows Vista に対応: スイッチバック コンテキストの既定値です。
- {35138b9a-5d96-4fbd-8e2d-a2440225f93a}
  Windows 7 に対応: アプリ マニフェストにこの値が設定されているアプリには、Windows 7 モードの動作が提供されます。
- {4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}
  Windows 8 に対応: アプリ マニフェストにこの値が設定されているアプリには、Windows 8 モードでの動作が提供されます。

今後リリースされる Windows の GUID については、必要に応じてマイクロソフトが生成します。

次に、マニフェストを更新するための XML の記述例を示します。

注: アプリ マニフェストの属性名およびタグ名では、大文字と小文字が区別されます。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
       <application>
              <!--The ID below indicates app support for Windows Vista -->
              <supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"/>
              <!--The ID below indicates app support for Windows 7 -->
              <supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/>
              <!--The ID below indicates app support for Windows Developer Preview
-->
              <supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>
       </application>
</compatibility>
</assembly>
```

上記の例でオペレーティング システムの GUID をすべて列挙しているのは、下位互換性があることを表しています。複数のプラットフォームをサポートするアプリで、プラットフォームごとにマニフェストを分ける必要はありません。

テスト

アプリでは、サポート対象のオペレーティング システム ID を複数指定できます。オペレーティング システム上でアプリをテスト済みの場合、またはテスト中の場合は、サポート対象のオペレーティング システム ID を追加する必要があります。Windows Vista 以前のオペレーティング システムは、こうしたエントリを考慮しません。Windows 7 以降の Windows では、マニフェスト内の最も新しいバージョンの GUID から、実行中の Windows のバージョンまでを選択し、そのレベルでアプリをサポートします。

新しいアプリ マニフェストの Compatibility セクションを適用し、アプリの動作を確認します。

1. 新しい Compatibility セクションを適用し、SupportedOS ID = { 4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38} として、最新の Windows 8 モードでアプリが適切に動作することを確認します。
2. 新しい Compatibility セクションを適用し、SupportedOS ID = {35138b9a-5d96-4fbd-8e2d-a2440225f93a} として、Windows 7 モードでアプリが適切に動作することを確認します。
3. 新しい Compatibility セクションを適用し、SupportedOS ID = {e2011457-1546-43c5-a5fe-008deee3d3f0} として、Windows Vista モードでアプリが適切に動作することを確認します。

参考資料

QueryActCtxW 関数
http://go.microsoft.com/fwlink/p/?linkid=325439

UAC マニフェスト
http://go.microsoft.com/fwlink/p/?linkid=228372

Windows アプリケーションのアプリ マニフェスト
http://go.microsoft.com/fwlink/p/?linkid=325440

デスクトップ ウィンドウ マネージャー (DWM)
http://go.microsoft.com/fwlink/p/?linkid=325441

コンテキストの不一致の更新
http://go.microsoft.com/fwlink/?LinkId=205035

プログラム互換性アシスタント
http://go.microsoft.com/fwlink/p/?linkid=228376

キューを使った表示モデルの廃止

プラットフォーム

クライアント – Windows 8 より後

サーバー – Windows Server 2012 より後

説明

Windows 8 より後の Windows リリースでは、以下の API によって E_NOTIMPL が返されます。

- DwmSetPresentParameters
- DwmSetDxFrameDuration
- DwmModifyPreviousDxFrameDuration

さらに、次の API は、hwnd パラメーターの値として null 値のみしか受け入れなくなります。

- DwmGetCompositionTimingInfo

DwmGetCompositionTimingInfo での hwnd != NULL の使用はサポートされなくなり、関連する機能も廃止されます。hwnd に NULL 以外の値を指定すると、この API は E_INVALIDARG を返します。

また、開発者に対しては、DwmGetCompositionTimingInfo API を使用する代わりに、DXGI フリップモデルおよび関連する DX 表示統計 API を使用することをお勧めします。

影響

キューを使用した表示モデルを採用しているアプリケーションは、正しく動作しなくなります。厳密な影響はアプリケーションごとに異なりますが、表示タイミングの異常や、予期せぬアプリケーションの終了などが考えられます。実際には、影響を受けるアプリケーションがあったとしても、その数は多くないと思われます。このモデルは Vista のメディア プレーヤーで使わていましたが (古い Zune プレーヤーでも使用されている可能性があります)、Windows 9 では廃止されます。現在のところ、それ以外のアプリケーションで、実際にこのモデルが使われているという情報はありません。

解決策

開発者は、キューを使用した表示方法 (Windows 7 以降での DX9 ランタイム、および Windows 8 での DX10 ランタイムと DX11 ランタイムで利用可能) の代わりに、DXGI フリップ表示モードを使用する必要があります。

テスト

一般的なテストを実施し、既存の Windows コンポーネントおよび主要製品が、Windows の次期バージョンでも引き続き動作することを確認してください。

Windows 8 のプログラム互換性アシスタント シナリオ

プラットフォーム

クライアント – Windows XP | Windows Vista | Windows 7 | Windows 8

説明

Windows 8 のプログラム互換性アシスタント (PCA) は、旧バージョンの Windows 用に設計されたデスクトップ アプリをエンド ユーザーが実行できるようにするための機能です。Windows 8 は優れたアプリ互換機能を備えており、Windows 7 またはそれ以前のバージョンの Windows 用に設計されたアプリを、自動的に Windows 8 上で正しく動作させることができます。ただし、ごく一部のアプリでは、実行時のトラブルを避けるために対策が必要になる場合があります。

ユーザーがアプリを実行すると、PCA はそのアプリを追跡し、Windows 8 で発生する特定の既知の互換性問題について、症状の有無を確認します。問題の症状を検出した場合、PCA は Windows 8 でのアプリの動作改善に役立つ修正プログラムを示し、ユーザーがそれを適用できるようにします。

シナリオ

PCA は、Windows 8 で発生する一連の既知の互換性問題についてアプリを追跡します。PCA はこうした問題を追跡し、修正プログラムを特定した後、ダイアログを表示して、推奨する修正プログラムを適用するように促します。提示された修正プログラムを適用するかどうかはユーザーが判断でき、何もしないことを選択した場合は、推奨策をキャンセルできます。推奨策をキャンセルすると、PCA は対象のアプリの追跡を終了します。

PCA は通常、プログラム (またはそのセットアップ) が実行されたタイミングに応じて、3 つある Windows 互換モードのうちの 1 つ (Windows XP SP3、Windows Vista SP2、または Windows 7) を適用します。PCA は、対象プログラムの LINK_DATE 属性および SUBSYSTEM VERSION 属性と、実行可能ファイルのマニフェストの TRUSTINFO セクションおよび COMPATIBILITY セクションを使用して、どのモードが適切かを判断し、Windows XP SP3 (管理特権を含む)、Windows Vista SP2、または Windows 7 を個々に適用します。PCA が適用する各互換モードおよびその説明については、このセクションの末尾にある用語集を参照してください。

以下に挙げるシナリオではすべて、修正プログラムの適用後に、PCA によって改めてアプリが追跡されます。互換性修正プログラムを適用した後も、引き続きアプリで同様の障害が発生する場合、PCA は適用した修正プログラムを元に戻します。以降、障害が発生したその特定のアプリを、PCA は二度と追跡しません。

PCA は多くの潜在的な問題を追跡しますが、こうした問題のすべてが実際にアプリ障害を引き起こすわけではありません。PCA は、アプリの障害の原因が Windows の互換性である可能性が高い場合にのみ、修正プログラムを提示します。以下の各セクションでは、Windows 8 で発生する、PCA が関与する各種のシナリオについて順番に解説します。セクションごとに問題の発生シナリオを説明すると共に、Windows 8 で引き続きアプリを適切に動作させるために、PCA が提示する推奨策を示します。

Windows 8 での互換性に関する変更点の詳細については、Windows 8 互換性ガイドブックのその他のトピックを参照してください。

PCA が追跡し、修正プログラムを提案するシナリオは次のとおりです。

- アプリのインストールまたはアンインストールに失敗する
- Windows のバージョン チェック メッセージが表示され、アプリの実行に失敗する
- 管理特権が原因でアプリの起動に失敗する
- メモリに関する特定の問題が原因でアプリがクラッシュする
- システム ファイルの不一致が原因でアプリに障害が発生する
- 64 ビット Windows で、未処理のエラーがあるためにアプリに障害が発生する
- 保護された非 Windows ファイルを削除しようとするとアプリに障害が発生する
- Windows ファイルを修正しようとするとアプリに障害が発生する
- 8 ビットまたは 16 ビットのカラー モードの使用が原因でアプリに障害が発生する
- グラフィックスとディスプレイの問題が原因でアプリに障害が発生する
- DPI への対応がアプリで宣言されていない
- Windows の機能が見つからないことが原因でアプリに障害が発生する
- 64 ビット Windows で、署名されていないドライバーがあるためにアプリに障害が発生する
- 互換設定を通じてインストールされたアプリを追跡する
- アプリがインストーラーや更新ツールの起動に失敗する
- アプリのインストーラーの実行に管理特権が必要になる
- 以前のコントロール パネル アプレットの実行に管理特権が必要になる

上記の各シナリオについて、以下で詳しく説明します。

アプリのインストールまたはアンインストールに失敗する

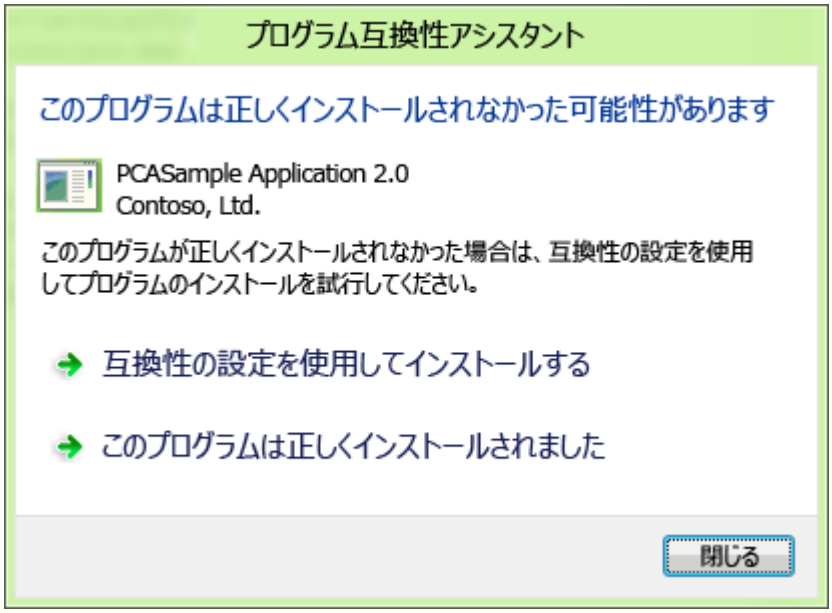
これは、アプリのインストール中に発生する、最もよくある種類のアプリ障害の 1 つです。以前作成されたセットアップ プログラムで最も多いのは、次の 2 つの障害です。

- Windows 8 のユーザー アカウント制御 (UAC) 機能が認識されないために、フル特権を使用せずにセットアップ プログラムが実行され、Windows 8 の保護された領域に対し、システムの変更を実行できない場合がある
- セットアップ プログラムが Windows のバージョンを確認し、想定よりも新しいバージョンを検出した場合に、セットアップの実行が中止される

上記の 2 つの不具合は、セットアップ時に発生する互換性障害のうち、最も一般的な部類に当たります。PCA は、UAC などの他のさまざまな Windows コンポーネントと連携しながら、セットアップ プログラムの起動を検知し、インストールが完了するまで追跡を続けます。セットアップ プログラム

がファイルの追加に失敗した場合や、Windows コントロール パネルの "プログラムの追加と削除" の部分に有効なエントリを追加できなかった場合、PCA はセットアップが失敗したと見なします。

このとき、PCA はアプリに適した互換モードを提案します。この互換モードを使用すると、本来の設計対象だった Windows のモードでセットアップ プログラムを実行でき、確実に管理特権を使用してアプリが実行されます。PCA は、Windows XP、Windows Vista、または Windows 7 の互換モードと併せて、RUNASADMIN 互換モードを適用します。インストールに失敗すると、次のような PCA の推奨策を示したダイアログが表示されます。



ユーザーは次のいずれかを選択できます。

- 互換設定 (推奨オプション) を使用してプログラムを実行します。PCA は推奨設定 (互換モード) を適用し、セットアップ プログラムを再度起動して、セットアップが正常に完了するまで追跡します。
- プログラムが正しくインストールされたことを示すオプションをクリックします。この場合、PCA はそれ以上設定を追加せず、このセットアップの追跡を終了します。
- [閉じる] をクリックします。この場合、PCA はそれ以上設定を追加せず、このセットアップの追跡を終了します。

Windows の [プログラムの追加と削除] セクション、またはアプリのアンインストーラーへのショートカットを通じて、ユーザーがアプリをアンインストールしようとすると、アプリのアンインストールを支援するために、上記と同じメカニズムが使用されます。

Windows のバージョン チェック メッセージが表示され、アプリの実行に失敗する

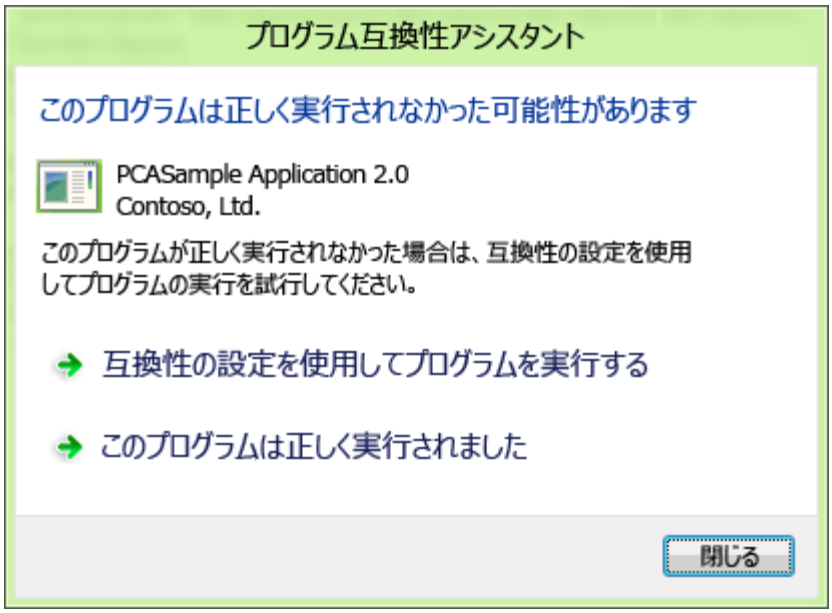
アプリの実行時に発生しやすい互換性障害の 1 つに、Windows のバージョン チェックによる障害があります。多くのアプリは、起動時に Windows のバージョンを確認します。このときにバージョンを認識できないと、たとえ動作に問題がなくても、アプリ自体が処理の続行を中止します。

一般にこうしたチェックは、PCA の追跡対象である次の 2 つの状態に関連しています。

- アプリによって、ユーザーへの警告を示すメッセージ ボックスが表示される。次に例を示します。

- アプリがすぐに終了するか、クラッシュする。

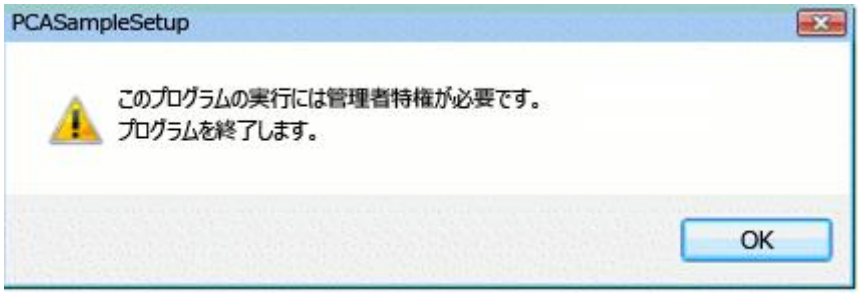
これらのいずれかの状態をアプリで検出すると、PCA はユーザーへの推奨策を提示します。ユーザーは互換設定を使用して、アプリを再実行できるようになります。PCA は対象のアプリに応じて、Windows XP、Windows Vista、Windows 7 のうち適切な互換モードを適用します。他のシナリオと同様、ユーザーはアプリが正常に動作していることを示すオプションをクリックするか、[閉じる] ボタンをクリックし、推奨設定の適用をキャンセルできます。以下は、ダイアログの一例です。



管理特権が原因でアプリの起動や実行に失敗する

アプリを動作させ、機能を実行するには、管理特権を必要とする場合があります。ただし Windows 8 では、Windows 7 や Windows Vista と同じく、UAC の影響により、既定では低い権限レベルでアプリを実行します。Windows Vista 以降のために設計された比較的新しいアプリでは、EXE のマニフェストの TRUSTINFO セクションを使用して、実行に必要となる特権レベルを宣言するのが一般的です。ただし、より古いアプリでは通常、次の 2 通りの障害が発生します。

- アプリによって、管理特権が必要であることを示すメッセージが表示される (下の例を参照)。



- アプリがすぐに終了するか、クラッシュする。

これらのいずれかの状態をアプリで検出すると、PCA はユーザーへの推奨策を提示します。ユーザーは管理特権を使用して、アプリを再実行できるようになります (PCA が RUNASHIGHEST 互換モードを適用します)。アプリを再実行すると、UAC のダイアログが表示されます。他のシナリオ同様、ユーザーは推奨設定を適用してアプリを再実行するか、[閉じる] ボタンをクリックし、推奨設定の適用をキャンセルできます。以下は、ダイアログの一例です。

メモリに関する特定の問題が原因でアプリがクラッシュする

アプリによっては、メモリに関する既知の問題が原因でクラッシュする場合があります。こうしたアプリはメモリから DLL を逆参照し、同じ DLL 内のコードを実行する関数を呼び出します。これを実行しようとすると、アプリは直ちにクラッシュします。この問題の原因は、Windows 8 の互換性の変更点ではありませんが、多岐にわたるアプリケーションで相対的によく発生する問題です。PCA はこの問題を追跡し、アプリの動作の信頼性向上に役立つ対処法をユーザーに提示します。

該当するアプリに対し、PCA はユーザーに通知することなく、自動的に PINDLL 互換モードを適用します。PCA がこの互換モードを呼び出すと、使用する DLL をアプリがメモリから消去できなくなります。そのため、アプリによる DLL への関数呼び出しが機能し、アプリのクラッシュを回避して、アプリの動作を適切に継続できるようになります。

システム ファイルの不一致が原因でアプリに障害が発生する

Windows XP 以前の OS 用に設計されたアプリの場合、インストーラーに Windows のシステム DLL のコピーが付属していることがあります。こうしたアプリをインストールすると、Windows のシステムフォルダー内にある最新バージョンの DLL に加え、このアプリ自体のフォルダーに、同じ DLL の古いバージョンが格納されます。

Windows Vista 以降ではこの状態が原因となり、アプリがローカルの DLL を読み込もうとして障害が発生することがあります。これはこのローカルの DLL が、その他の最新の Windows システム DLL とうまく連携しないためです。こうしたアプリは通常、より新しいバージョンの DLL があることを認識しないので、適切に動作しなくなります。

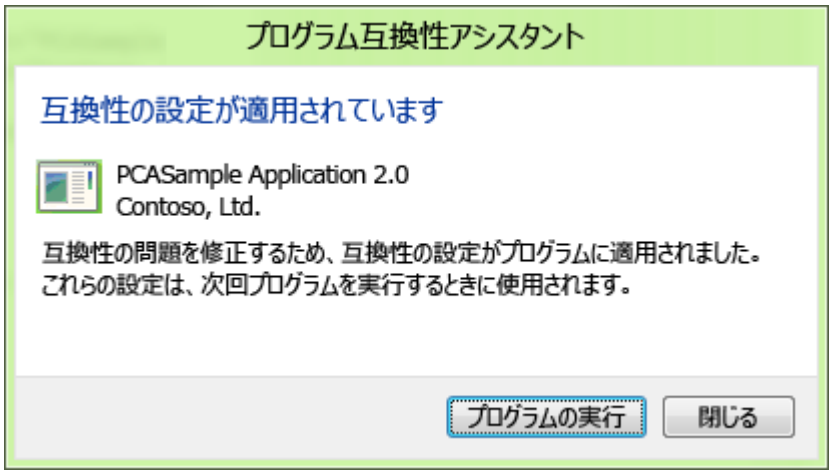
このように、DLL が正しく読み込まれていないことを検知すると、PCA は互換設定を適用し、Windows が Windows システム フォルダーから最新バージョンの DLL を読み込むようにして、アプリを正しく実行できるようにします。

初回にアプリの実行に失敗すると、次のような PCA のダイアログが表示され、互換設定を適用したことが通知されます。

64 ビット Windows で、未処理のエラーがあるためにアプリに障害が発生する

64 ビット バージョンの Windows 8 では、メッセージ ループのコールバック メカニズムに対する新しい例外が有効になりました。この例外が最初に導入されたのは Windows 7 ですが、このエラーの処理は必須ではありませんでした。Windows 8 では、アプリでメッセージ ループを使用する場合、この新しい例外を処理する必要があります。例外処理を行わないと、アプリはクラッシュします。以前のバージョンの Windows 用に設計されたアプリは、この例外を認識しない場合があるため、このエラー (例外) が適切に処理されない可能性があります。

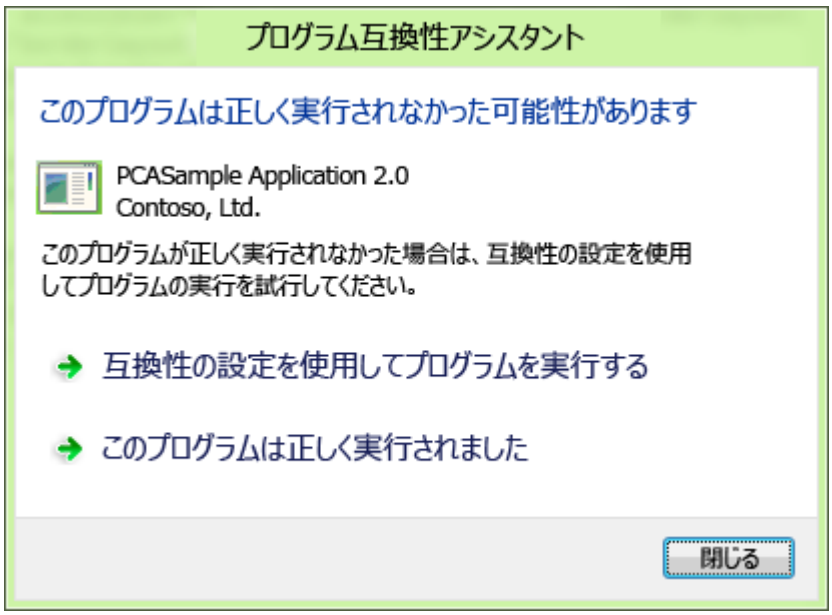
こうした未処理のエラーが原因で障害が発生したアプリを検知すると、PCA はそのアプリに対し、DISABLEUSERCALLBACKEXCEPTION 互換モードを自動的に適用します。この設定を適用してアプリの実行を終了する際、ユーザーには以下のメッセージが表示されます。アプリの次回起動時には互換モードが有効になり、このエラーは発生しなくなります。



保護された非 Windows ファイルを削除しようとするとアプリに障害が発生する

Windows XP 以前の OS 用に設計されたアプリの一部は、普段からフル管理者特権で実行されることが前提になっています。こうしたアプリは、通常動作の一環として、保護された非 Windows ファイル (プログラム ファイルまたは Windows のフォルダーに含まれる) を削除しようとする場合があります。この削除処理に失敗すると、該当する多くのアプリはクラッシュする可能性があります。

このように、保護されたファイルの削除に失敗し、クラッシュしたアプリを検出すると、PCA は推奨策をユーザーに提示します。ユーザーは互換設定を使用して、アプリを再実行できるようになります。他のシナリオと同様、ユーザーはアプリが正常に動作していることを示すオプションをクリックするか、[閉じる] ボタンをクリックし、推奨設定の適用をキャンセルできます。このシナリオの場合、PCA は VIRTUALIZEDELETE 互換モードを適用します。以下は、ダイアログの一例です。

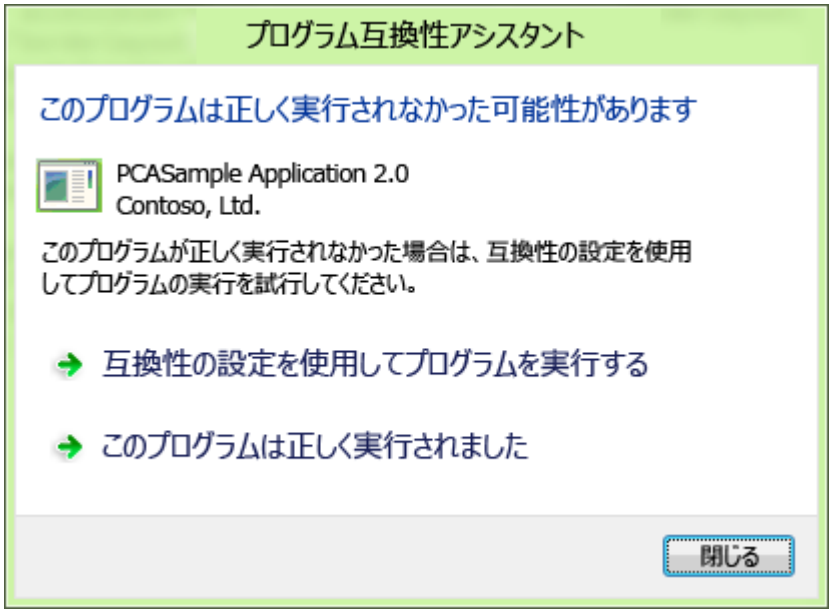


Windows ファイルやレジストリ キーを修正しようとするとアプリに障害が発生する

Windows XP 以前の OS 用に設計されたアプリの一部は、普段からフル管理特権で実行されることが前提になっています。こうしたアプリは通常動作の一環として、保護された Windows ファイル (プログラム ファイルまたは Windows のフォルダーに含まれる) や、Windows が所有するレジストリ キーに対する修正、削除、または書き込みを実行しようとする場合があります。ファイルまたはレ

ジストリ キーに対する書き込み、削除、修正のいずれかの処理に失敗すると、該当する多くのアプリはクラッシュするか、重大な障害を起こす可能性があります。

このように、保護された Windows ファイルまたはレジストリ キーへの書き込みに失敗したアプリを検出すると、そのアプリを終了する際に PCA はユーザーへの推奨策を提示します。ユーザーは互換設定を使用して、アプリを再実行できるようになります。他のシナリオと同様、ユーザーはアプリが正常に動作していることを示すオプションをクリックするか、[閉じる] ボタンをクリックし、推奨設定の適用をキャンセルできます。このシナリオの場合、PCA は WRPMITIGATION 互換モードを適用します。以下は、ダイアログの一例です。



8 ビットまたは 16 ビットのカラー モードの使用が原因でアプリに障害が発生する

Windows 8 を Windows ストア アプリに対応させるために行った主な変更点の 1 つとして、Windows 8 では、デスクトップ ウィンドウ マネージャー (DWM) が、32 ビット色のみをサポートするようになります。今後、それより低画質のカラー モードはシミュレートされます。

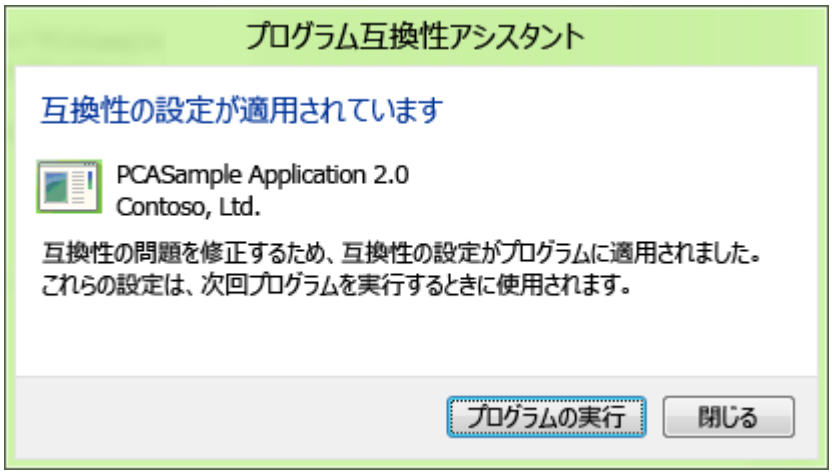
Windows XP 以前の OS 用に設計された多くの古いアプリやゲームは、8 ビットまたは 16 ビットのカラー モードを使用します。軽減策を講じない場合、こうしたアプリは Windows 8 上で実行できなくなる可能性があります。ただし、こうしたアプリが列挙処理を実行したり、8 ビットまたは 16 ビットのカラー モードを使用して表示を試みたりすると、PCA がこの問題を即座に特定し、DWM を利用して、シミュレートされたカラー モードでアプリが正しく動作するようにします。

この処理は、アプリから低画質のカラー モードが要求され次第すぐに発生し、ユーザーには通知されない点に注意してください。この軽減策を適用するために、ユーザーがアプリを再起動する必要はありません。アプリの正常な動作を確保するには、この修正プログラムが常に必要となるためです。

グラフィックスとディスプレイの問題が原因でアプリに障害が発生する

Windows 8 では常にデスクトップ ウィンドウ マネージャー (DWM) が有効化されるため、Windows XP 時代の古いアプリで障害が発生することがあります。これは以下に示すように、画面への描画に GDI と DirectX の両方の API を使用する場合など (ほとんどが古いゲーム)、アプリに混在モードのグラフィックス API が使用されていて、全画面モードを適用しようとする場合です。

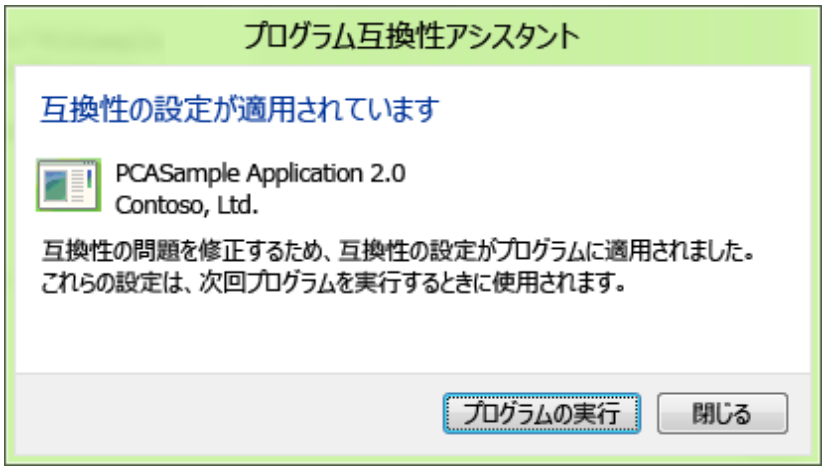
- DWM の影響で、デスクトップに直接描画することができないため、対象のゲームやアプリに障害が起こるか、デスクトップに黒い画面が描画され、グラフィックスがまったく表示されません。
- こうした状況では、対象のアプリが終了する際に、問題のアプリまたはゲームが全画面モードで不具合を起こすことを Windows が検知します。Windows は DXMAXIMIZEDWINDOWEDMODE 互換モードを適用し、対象のアプリやゲームを、全画面モードではなく最大化されたウィンドウで実行できるようにします。
- この設定を適用してアプリの実行を終了する際、ユーザーには以下のような PCA による通知画面が表示されます。アプリの次回起動時には互換モードが有効になり、アプリを正常に実行できるようになります。



DPI への対応がアプリで宣言されていない

多くの古いアプリでよく発生する、表示に関するもう 1 つの問題として、Windows およびアプリが高 DPI モードで実行されるにもかかわらず、EXE のマニフェストを通じて、アプリが高 DPI への対応を宣言していない場合が挙げられます。これは設定の不一致によって発生することがある一般的な問題で、UI 要素やテキストの一部が表示されない場合や、正しいフォント サイズが適用されない場合などがあります。こうした問題について詳しくは、こちらをご覧ください。

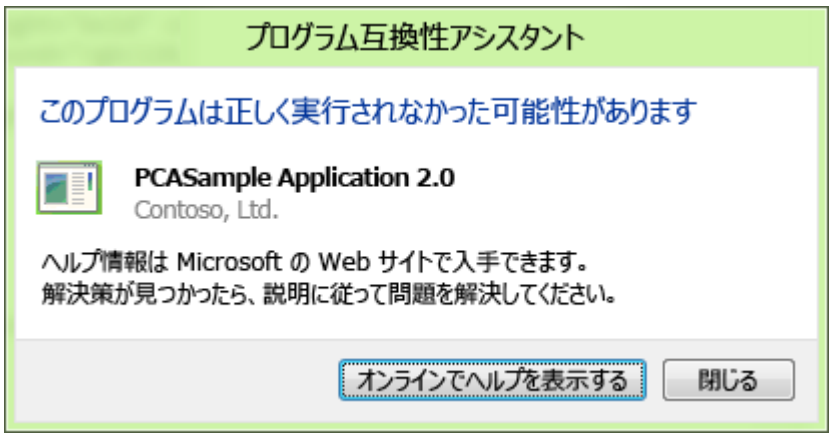
該当する状況では、アプリが高 DPI に対応していることを Windows が検知し、初回の実行を終了する際に HIGHDPIAWARE 互換モードをアプリに適用します。PCA はこの処置に関し、次のメッセージを表示してユーザーに通知します。



Windows の機能が見つからないことが原因でアプリに障害が発生する

アプリによっては、Windows Vista 以降に廃止された Windows の機能に依存していることがあります。こうしたアプリが、存在しない DLL や COM コンポーネントを読み込もうとすると、動作障害が発生します。

存在しない Windows の機能を読み込もうとするアプリを検知すると、PCA は推奨策として、アプリの終了後に必要なコンポーネントをダウンロードし、インストールするように促します。[オンラインでヘルプを表示する] をクリックすると、代替策を探すか、目的の機能をダウンロードしてインストールできます。必要に応じて、[閉じる] をクリックし、何も対応しないこともできます。

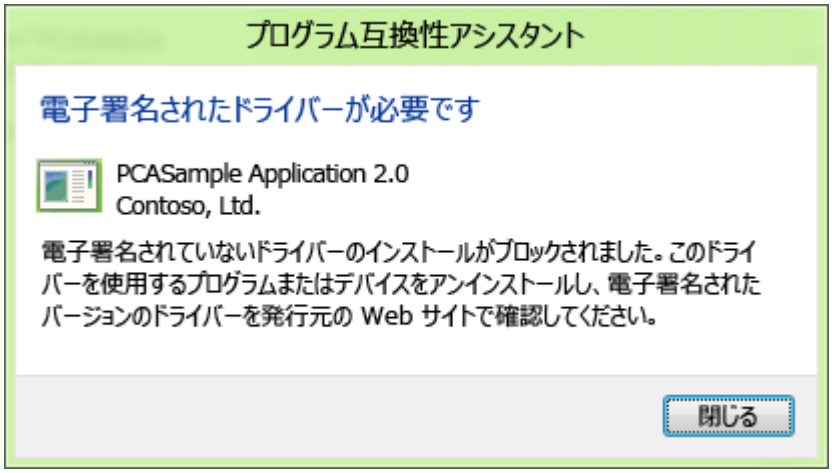


64 ビット Windows で、署名されていないドライバーがあるためにアプリに障害が発生する

Windows Vista 以降、64 ビット Windows では、デジタル署名されたドライバー (SYS ファイル) が必要になりました。ただし、Windows Vista よりも前のリリースの OS 用に設計された古いアプリには、デジタル署名されていないドライバーが付属しています。こうした未署名のドライバーをインストールしても、Windows は読み込みません。ごくまれなケースですが、該当するドライバーが起動時ドライバーとしてマークされている場合、Windows が起動しない可能性があります。

一部の古いアプリは、64 ビット Windows 上に未署名のドライバーをインストールします。このドライバーを読み込もうとすると、そのデバイスやアプリに障害が発生したり、システム クラッシュにつながることがあります。こうした状況を避けるために、未署名のドライバーをインストールしたアプリを検知すると、対象のドライバーが起動時ドライバーとしてマークされている場合、PCA はそのドライバーを無効化します。

PCA はさらに、アプリが正常に動作するよう、デジタル署名されたドライバーの入手を促すメッセージを表示します。このメッセージは、対象のドライバーのインストールと、関係するアプリのインストールに伴って表示されます。別のアプリが同じドライバーをインストールすると、そのアプリでも同じメッセージが表示されます。



互換設定を通じてインストールされたアプリを追跡する

インストーラーに障害が発生した場合、PCA はその障害の種類に応じ、さまざまな互換モードを使用してインストールを支援します。互換設定を使用してインストールに成功すると、PCA はそのインストーラーによって追加されたショートカットを追跡します。これにより、インストールされたアプリを追跡して、対応するインストーラーに適用したのと同じ互換設定が必要となる場合に備えます。

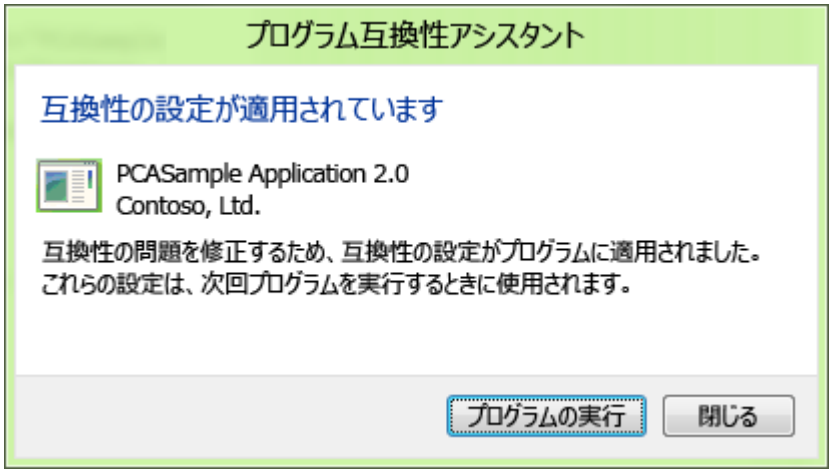
ユーザーが該当するアプリを起動すると、PCA はアプリが正しく動作しているかどうかを確認するために、ダイアログを表示します。ユーザーの答えが "はい" の場合、PCA はアプリの追跡を終了します。ユーザーの答えが "いいえ" の場合、PCA は対象のアプリのインストーラーと同じ互換モードを適用し、互換モードを適用した状態でアプリを再実行します。

アプリがインストーラーや更新ツールの起動に失敗する

アプリは、管理者として実行する必要のある子プログラムを起動することがあります。この典型的な例が、アプリが更新ツール ソフトウェアを起動して、アプリの新しい更新プログラムを確認し、インストールしようとする場合です。アプリがこうした子プログラムを直接実行すると、子プログラムの起動に失敗する可能性があります。これは、アプリ自体に管理特権がないためか、UAC マニフェストによって、特権の昇格に関して子プログラムが適切にマークされていないためです。

PCA はこうしたエラーを追跡し、元となるアプリを閉じる際に、自動的に ELEVATECREATEPROCESS 互換モードを適用して子プログラムが正しく動作するようにします。その後、対象のアプリを実行し、子アプリが起動されると、この子アプリに関する UAC ダイアログが表示されます。

以下は PCA のダイアログの一例です。



アプリのインストーラーの実行に管理特権が必要になる

Windows デスクトップ アプリのインストーラーは、保護されたシステム領域に対するファイル、フォルダー、レジストリ エントリの書き込みを実行するために管理特権を必要とします。Windows (UAC) はインストーラーの実行を特定する検出ロジックを備えており、UAC ダイアログを通じて、管理特権が必要であることを示すメッセージを直ちに表示します。ただし、一部のケースでは、アプリが確かにインストーラーであるかどうかをこのロジックが判断できず、管理特権を得られない場合があります。これらは通常、カスタム仕様のインストーラーであり、Windows インストーラーや Install Shield といった、よく知られているインストール テクノロジを使用していません。

PCA はこうした状況で、インストーラーによるファイルの書き込みが失敗したことを検知します。インストールに失敗した場合は、インストーラーを終了する際に、PCA が推奨策を提示し、互換設定の適用を促します。ユーザーが [プログラムの実行] をクリックすると、PCA は RUNASADMIN 互換モードを適用し、インストーラーを再実行します。[閉じる] をクリックすると、設定は適用されません。以下は PCA のダイアログの一例です。

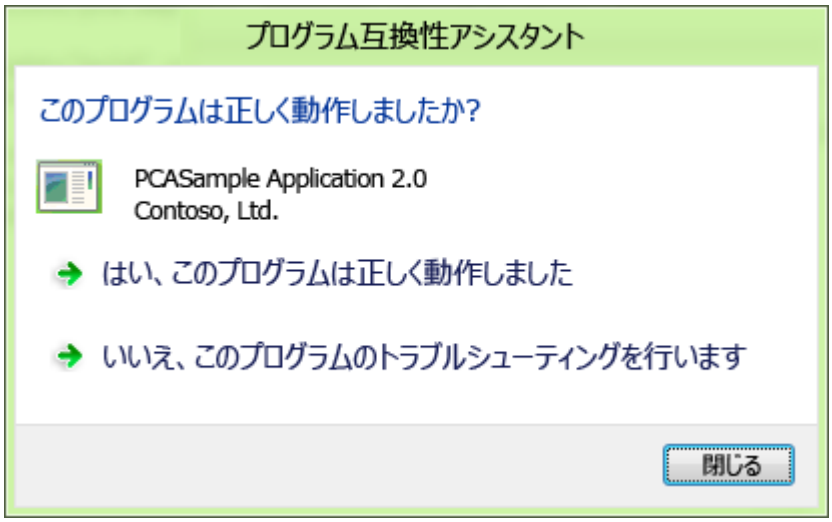
以前のコントロール パネル アプレットの実行に管理特権が必要になる

コントロール パネル アプレットは通常、システム設定を変更するために、管理者としての実行が可能でなくてはなりません。ただし、Windows Vista よりも前に作成されたコントロール パネル アプレットには、EXE のマニフェストがないか、必要な特権レベルを宣言する TRUSTINFO セクションがありません。

該当するアプレットの実行を検知すると、初回の実行を終了する際に、PCA が推奨策を提示し、管理者設定を適用して実行するように促します。ユーザーが [プログラムの実行] をクリックすると、PCA は RUNASADMIN 互換モードを適用し、インストーラーを再実行します。[閉じる] をクリックすると、設定は適用されません。以下は PCA のダイアログの一例です。

ユーザーのフィードバックを通じて推奨設定を検証する

各シナリオの最後に (推奨する互換設定でアプリを実行した後)、PCA はユーザーに簡単な質問をします。



ユーザーは、互換設定によってアプリが動作したか、それとも障害が起きたかをフィードバックできます。このデータは匿名でマイクロソフトに送信されます。この情報は、こうした修正プログラムを Windows の更新プロセスを通じて Windows 8 に組み込んでもよいかどうかを確認するために使用されます。これにより、今後 Windows 8 を使用するユーザーが同じアプリの障害を経験することがなくなり、PCA も同じアプリの同じ障害を追跡する必要がなくなります。

推奨策のない問題を追跡する

互換性が原因のアプリ障害は、さまざまな症状を示す可能性があります。PCA は、これまでに説明したシナリオ以外にも、多くの互換性問題を追跡しています。こうしたケースでは、多くの場合、問題の影響がアプリによって異なります。つまり、問題点を適切に処理し、回復するアプリもあれば、

そうでないアプリもあります。こうした問題について、PCA は引き続きアプリを追跡しますが、直接修正プログラムを提示することはありません。

PCA が問題を追跡していても、推奨設定やダイアログが表示されないアプリには、次のようなものが含まれます。

- 実行時間がきわめて短く、3 秒程度しかないアプリ
- 管理特権なしにグローバル メモリ オブジェクトを作成するアプリ
- 起動時にエラー (Win32 の例外) が発生するアプリ
- 管理特権をチェックするアプリ (障害が発生するとは限らない)
- Indeo コーデック (Windows Vista 以降廃止) を使用するアプリ
- HKLM など、保護されたレジストリの格納場所に対し、キーの書き込み/削除を行うアプリ
- 起動時にクラッシュするアプリ

[互換性] タブと互換性トラブルシューティング ツールから修正プログラムを適用する

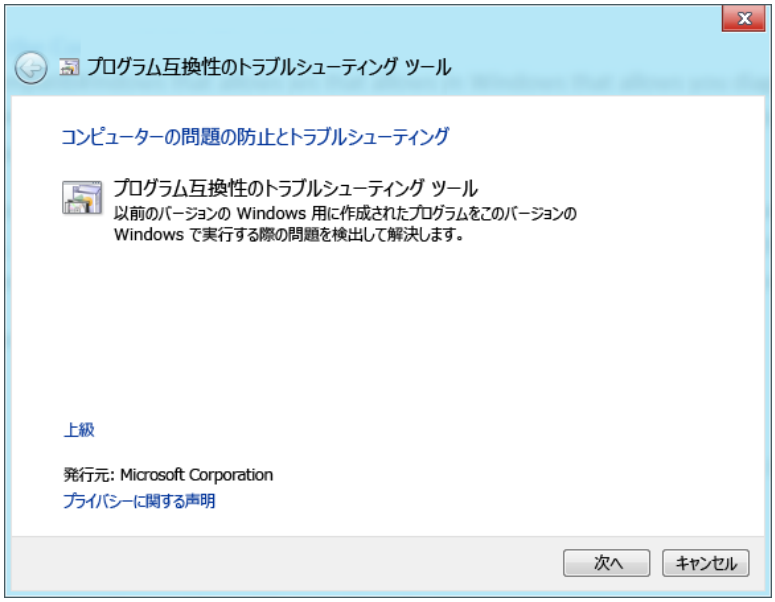
既に説明したように、アプリはさまざまな互換性問題が原因で障害を起こす可能性があります。アプリによって設定が異なるため、こうした障害のすべてについて PCA の明確な推奨策があるとは限りません。ただし、互換性トラブルシューティング ツールや [互換性] タブを使用すると、所定の共通の修正プログラムを適用し、障害を起こすアプリが Windows 8 上で正常に動作するかどうか試すことができます。この場合、修正プログラムを適用する前と後とを通じて、PCA は引き続き、互換性問題に関して対象のアプリを追跡します。修正プログラムを適用してアプリを実行した後、PCA は修正プログラムの効果を確認するメッセージを表示します。この質問にユーザーが答えると、遠隔測定機能によって、そのデータが匿名でマイクロソフトに送信されます。多くのユーザーから集めたこのデータを分析し、効果が確認された修正プログラムは、Windows Update を通じて Windows 8 の全ユーザーに広く配布されます。

互換性トラブルシューティング ツールを使用する

互換性トラブルシューティング ツールは、アプリに関する問題を診断し、推奨された修正プログラムを適用して、アプリが適切に動作するようにできる Windows のメカニズムです。このツールは、PCA がアプリに関する推奨策を何も提示しない場合にのみ必要となります。

このトラブルシューティング ツールを使用すると、ユーザーが一連の質問に順番に答えることで、その答えに基づいて各種の修正プログラムが適用されるため、ユーザーはアプリをテストし、修正プログラムの効果を確認できます。効果が確認されると、Windows 8 上でのアプリの動作を改善するために、対象の修正プログラムはそのままアプリに適用されます。

このトラブルシューティング ツールの UI を以下に示します。



互換性トラブルシューティング ツールは、次の 2 つの方法で起動できます。

- [スタート] 画面からの操作
  1. 「互換性トラブルシューティング ツール」と入力します。
  2. [設定] セクションの下に表示される [以前のバージョンの Windows 用に作成されたプログラムの実行] タイルをクリックします。
- アプリのタイルからの操作
  1. [スタート] 画面で、対象のアプリのタイルを右クリックします。
  2. [ファイルの場所を開く] をクリックします (デスクトップ アプリのみ)。
  3. エクスプローラーのリボン UI で、[アプリ] タブをクリックします。
  4. [互換性のトラブルシューティング] をクリックします。

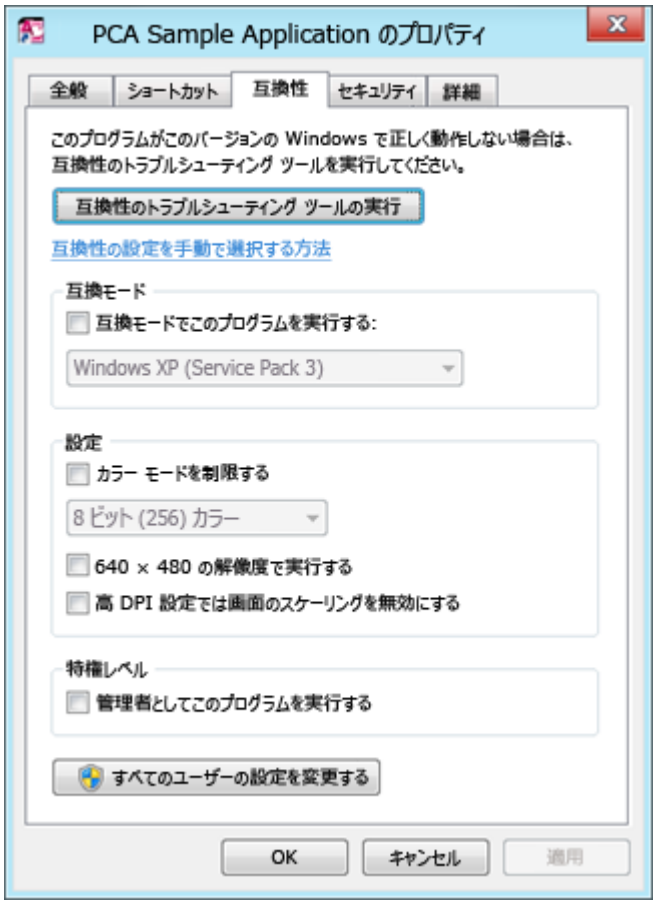
[互換性] タブを使用する

この方法は、さまざまな互換設定を試してみたことのあるユーザーのみにお勧めします。この方法では、アプリに適用する修正プログラムの種類について、一切の推奨策を示しません。以降の手順は、アプリを動作させるために、どの修正プログラムを適用すべきかについて、ユーザーが理解していることを前提としています。適用する修正プログラムに自信がない場合は、互換性トラブルシューティング ツールを使用して、アプリに適した修正プログラムを特定してください。

[互換性] タブへのアクセス方法

- [スタート] 画面からの操作
  1. 目的のアプリのタイルを右クリックします。
  2. ファイルの場所を開きます (デスクトップ アプリのみ)。

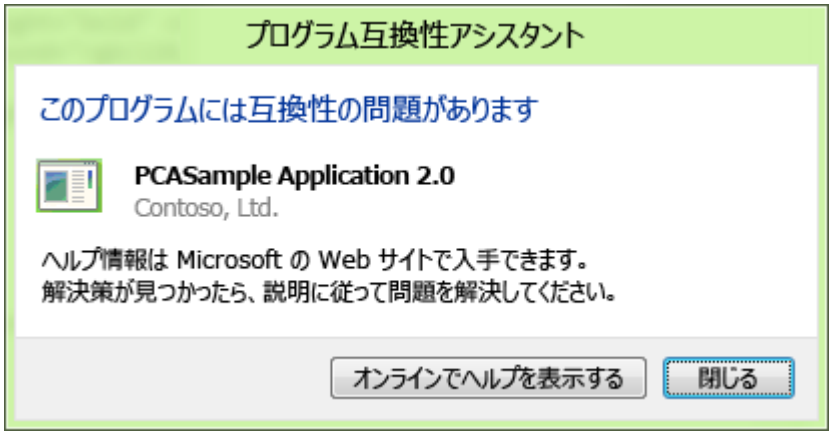
- エクスプローラーのリボン UI からの操作
  1. [プロパティ] をクリックします。
  2. [互換性] タブに移動します。
  3. 互換性修正プログラムを選択します。
  4. アプリを再実行します。
     注: 同じ場所に再度戻れば、修正プログラムの変更や削除も可能です。タブ内のボタンを使用して、使用するマシン上の全ユーザーに対して修正プログラムを適用することもできます。



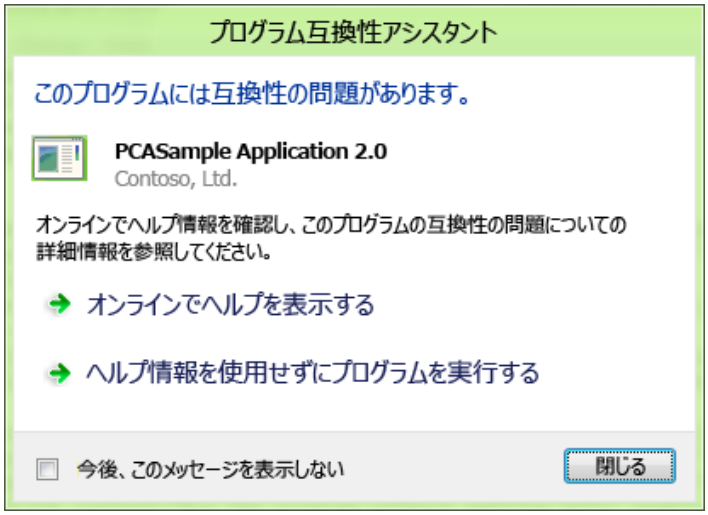
既知の互換性問題を伴うアプリ

これまで示したような実行時の問題の検知シナリオ以外にも、PCA はアプリのスタートアップ時に、アプリに既知の互換性問題があるかどうかを通知します。このリストは、システムのアプリ互換性データベースに格納されています。表示されるメッセージには、次の 2 種類があります。

- **ハード ブロック メッセージ** – 対象のアプリに互換性がないことが既にわかっており、アプリの実行を許可するとシステムに深刻な影響を与える場合 (たとえば、インストール後に Windows がクラッシュする場合や、起動できなくなる場合など) は、アプリをブロックしたことを示す次のようなメッセージが表示されます。



- **ソフト ブロック メッセージ** – 対象のアプリに既知の互換性問題があり、正しく動作しない可能性がある場合は、次のメッセージが表示されます。



いずれのケースでも、[オンラインでヘルプを表示する] オプションをクリックすると、Windows エラー報告が送信され、マイクロソフトからのオンライン回答がユーザーに表示されます。通常、この回答では、次の 3 種類のリソースのうちの 1 つがユーザーに示されます。

- アプリ ベンダーが提供する更新プログラム
- 詳細情報を確認できるアプリ ベンダーの Web サイト
- 詳細情報を確認できるマイクロソフト サポート技術情報の記事

PCA の利用統計情報

PCA は、Windows 8 マシン上のアプリの問題を解決し、ユーザーからのフィードバックがすべて得られた後、対象のアプリ、インストーラー、検知された問題点、アプリに適用した互換設定に関する匿名データを収集して、マイクロソフトに送り返します。このデータは、該当する匿名データの提供

に同意したすべてのユーザーから (カスタマー エクスペリエンス向上プログラム (CEIP) を通じて) 収集されます。このデータが収集されると、アプリ障害および修正プログラムについて分析した後、その修正プログラムが Windows Update メカニズムを通じて Windows エコシステム全体に配布され、今後同じアプリを使用するユーザーが、自動的にその修正プログラムを利用できるようになります。

管理者による制御と PCA 設定の管理

IT 管理者は、次の 2 つの方法で PCA の動作を制御できます。

- **PCA の無効化** – この設定により、IT 管理者は、ユーザーに対して PCA が表示するダイアログを無効化できます。PCA は引き続き問題を追跡および検知して、遠隔測定データを返送します。
- **アプリの遠隔測定の無効化** – この設定により、PCA による遠隔測定データの収集と送信がすべて無効化されます。
  CEIP が無効化されている場合、この設定は何の影響も及ぼしません。

PCA と連携するアプリを設計する

開発者は、既に説明した互換性シナリオのすべてについて、問題なく動作するアプリを確実に作成する必要があります。必ず前述の各シナリオについてアプリのテストと検証を実施し、互換性問題が発生しないようにしてください。互換性問題が見つかった場合は、そのアプリのための修正プログラムを作成し、互換性問題を確実に解決する必要があります。以下に、開発者が作成する必要のある一般的な修正プログラムの例を示します。

- インストール時および実行時の Windows オペレーティング システムのバージョン チェックを取り消す
- 特権チェック (管理者としてのアクセス権のチェック) を取り消す。EXE のマニフェストを使用して、必要となる適切なレベルの特権を宣言する
- アプリ インストーラーに、Windows のバイナリ ファイルが含まれていないことを確認する
- 保護された領域 (レジストリ、フォルダー) への書き込み、または保護されたファイルの上書きを取り消す
- すべてのバイナリ ファイル (EXE、DLL、SYS) にデジタル署名する
- インストーラーについて、"プログラムの追加と削除" に適切なエントリが追加されることを確認する。
  このアプリのメタデータ エントリは少なくとも、アプリ名、発行元、バージョン文字列、およびサポートされる言語を含んでいる必要があります。このエントリは、PCA に対してインストーラーが正常に実行されたことを示すと共に、ユーザーがアプリを簡単にアンインストールする方法を提供します。

Windows 8 の互換性ガイドブックの説明に従い、アプリ (実行可能ファイル) のマニフェストの TRUSTINFO セクションと COMPATIBILITY セクションを更新すれば、PCA はアプリが Windows 8 用に設計されていることを認識し、いずれの互換モードも適用せずに、常にネイティブの状態でアプリが実行されるようになります。

アプリが Windows 8 用に設計されていることを PCA に認識させるには、次の要件を満たす必要があります。

- すべての EXE ファイル (インストーラーまたはランタイム) について、マニフェストの TRUSTINFO セクションと COMPATIBILITY セクションで Windows 8 への対応を宣言する
- インストーラーが "プログラムの追加と削除" に確実にエントリを追加する

用語集

以下に、PCA が使用する互換モード一覧と、各モードの効果に関する簡単な説明を示します。

モード	説明
Windows7RTM	このモードは、オペレーティング システム バージョン番号 6.1 を含む、一般的な Windows 7 の動作をエミュレートします。
WindowsVistaSP2	このモードは、オペレーティング システム バージョン番号 6.0 を含む、一般的な Windows Vista SP2 の動作をエミュレートします。
WindowsXPSp3	このモードは、オペレーティング システム バージョン番号 5.1 を含む、一般的な Windows XP SP3 の動作をエミュレートします。 このモードには、RUNASHIGHEST モードも含まれます。
RUNASHIGHEST	このモードは、ユーザーにできるだけ上位の特権を使用してアプリを実行するように求めます。管理特権を持つユーザーには、対象のアプリに関する、UAC の昇格時のプロンプトが表示されます。
RUNASADMIN	このモードは常に、ユーザーに管理特権を使用してアプリを実行するように求めます。このモードを適用したアプリでは、必ず UAC の昇格時のプロンプトが表示されます。
ELEVATECREATEPROCESS	このモードは、管理特権で実行されている親アプリの子プロセスを作成します。子プロセスでは、UAC の昇格時のダイアログが表示されます。
PINDLL	このモードを適用すると、DLL をアプリがアンロードした場合でも、その DLL がメモリ内で保持されます。
DISABLEUSERCALLBACKEXCEPTION	このモードは、ユーザーによるコールバックの例外をインターセプトし、例外を処理することなくアプリの実行を継続できるようにします。
VIRTUALIZEDELETE	このモードは、保護されたファイルの削除処理をインターセプトし、削除処理の未処理の例外によって、アプリ障害が発生するのを防ぎます。
WRPMITIGATION	このモードは、保護された Windows ファイルやレジストリ エントリに対する書き込み、修正、削除をアプリが試行した場合に、(実際には処理が完了していなくても) 処理の成功を返します。
DXMAXIMIZEDWINDOWEDMODE	このモードは、全画面表示モードで実行されるアプリを特定し、最大化されたウィンドウで実行されるようにリダイレクトします。
HIGHDPIAWARE	このモードは、Windows の他の部分に対してアプリが高 DPI 対応であることを知らせ、UI 要素、テキスト、フォントなどの適切なレンダリングを支援します。

デスクトップ ガジェットの削除

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

機能的に新しいライブ タイルやアプリの方が優ることから、ガジェットはお勧めしていません。ガジェットにはそれ以外にも、セキュリティにかかわるリスクがあります。プラットフォームとして脆弱であるため、悪意のあるガジェットでなくとも悪用されかねません。このようなことから、Windows 8 を最新鋭の安全な OS にするために OS からガジェットをすべて削除することにしています。Windows 7 でデスクトップにガジェットを使っている場合には、ガジェットがアンインストールされる旨の通知が表示されます。

影響

Windows デスクトップではデスクトップ ガジェットを使うことができなくなります。

軽減策

デスクトップ ガジェットの API とフォルダー構造は Windows 8 でもそのまま残ります。ガジェットをプログラムからインストールし、この API を使ってガジェットを実行するアプリケーションは、Windows 8 でも問題なく動作します (ただし、ガジェットそのものは機能しません)。

解決策

デスクトップ アプリを開発するときには、インストーラー パッケージにガジェットを入れないようにしてください。ユーザーにとってはガジェットが入っていても領域を占有するばかりで、システムで動作しません。

テスト

Windows 7 のデスクトップ ガジェットのオプション コンポーネントを無効にすることによって、この変更の互換性をテストできます。Windows 7 PC でガジェットを無効化するには、以下の手順を実行します。

1. コントロール パネルから [Windows の機能の有効化または無効化] に移動します。
2. [Windows ガジェット プラットフォーム] の隣にあるチェック ボックスをオフにします。
3. [OK] をクリックし、画面上の指示に従います。

参考資料

「マイクロソフト セキュリティ アドバイザリ: ガジェットの脆弱性により、リモートでコードが実行される」
http://go.microsoft.com/fwlink/p/?linkid=325442

Advanced Format (4K) ディスクの互換性の更新

クライアント – Windows XP | Windows Vista | Windows 7 | Windows 7 SP1 | Windows 8

サーバー – Windows Server 2003 | Windows Server 2008 | Windows Server 2008 R2 | Windows Server 2008 R2 SP1 | Windows Server 2012

説明

ここで取り上げる内容は、Windows 7 SP1 と Windows Server 2008 R2 SP1 向けに書かれた記事「512-byte Emulation (512e) Disk Compatibility Update (512 バイト エミュレーション (512e) ディスクの互換性の更新)」を改訂したものです。この改訂版には新たに多くの情報が盛り込まれており、その一部は Windows 8 と Windows Server 2012 のみが対象です。

面記録密度は年々向上傾向にあり、最近では 3 TB ディスクが登場したことにより、S/N 比 (SNR) の減少に対処するために使用される、エラー訂正メカニズムのスペース効率が低下しています。そのため、メディアを確実に利用するのに、より多くのオーバーヘッドが必要になります。こうしたエラー訂正メカニズムの改善策の 1 つとして、ストレージ業界は、物理セクター サイズを大きくした新たな物理メディア形式を導入しています。この新しい物理メディア形式は、"Advanced Format" と呼ばれます。そのため、最近の記憶装置では従来のセクター サイズに関する前提が通用しなくなっており、開発者は既存のコードが基盤とする前提条件を調べて、影響がないかどうかを確認する必要があります。

ここでは、ソフトウェアに対する Advanced Format 記憶装置の影響を示し、このタイプのメディアをサポートするためにアプリで可能な対応について説明するほか、こうしたタイプのデバイスを開発者がサポートできるよう、マイクロソフトが Windows Vista、Windows 7、Windows 8 に導入したインフラストラクチャを紹介します。Advanced Format ディスクとの互換性を向上させるためのガイドラインを示していますが、この情報は、Windows Vista、Windows 7、Windows 8 を実行し、Advanced Format ディスクを使うシステム全般に適用されます。

新しい大容量セクター関連の機能の概要

一般ユーザーと開発者がセクター サイズの大きいディスクを使用する際の操作性向上に役立つ、Windows 8 および Windows Server 2012 の新機能を以下にまとめ、各項目の下にその内容を記載しています。

- Windows 7 SP1 の、エミュレーションを使用した 4K ディスクのサポート (512e) を基盤とし、エミュレーションを使用せずに、4K セクター サイズのディスクを標準でフル サポートします (4K ネイティブ)。サポート対象のアプリおよびシナリオには、以下が挙げられます。
    - エミュレーションを使用しない 4K セクター ディスク (4K ネイティブ ディスク) に Windows をインストールしてブート可能
    - 新しい VHDx ファイル形式
    - Hyper-V のフル サポート
    - Windows バックアップ
    - NT ファイル システム (NTFS) でのフル サポート
    - **新機能** Storage Spaces and Pools (SSP) によるフル サポート
    - Windows Defender によるフル サポート

- 物理セクター サイズを照会する新しい API (FileFsSectorSizeInformation) を提供します。
  - ネットワーク ボリュームに利用可能
  - 任意のファイル ハンドルに対して実行可能
  - 特権のないアプリに利用可能
  - 馴染みやすい使用モデル
- 拡張された "fsutil" コマンド ライン ユーティリティを含み、ボリュームのセクターの論理サイズと物理サイズと共に、配置情報を照会できます (配置情報を照会しないベーシック バージョンのユーティリティについては、Windows 7 用を Microsoft KB 982018 で、Windows Server 2008 R2 用を Microsoft KB 982018 で提供しています)。

Advanced Format (4K) ディスクの概要

このメディア形式に関する変更の導入に伴う問題点の 1 つは、既存のソフトウェアおよびハードウェアで互換性問題が発生する可能性があることです。互換性への一時的な対処策として、ストレージ業界はさしあたり、通常の 512 バイト セクター ディスクをエミュレートするディスクを導入し、標準の ATA コマンドと SCSI コマンドを通じて、実際のセクター サイズに関する情報も利用できるようにしています。このエミュレーションの結果、実質的には、以下の 2 つのセクター サイズが存在することになります。

論理セクター: メディアの Logical Block Addressing に使われる単位。記憶域で許容される、書き込みの最小単位と考えることもできます。これが "エミュレーション" です。

物理セクター: 1 回の処理で完了される、デバイスへの読み取り処理と書き込み処理の単位。これは、アトミックな書き込みの単位です。

最新の Windows API (IOCTL_DISK_GET_DRIVE_GEOMETRY など) は論理セクター サイズを返しますが、IOCTL_STORAGE_QUERY_PROPERTY 制御コードを使うと、物理セクター サイズと STORAGE_ACCESS_ALIGNMENT_DESCRIPTOR 構造体内の BytesPerPhysicalSector フィールドに含まれる関連情報を併せて取得できます。詳細については、本稿で後述します。

大容量セクター メディアの初期のタイプ

ストレージ業界は、物理セクター サイズが 4 KB のメディアに対応する、この新しい Advanced Format タイプの記憶域に移行するために、すばやく取り組みを強化しています。市場では、次の 2 つのタイプのメディアが販売される予定です。

4 KB ネイティブ: このメディアにはエミュレーション レイヤーがなく、論理セクター サイズおよび物理セクター サイズとして、直接 4 KB が使用されます。この新タイプのメディアに付随する全体的な問題として、大多数のアプリおよびオペレーティング システムは物理セクター サイズの照会および I/O の位置合わせを行わないため、予期せぬ I/O の失敗が起こる可能性があります。

512 バイト エミュレーション (512e): 前のセクションで説明したとおり、このメディアにはエミュレーション レイヤーがあり、論理セクター サイズとして (現在の通常のディスクと同様に) 512 バイトを使います。ただし、物理セクター サイズ情報 (4 KB) も利用可能です。この新タイプのメディアに付随する全体的な問題として、大多数のアプリとオペレーティング システムは物理セクター サイズの存在を認識しないため、以降で説明するような多数の問題が起こる可能性があります。

Windows による大容量セクター メディアのサポート状況

次の表に、さまざまなメディアに関する、マイクロソフトの公式サポート ポリシーとセクター サイズのレポート結果を示します。詳しくは、サポート技術情報の記事をご覧ください。

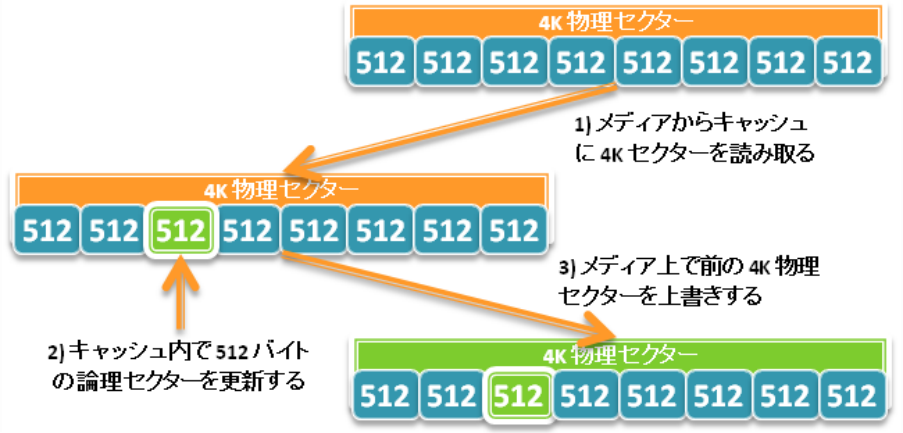
一般的な名称	レポートされる論理セクターサイズ	レポートされる物理セクターサイズ	サポートされる Windows のバージョン
512 バイト ネイティブ、512n	512 バイト	512 バイト	全バージョンの Windows
Advanced Format、512e、AF、512 バイト エミュレーション	512 バイト	4 KB	Windows 8 Windows Server 2012 Windows 7 (MS KB 982018 を適用) Windows 7 SP1 Windows Server 2008 R2 (MS KB 982018 を適用) Windows Server 2008 R2 SP1 Windows Vista (MS KB 2553708 を適用) Windows Server 2008 (MS KB 2553708 を適用)
Advance Format、AF、4K ネイティブ、4Kn	4 KB	4 KB	Windows 8 Windows Server 2012
その他	4 KB または 512 バイト以外	4 KB または 512 バイト以外	サポート対象外

注: 上の表では強調していませんが、Windows XP、Windows Server 2003、Windows Server 2003 R2 は 512e や 4Kn のメディアをサポートしていません。システムを起動し、最低限の運用はできる可能性がありますが、機能上の問題、データ損失、パフォーマンス低下といった未知のシナリオの発生も考えられます。そのため、Windows XP による 512e メディアの使用や、Windows XP コードベースに基づいたその他の製品 (Windows Home Server 1.0、Windows Server 2003、Windows Server 2003 R2、Windows XP 64-bit Edition、Windows XP Embedded、Windows Small Business Server 2003、Windows Small Business Server 2003 R2 など) の使用には特に注意が必要です。

エミュレーションのしくみ: Read-Modify-Write (RMW)

記憶域メディアでは、一定の決められた単位内で物理メディアを書き換えることができます。つまり、メディアには、物理セクター サイズの単位でのみ書き込み/再書き込みを行えます。そのため、この単位レベル以外で書き込みを実行すると、余分な手順が必要となります。この点について、次の例で詳しく説明します。

このシナリオでは、あるアプリが、512 バイトの論理セクター内に格納されている Datastor レコードの内容を更新する必要があるとします。次の図に、この書き込みを完了するために記憶装置で必要となる手順を示します。



上図からわかるように、このプロセスでは記憶装置による所定の処理が発生し、パフォーマンスを低下させる可能性があります。こうした追加処理を回避するには、以下の処理を行うようにアプリを更新する必要があります。

- 物理セクター サイズを照会する
- レポートされた物理セクター サイズに基づいて、書き込みの位置合わせを行う

このことは一見、パフォーマンス上の問題にすぎないと思えるかもしれませんが、より深刻な問題にもなりかねません。以降で詳しく説明していきます。

回復性: Read-Modify-Write の見えないコスト

回復性とは、複数のセッションをまたいでアプリの状態を復元する能力を表します。マイクロソフトは、512 バイト セクターの書き込み (Read-Modify-Write サイクル) を実行するために、512e 記憶装置に何が必要かを検討しました。次に、メディア上で以前の物理セクターを上書きする処理が中断された場合、何が起こるかを考えてみましょう。結果はどうなるでしょうか。

- 大部分のハード ディスク ドライブの更新が実施されているため、部分的に上書きされたことにより、物理セクター (つまり、メディア上の物理セクターが位置する箇所) が破損して情報が不完全になっている可能性があります。言い換えれば、8 つの論理セクター (対象の物理セクターに論理的に含まれる) がすべて損失している可能性があるということです。
- データ ストアを備えるアプリは、メディア エラーからの回復機能を備えている場合がほとんどですが、8 セクター (つまり 8 つのコミット レコード) の損失により、データ ストアの正常な回復が不可能になってもおかしくありません。管理者がバックアップから、データベースを手動で復元しなければならなくなるほか、時間のかかる再ビルドが必要となることも考えられます。
- もう 1 つ、より重要な影響として、Read-Modify-Write サイクルを発生させる他のアプリの動作によって、対象のアプリを実行していないにもかかわらず、関連するデータが失われる可能性もあります。これは単純に、対象のデータと他のアプリのデータが、同じ物理セクター内に格納されていたために発生します。

上記を踏まえ、アプリケーション ソフトウェアではコードの前提となっている条件をすべて再評価し、論理セクター サイズと物理セクター サイズの違いを把握しておくことが重要です。また、本稿で後述する、興味深い顧客事例も参考にしてください。

適切な対応 (Read-Modify-Write の回避)

一部のストレージ ベンダーは、特定の 512e 記憶装置にいずれかのレベルの軽減策を導入し、Read-Modify-Write サイクルのパフォーマンスおよび回復性の問題を緩和しようとする可能性もありますが、ワークロードに関して軽減策ができる対処には限界があります。そのため、アプリでは、長期的な解決策としてこうした軽減策に頼るべきではありません。さらに、すべてのクラスのディスクでこうした軽減策が講じられる保証はなく、その軽減策に十分効果があるかどうかも不明です。

この問題を解決するには、ドライブの軽減策に頼るのではなく、アプリが適切な手順を実行することで、このタイプのメディアをサポートできるように設計する必要があります。以降では、大きいセクター サイズのディスクを使用した場合に、アプリに問題が発生する可能性のある一般的なシナリオについて説明し、各問題の解決を図るための調査方法を提案します。

問題 1: パーティションと物理セクターの境界が一致しない

管理者またはユーザーがディスクのパーティションを作成する場合、1 つ目のパーティションがセクターの境界に合わせて作成されるとは限りません。そのため、後続の書き込みがすべて、物理セクターの境界と一致しなくなる可能性があります。Windows Vista SP1 および Windows Server 2008 の場合、1 つ目のパーティションはディスクの最初から 1,024 KB の位置に配置されるため (4 GB 以上のディスクの場合。それ以外では、64 KB に配置)、4 KB の物理セクター境界と一致します。ただし、Windows XP の既定のパーティション分割またはサード パーティ製のパーティション分割ユーティリティを使用した場合や、Windows API の使用方法が正しくない場合は、作成されたパーティションが物理セクターの境界と一致しないことがあります。位置合わせを確実に行うため、開発者は正しい API が使用されていることを確認する必要があります。以下に、パーティションを確実に位置合わせするために、推奨する API を示します。

`IVdsPack::CreateVolume API` および `IVdsPack2::CreateVolume2` API は、新たにボリュームを作成する場合、指定された配置パラメーターの代わりにオペレーティング システムの配置の既定値を使用します (Windows Vista SP1 より前は 63 バイト、Windows Vista SP1 以降は上記の既定値を使用)。代わりに、`IVdsCreatePartitionEx::CreatePartitionEx API` または `IVdsAdvancedDisk::CreatePartition API` を使用すれば、パーティションの作成を要するアプリに、指定した配置パラメーターが適用されます。

適切な位置合わせを確実に行う最も良い方法は、最初にパーティションを作成する時点で、正しく位置を合わせておくことです。それ以外の場合は、位置合わせに配慮しながらアプリで書き込みまたは初期化を実行しなければならないため、処理が非常に複雑化します。

問題 2: バッファーを介さない書き込みが物理セクター サイズと一致しない

最もシンプルな問題として、バッファーを介さない書き込みは、記憶域メディアのレポートされた物理セクター サイズと一致しません。一方で、バッファーを介した書き込みは、ページ サイズの 4 KB に調整されるため、大容量セクター メディアの第 1 世代の物理セクター サイズとも一致します。ただし、データ ストアを備えたアプリは、バッファーを介さずに書き込みを実行することがほとんどであるため、こうした書き込みが物理セクター サイズの単位で実行されるようにする必要があります。

次の例に挙げるようなシナリオでは、実行されるアプリ I/O の位置合わせが行われません。

512 バイト セクターに合わせて調整されるコミット レコード: データ ストアを備えたアプリでは、一定の形式のコミット レコードを使って、メタデータの変更に関する情報か、データ ストアの構造を保持しているのが一般的です。セクターの損失が複数のレコードに影響を及ぼさないようにするため、このコミット レコードは通常、セクター サイズと一致するようにパディングによって調整されます。物理セクター サイズを大きくしたディスクを使用する場合、前のセクションで述べたように、アプリは物理セクター サイズを照会し、そのサイズと一致するように各コミット レコードを調整する必要があります。4K ディスクでは、この方法によって I/O の障害を防ぐことができます。512e では、これによって Read-Modify-Write サイクルを回避できるだけでなく、物理セクターが損失した場合に、失われるコミット レコードを 1 件のみに抑えることができます。

位置が未調整のチャンクに書き込まれるログ ファイル: 通常、バッファーを介さない I/O は、ログ ファイルの更新や追加書き込みの際に使用されます。バッファーを介した I/O に切り替えるか、ログの更新を内部的にバッファー処理して物理セクター サイズに合わせることにより、アプリでの I/O の障害や Read-Modify-Write の発生を回避できます。

アプリがバッファーを介さずに I/O を実行しているかどうかを確認できるようにするため、CreateFile 関数を呼び出す際は、**dwFlagsAndAttributes** パラメーターに必ず FILE_FLAG_NO_BUFFERING フラグを含めます。

さらに、現在書き込みをセクター サイズに合わせている場合、このセクター サイズは "論理" セクター サイズである可能性がきわめて高いといえます。これは、既存のほとんどの API が照会するメディアのセクター サイズは、アドレス指定の単位、つまり論理セクター サイズであるためです。ここで対象となるセクター サイズは物理セクター サイズであり、アトミックな実際の単位です。論理セクター サイズを取得する API には、次のようなものがあります。

- GetDiskFreeSpace、GetDiskFreeSpaceEx
- FileFsVolumeInformation
- IOCTL_DISK_GET_DRIVE_GEOMETRY、IOCTL_DISK_GET_DRIVE_GEOMETRY_EX
- IVdsDisk::GetProperties、IVdsDisk3::GetProperties2

次に、物理セクター サイズを照会する方法を説明します。

Windows 8 で推奨される方法

Windows 8 には、開発者がアプリに 4K のサポートを簡単に統合できるようにするため、新しい API が導入されています。この新しい API では、以下で説明するように、Windows Vista と Windows 7 の従来の方法よりもさらに多くのシナリオがサポートされています。この API では、次の呼び出しシナリオを使うことができます。

- 特権のないアプリからの呼び出し
- 任意の有効なファイル ハンドルへの呼び出し
- SMB2 を経由した、リモート ボリューム上のファイル ハンドルへの呼び出し
- 簡素化されたプログラミング モデル

API は、以下に定義されているように、新しい情報クラス FileFsSectorSizeInformation に FILE_FS_SECTOR_SIZE_INFORMATION という関連付けられた構造体が指定された形式になっています。

```
typedef struct _FILE_FS_SECTOR_SIZE_INFORMATION {
    ULONG LogicalBytesPerSector;
    ULONG PhysicalBytesPerSectorForAtomicity;
    ULONG PhysicalBytesPerSectorForPerformance;
    ULONG FileSystemEffectivePhysicalBytesPerSectorForAtomicity;
    ULONG Flags;
    ULONG ByteOffsetForSectorAlignment;
    ULONG ByteOffsetForPartitionAlignment;
} FILE_FS_SECTOR_SIZE_INFORMATION, *PFILE_FS_SECTOR_SIZE_INFORMATION;
```

Windows 7 および Windows Vista での従来の方法

Windows Vista および Windows Server 2008 には、AHCI ベースの記憶域コントローラーを対象に、接続された記憶装置の物理セクター サイズを照会する API が導入されています。Windows 7 および Windows Server 2008 R2 では、SP1 の時点で (またはマイクロソフト サポート技術情報の 982018 により)、このサポートが Storport ベースの記憶域コントローラーにも拡張されています。マイクロソフトは MSDN で、アプリからボリュームの物理セクター サイズを照会する方法について説明したコード サンプルを公開しています。コード サンプルは、http://msdn.microsoft.com/library/ff800831(v=VS.85).aspx にあります。

上記のコード サンプルを使用すると、ボリュームの物理セクター サイズを取得できますが、レポートされた物理セクター サイズを使用する前に、所定の基本的なサニティ チェックを実行する必要があります。これは、一部のドライバーに関して、フォーマット済みデータが正しく返されないという事例が見られるためです。

- レポートされた物理セクター サイズが、レポートされた論理セクター サイズ以上であることを確認する。これに該当しない場合、アプリではレポートされた論理セクター サイズと等しい物理セクター サイズを使用する必要がある
- レポートされた物理セクター サイズが、2 のべき乗であることを確認する。これに該当しない場合、アプリではレポートされた論理セクター サイズと等しい物理セクター サイズを使用する必要がある
- 物理セクター サイズが、512 バイトから 4 KB の間の 2 のべき乗値である場合、使用する物理セクター サイズの端数を切り捨て、レポートされた論理セクター サイズと同じにすることを検討する必要がある
- 物理セクター サイズが 4 KB を超える 2 のべき乗値である場合、その値を使用する前に、このシナリオに関するアプリの処理能力を評価する必要がある。それ以外の場合は、使用する物理セクター サイズの端数を切り捨て、4 KB にすることを検討する

この IOCTL を使用して物理セクター サイズを取得するには、いくつか制約があります。具体的には以下のとおりです。

- 昇格された特権が必要。特権なしでアプリを実行している場合は、必要に応じて上述の Windows サービス アプリケーションを記述する
- SMB ボリュームをサポートしない。これらのボリュームに関する物理セクター サイズの照会をサポートする場合も、必要に応じて Windows サービス アプリケーションを記述する

- ファイル ハンドルに対しては実行できない (IOCTL はボリューム ハンドルに対して実行する必要がある)

問題 3: 512 バイト セクターに依存するファイル形式

こうしたファイルは、標準のファイル形式を使用する一部のアプリ (VHD 1.0 など) で使用され、512 バイトのセクター サイズを前提としてハードコーディングされています。そのため、対象ファイルの更新と書き込みを実行すると、デバイス上で Read-Modify-Write サイクルが発生し、利用者の環境でパフォーマンスと回復性の問題を引き起こす可能性があります。ただし、アプリでこうしたタイプのメディア上での運用をサポートするには、複数の方法があります。

- バッファーを利用し、物理セクター サイズの単位で書き込みが実行されるようにする
- Read-Modify-Write を内部的に実行し、レポートされた物理セクター サイズの単位で更新が実行されるようにする
- 可能な場合は、物理セクター サイズと一致するように、パディングによってレコードを調整する。このとき、パディングは空のスペースと解釈されるようにする
- 大きいセクターをサポートするように、アプリ データ構造のバージョンの再設計を検討する

問題 4: レポートされる物理セクター サイズがセッション間で変化することがある

多くのシナリオでは、前述の Datastor をホストする基本となる記憶域について、物理セクター サイズとしてレポートされる値が変化することがあります。その最も一般的な例が、別のボリュームに Datastor を移行する (ネットワークをまたぐ移行も含む) 場合です。多くのアプリにとって、レポートされる物理セクター サイズの変更は想定外のイベントであるため、アプリによっては再初期化に失敗する可能性があります。

これは簡単に対応できるシナリオではなく、ここでは注意を促すために示しています。4K ネイティブや 512e メディアを使用することで利用者が悪影響を受けないようにするため、利用者のモビリティ要件を考慮し、それに応じてサポートを調整する必要があります。

ボリュームの論理セクター サイズと物理セクター サイズを取得する方法

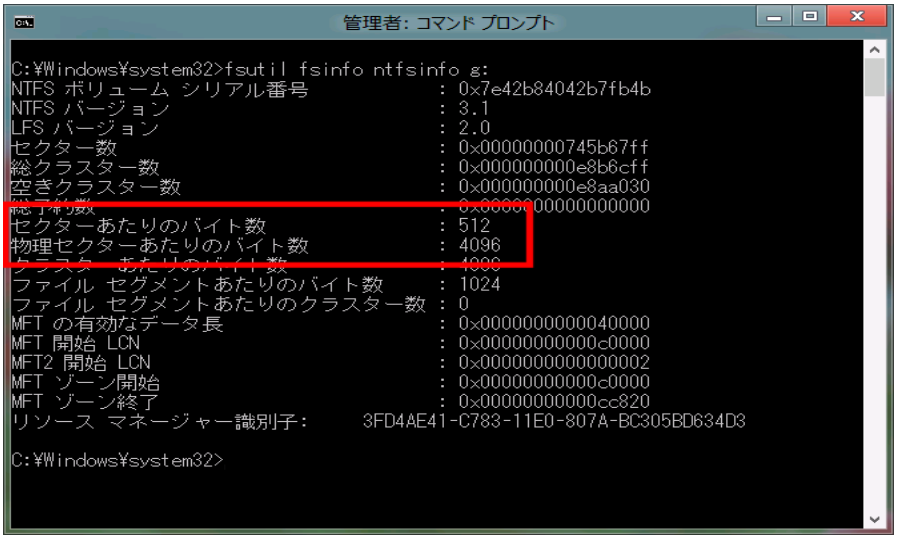
Windows には、ボリュームのセクター サイズ情報を表示するユーティリティが付属しています。"fsutil" をサポートする Windows のバージョンは以下のとおりです。

- Windows 8
- Windows Server 2012
- Microsoft サポート技術情報 982018 を適用した Windows 7 SP1
- Microsoft サポート技術情報 982018 を適用した Windows 7
- Microsoft サポート技術情報 982018 (v3) を適用した Windows Server 2008 R2 SP1
- Microsoft サポート技術情報 982018 (v3) を適用した Windows Server 2008 R2
- Microsoft サポート技術情報 2553708 を適用した Windows Vista
- Microsoft サポート技術情報 2553708 を適用した Windows Server 2008

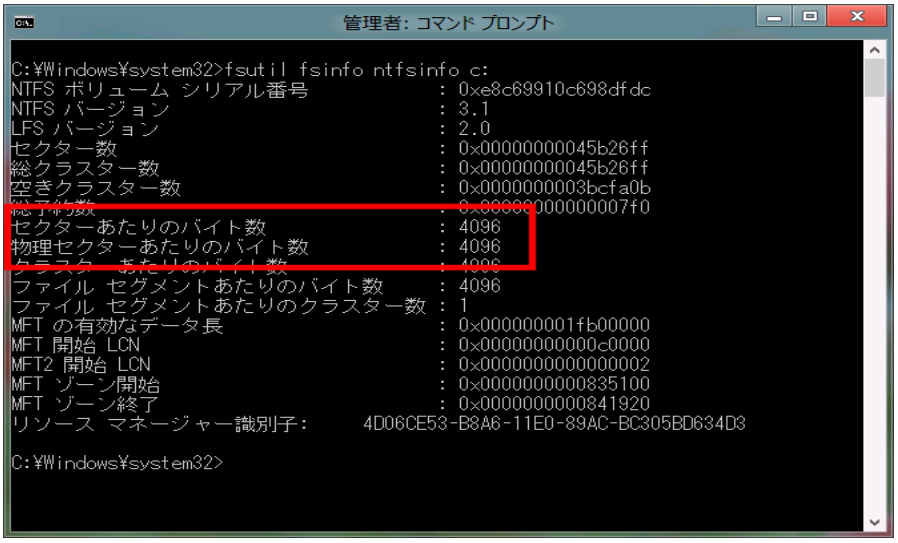
セクター サイズ情報を取得するには、管理者特権のコマンド プロンプトから次のようにユーティリティを呼び出します。

```
fsutil fsinfo ntfsinfo <drive letter>
```

512 バイト エミュレーションを適用した 4K セクター ディスクでは、セクターあたりのバイト数フィールドが 512 に、物理セクターあたりのバイト数フィールドが 4,096 に設定されています (下図を参照)。



4K ネイティブ ディスクでは、セクターあたりのバイト数フィールドおよび物理セクターあたりのバイト数フィールドが両方とも、4,096 に設定されています (下図を参照)。



512 バイト ネイティブ ディスクでは、セクターあたりのバイト数フィールドおよび物理セクターあたりのバイト数フィールドが両方とも、512 に設定されています。

注: "物理セクターあたりのバイト数" フィールドに "サポートなし" と表示される場合、記憶域ドライバーが IOCTL_STORAGE_QUERY_PROPERTY をサポートしていないか、情報の取得中にエラーが発生したことを示しています。

参考資料

Windows の一般的なサポートに関する声明
http://go.microsoft.com/fwlink/p/?linkid=325443

Windows 7 および Windows Server 2008 R2 の修正プログラム
http://go.microsoft.com/fwlink/p/?LinkId=325446

Windows Vista および Windows Server 2008 の修正プログラム
http://go.microsoft.com/fwlink/p/?LinkId=325447

Hyper-V のサポートに関する声明
http://go.microsoft.com/fwlink/p/?linkid=325448

IOCTL_STORAGE_QUERY_PROPERTY 制御コードに関する一般情報
http://go.microsoft.com/fwlink/p/?linkid=325449

IOCTL_STORAGE_QUERY_PROPERTY 制御コード
http://go.microsoft.com/fwlink/p/?LinkId=325450

STORAGE_ACCESS_ALIGNMENT_DESCRIPTOR 構造体に関する一般情報
http://go.microsoft.com/fwlink/p/?LinkId=325460

マイクロソフトのソフトウェアの更新で使用される一般的な用語の説明
http://go.microsoft.com/fwlink/p/?LinkId=325461

IOCTL_STORAGE_QUERY_PROPERTY 制御コードを呼び出すときに STORAGE_ACCESS_ALIGNMENT_DESCRIPTOR 構造体から報告される記憶域アクセス整列情報を抽出する方法が詳しく記載された **WDK サンプル コード**
http://go.microsoft.com/fwlink/p/?LinkId=325462

ImageX コマンド ライン オプションに関する一般情報
http://go.microsoft.com/fwlink/p/?LinkId=325463

4 KB セクター ドライブをサポートするためのインテル チップセット ドライバーの要件
http://go.microsoft.com/fwlink/p/?LinkId=325464

論理単位の仮想プロビジョニング

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows 仮想プロビジョニングは、エンド ツー エンドの記憶域プロビジョニング ソリューションへのインターフェイスとなります。可用性に優れ、スケーラブルでユーザー フレンドリな記憶域のプロビジョニング サービスを実現するには、サーバー ホストから目的の記憶装置までをカバーする、信頼性の高い実装が求められます。Windows の仮想プロビジョニング機能を利用すると、仮想プロビジョニングされた論理ユニット (LUN) の識別、しきい値の通知、消費リソースの処理、スペース再生、Logical Block Addressing (LBA) 状態の取得を通じ、システム管理者、IT マネージャー、およびストレージ管理者が、仮想プロビジョニングされた記憶装置の状態を同時に確認できます。Windows の仮想プロビジョニング機能は、記憶域のプロビジョニングに関する信頼性の高いサービス モデルを提供し、クライアント/サーバー方式のストレージ システム、仮想化記憶域、およびクラウドによる記憶域サービスに適用できます。仮想プロビジョニング機能の優れた品質を確保するため、マイクロソフトは記憶域アレイ製品を対象とする Thin Provisioning ロゴ プログラムを実施しています。

Windows の仮想プロビジョニング ストレージ ソリューションには以下の特徴があります。

- ストレージ管理者が、より少ない物理ディスク リソースで、より大きな LUN を作成できる
- 記憶域のプロビジョニング サービスを中断することなく、物理ディスク リソースを増減できる
- しきい値を設定し、ユーザーに記憶域ボリュームの使用量を通知できる
- 共有の記憶域プール構成を通じた記憶域のプロビジョニングをサポートする
- 記憶域スペースの要求量および使用量に基づいて、記憶域プールのサイズを増減できる

Windows の仮想プロビジョニング機能は、次のように要約できます。

- 仮想プロビジョニングされた LUN の識別
- しきい値の通知、一時的なリソース消費、および永続的なリソース消費の処理
- 記憶域スペースの再生
- 論理ブロックの状態クエリのマッピング

影響

仮想プロビジョニングされた LUN が Windows でどのように扱われるかを適切に把握していないと、想定外の動作や未知の要因によって、アプリに障害が発生する可能性があります。

仮想プロビジョニングされた LUN の使用

Windows 8 または Windows Server 2012 で仮想プロビジョニングされた LUN を使うには、システムと記憶装置の管理者は次の事項に注意する必要があります。

- 仮想プロビジョニングされた LUN の識別。システム管理者やエンド ユーザーは、デフラグ ツールと記憶域最適化ユーティリティを使うと、記憶装置のメディアの種類を確認できる
- しきい値や永続的なリソース消費に関するイベントが発生した場合、Windows はシステム イベントをログに記録してシステム管理者に通知する
- ファイル削除通知または記憶域の最適化ユーティリティのスケジューラを通じ、記憶域スペースを手動または自動で再生できる
- ストレージ管理者はしきい値に関する通知機能を実装し、システム管理者やエンド ユーザーへのアラートを生成して、記憶域サービスの予期せぬ中断を回避する必要がある

テスト

- Windows 仮想プロビジョニング機能をサポートするには、記憶域アレイが Windows 8 Thin Provisioning 認定を取得している必要がある
- 記憶域アレイの Windows Thin Provisioning ハードウェア認定キットをテスト ツールとして利用すると、記憶域アレイによる Windows 仮想プロビジョニング機能のサポート能力を確認できる
- Windows では、仮想プロビジョニングの LUN とフル プロビジョニングの LUN が、同一システム内で問題なく動作するように設計されている

参考資料

T10 SCSI コマンドの仕様 (SBC3r27)
http://go.microsoft.com/fwlink/p/?LinkId=325465

ハードウェア ダッシュボード サービス
http://go.microsoft.com/fwlink/p/?LinkId=327702

WinPE とサーバー SKU でオプションとなった拡張記憶域

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

拡張記憶域は、Windows 7 USB デバイスでは常に利用可能でした。Windows 8 のリリースに伴い、拡張記憶域はすべての記憶装置で利用できるようになります。拡張記憶域は、クライアントベースの装置では既定で有効化されますが、サーバー装置ではオプション扱いとなるため、WinPE を通じてプロビジョニングする必要があります。

影響

拡張記憶域が有効化されていないと、サーバー装置で障害が発生します。

起動デバイスは WinPE を通じてプロビジョニングし、この機能を有効化する必要があります。

解決策

拡張記憶域はランタイム サーバーではオプション扱いのため、開発者が WinPE に追加して、対象のドライブをプロビジョニングおよびアクティブ化する必要があります。詳細については、展開ガイドを参照してください。

その後、拡張記憶域を使えるよう、サーバー管理者が明示的にサーバーを構成する必要があります。

仮想ディスク サービスから Windows 記憶域管理 API への移行

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows 8 と Windows Server 2012 以降では、仮想ディスク サービスの COM インターフェイスが、WMI ベースのプログラミング インターフェイスである記憶域管理 API に置き換えられます。記憶域サブシステム、(Windows) ディスク、パーティション、ボリュームの管理については、記憶域管理 API を使うことを強くお勧めします。詳しくは、「Windows Storage Management APIWindows (記憶域管理 API)」をご覧ください。
ダイナミック ディスクは、ミラー ブート ボリューム (ミラー ボリュームを使ってオペレーティング システムをホストする) を除くすべての用途で廃止されます。ドライブ障害に対する回復性を必要とするデータについては、耐障害性を備えた記憶域仮想化ソリューションの記憶域スペースを使ってください。詳しくは、「Storage Spaces Technical Preview (記憶域スペースのテクニカル プレビュー)」をご覧ください。

移行が完了するまでの間、DiskPart、DiskRAID、およびディスク管理 GUI は引き続き使用できますが、こうしたツールは記憶域スペースとは併用できず、その他の新しい WMIv2 ベースの Windows 記憶域管理 API や、付属の記憶域管理ユーティリティまたはクライアントとも併用できません。

	API		ツール				
	VDS	**WMI**	**DiskPart**	**DiskRAID**	**ディスク管理 GUI**	**PowerShell**	**記憶域スペース コントロール パネル**
記憶域サブシステム	○	○	該当なし	○	該当なし	○	該当なし
ベーシック ディスク	○	○	○	該当なし	○	○	×
ダイナミック ディスク	○	×	○	該当なし	○	×	×
記憶域スペース	×	○	×	該当なし	×	○	○

この移行の結果、記憶域の回復性、可用性、およびスケーラビリティの向上や、スクリプティングとプログラミングの言語の統合が可能となるほか、記憶域の管理コストの削減と、リモート記憶域の管理作業の簡素化が実現されます。

影響

VDS 環境で使用される DiskPart ユーティリティと DiskRAID ユーティリティは、新しい記憶域スペースをサポートしません。同様に、新しい Storage PowerShell ユーティリティは、廃止されるダイナミック ディスクをサポートしません。

軽減策

マイクロソフトは、Windows 記憶域管理 API 上にいずれかの新しい記憶域管理アプリを導入し、標準の更新サイクルの間に、VDS インフラストラクチャに基づいた既存のアプリから、Windows 記憶域管理 API に移行できるように計画することを強くお勧めします。

参考資料

Windows 記憶域管理 API
http://go.microsoft.com/fwlink/p/?linkid=325468

Windows PowerShell の記憶域関連のコマンドレット
http://go.microsoft.com/fwlink/p/?linkid=325469

Windows Management Instrumentation
http://go.microsoft.com/fwlink/p/?LinkId=325470

Windows PowerShell
http://go.microsoft.com/fwlink/p/?LinkId=325471

ローカル ボリュームの [以前のバージョン] UI の削除

プラットフォーム

クライアント – Windows 8

説明

以前のバージョンの Windows では、ローカル ボリュームの "シャドウ コピー" を使ってユーザーが前のバージョンのファイルを個別に復元できました。こうしたシャドウ コピーは、ボリューム シャドウ コピー サービス (VSS) を通じ、"以前のバージョン" 機能によって作成されます。Windows 7 では、[システムのプロパティ] から利用できる [システムの保護] UI で、シャドウ コピーの設定とスケジュール指定を行えました。

ただし、"以前のバージョン" 機能はほとんど使われておらず、全体的な Windows のパフォーマンスに悪影響を及ぼしていたために削除されました。Windows 8 では、次の機能は利用できなくなっています。

- [以前のバージョン] UI から、以前のバージョンのファイルを閲覧、検索、または復元する機能
- [システムの保護] UI から、以前のバージョンのファイルを設定またはスケジュール指定する機能

ただし、Windows サーバー上のファイル共有にアクセスする場合は、引き続き [以前のバージョン] タブを使用できます。

影響

ローカル ボリュームについては、[以前のバージョン] オプションがエクスプローラーのプロパティ メニューに表示されなくなります。

解決策

開発者がローカル ボリュームのシャドウ コピーを作成する必要がある場合は、カスタム コードを記述し、VSS API を呼び出すことで引き続き対応できます。

MSAHCI から StorAHCI への移行

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

StorAHCI (Storport ミニポート) は、Windows で Serial Advanced Technology Attachment (SATA) Advanced Host Controller Interface (AHCI) コントローラーをサポートし、MSAHCI (ATAport ミニポート) の後継となります。

影響

機能やパフォーマンス上の変更はありません。

このドライバーは、MSAHCI とまったく同じデバイスをサポートします。

軽減策

ATAport のバインドに依存する、すべてのユーティリティおよびスクリプトを更新します。ドライバー名に依存しないようにしてください。代わりに、標準のディスク検出機能を使ってください。

Windows 7 の "バックアップと復元" の廃止

プラットフォーム

クライアント – Windows 8

説明

"バックアップと復元" に動作上の変更はありませんが、同機能は廃止予定で、今後は更新されません。"バックアップと復元" はあまり使用されなかったため、新しい "ファイル履歴 " 機能に置き換えられました。"バックアップと復元" 機能は Windows 8 にも含まれます。Windows 7 から Windows 8 にアップグレードしても、"バックアップと復元" 機能をどうしても使用したい場合は、引き続き同機能を利用できます。ただし、"バックアップと復元" へのアクセス ポイントは、コントロール パネルを除いてすべて削除されています。"バックアップと復元" によって使用されていたコントロール パネル アプレットは、"Windows 7 のファイルの回復" に名称変更されました。

自社のイメージで、バックアップ通知を無効化するようにレジストリ キーを設定していた OEM では、この作業が必要なくなります。

両機能を同時に使用するのは控えてください。"ファイル履歴" はアクティブなバックアップ スケジュールがあるかどうかを確認し、存在する場合は、そのスケジュールを有効化できないようにします。この場合、"ファイル履歴" を使用するユーザーは、対象の Windows バックアップのスケジュールを削除する必要があります。

影響

ワークフローが中断される可能性があります。Windows の "バックアップと復元" 機能へのアクセスがかかわる、以前の方式について言及したドキュメントには、上記の変更点を反映する必要があります。

軽減策

"バックアップと復元" を起動する可能性のあるアプリは、"ファイル履歴" が有効になっているかどうかを確認し、ユーザーがどちらの方式を利用するかを選択できるようにコードを変更する必要があります。

オフロード データ転送

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

記憶域のデータ移動を効率化するため、マイクロソフトはオフロード データ転送 (ODX) という新しいデータ転送テクノロジを開発しました。Windows ODX でコピー処理を開始する場合は、バッファーを介して読み取り/書き込み処理を実行する代わりに、読み取りをオフロードし、記憶装置のデータを表すトークンを取得します。次に、そのトークンを使用して書き込みをオフロードするコマンドを実行し、コピー元のディスクからコピー先のディスクへとデータを移動するように要求します。このデータ移動はトークンに基づいて、記憶装置のコピー マネージャーによって実行されます。Windows® 8 では、IT マネージャーおよびストレージ管理者が、Windows ODX 機能を使用して記憶装置を対話的に操作し、高速な記憶域ネットワークを通じて、容量の大きなファイルやデータを移動することができます。Windows ODX を使用すると、大容量データを転送する間の、クライアント/サーバー ネットワークのトラフィックおよび CPU 時間の使用率が大幅に削減されます。これは、データの移動がすべて、バックエンドの記憶域ネットワークで処理されるためです。ODX は、仮想マシンの展開や大規模なデータ移行に利用できるほか、記憶装置の階層化をサポートします。このODX および記憶域の仮想プロビジョニング機能を利用することで、物理ハードウェアの導入コストを抑えることができます。

注: この機能は、SPC4 および SBC3 仕様を実装済みの記憶装置でのみ利用可能です。

機能の詳細

- Windows ODX 機能は Windows オペレーティング システムのコピー エンジンに組み込まれており、Windows は記憶域をリストアップする際に、記憶装置が ODX に対応しているかどうかを照会する
- コピーのオフロードをサポートするには、コピー元とコピー先の記憶装置が同一のコピー マネージャーで管理されている必要がある
- コピーのオフロード処理に失敗した場合、記憶装置のコピー マネージャーはアプリのエラー処理のため、適切な追加のセンス データを返す必要がある
- コピーのオフロード処理に失敗すると、Windows コピー エンジンは、従来のコピー処理にフォールバックする

ODX の使用方法

- データ転送アプリでは、コピー元とコピー先の LUN が両方とも ODX に対応していることを確認したうえで、ODX API ルーチンを呼び出す必要がある
- エクスプローラーを使用すると、"ドラッグ アンド ドロップ" や "コピーと貼り付け" を使用してコピーのオフロードを実行できる

- コピー元とコピー先の LUN にファイル システムがマウントされている場合、アプリは FSCTL_Offload_Read および FSCTL_Offload_Write を呼び出すだけで、コピー元 LUN からコピー先 LUN にデータを転送できる
- コピーのオフロード処理に失敗した場合、記憶装置のコピー マネージャーはアプリのエラー処理のため、適切な追加のセンス データを返す必要がある
- コピー元 LUN またはコピー先 LUN にファイル システムがマウントされておらず、ロックされていない場合、コピーのオフロードを実行するには、IOCTL_STORAGE_MANAGE_DATA_SET_ATTRIBUTES と共に、DeviceDsmAction_OffloadRead アクションまたは DeviceDsmAction_OffloadWrite アクションをアプリが呼び出す必要がある
- コピー元とコピー先の両方の LUN にファイル システムがマウントされておらず、ロックされていない場合は、記憶域管理アプリで SCSI_PASS_THROUGH API を使用すれば、データ転送をオフロードできる

テスト

- 安定したユーザー エクスペリエンスを確保するため、記憶域アレイが Windows ODX 認定を受けていることを確認する
- ODX 機能をサポートするには、記憶装置が Windows Offloaded Data Transfers 認定 (従来のロゴ プログラム) 要件を満たしている必要がある
- Windows Offloaded Data Transfers ハードウェア認定キットを使用すると、記憶装置が ODX 機能をサポートするかどうかを確認できる

参考資料

T10 XCOPY Lite の仕様 (11-059r8)
http://go.microsoft.com/fwlink/p/?LinkId=325473

http://go.microsoft.com/fwlink/p/?LinkId=325474

ハードウェア ダッシュボード サービス
http://go.microsoft.com/fwlink/p/?LinkId=327702

SCSI_PASS_THROUGH 構造体
http://go.microsoft.com/fwlink/p/?LinkId=325475

デスクトップ ウィンドウ マネージャーの常時有効化

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows 8 では、デスクトップ ウィンドウ マネージャー (DWM) が常時有効化され、エンド ユーザーとアプリが無効化することはできません。Windows 7 と同様に、DWM はデスクトップを合成するために使用されます。Windows 7 で有効化されていたエクスペリエンスに加え、DWM デスクトップ コンポジションでは新たに、全テーマのデスクトップ コンポジション、ステレオスコピック 3D のサポートに加え、Windows ストア アプリによるエクスペリエンスの管理、分離、保護が可能になります。

全テーマのデスクトップ コンポジション

Windows Vista と Windows 7 では、Aero グラス テーマを使う場合にのみデスクトップ コンポジションが有効化されます。そのため、Windows クラシック スタイルおよびハイ コントラスト テーマのユーザーは、デスクトップ コンポジションによって有効となる、Windows フリップ、高解像度 (DPI) スケーリングのための自動拡大/縮小、サムネイル プレビュー、全画面の拡大鏡といった各種のエクスペリエンスを使用できません。さらに、こうした以前のバージョンの Windows では、デスクトップ コンポジションを有効化した場合と無効化した場合について、アプリケーション開発者が複数のコード パスを作成および維持する必要があります。

Windows 8 では、デスクトップ コンポジションがすべてのテーマに対して有効化されます。そのため、Windows クラシック スタイルおよびハイ コントラスト テーマのユーザーも、デスクトップ コンポジションによって有効となる、Windows フリップ、高解像度 (DPI) スケーリングのための自動拡大/縮小、サムネイル プレビュー、全画面の拡大鏡といった各種のエクスペリエンスを使用できます。さらに、開発者が複数のコード パスを作成して維持する必要がなくなり、開発作業がシンプルになります。

ステレオスコピック 3D のサポート

DWM デスクトップ コンポジションは、ウィンドウを使用した、全画面のステレオスコピック 3D アプリ コンテンツのレンダリングとプレゼンテーションをサポートしています。

Windows ストア アプリによるエクスペリエンスの管理、分離、保護

DWM デスクトップ コンポジションでは、デスクトップ アプリのウィンドウと Windows ストア アプリのウィンドウを分けて管理することにより、デスクトップ アプリのウィンドウを、新しい Windows ストア アプリのウィンドウと分離して保護することができます。デスクトップ コンポジションは、すべてのアプリのウィンドウを管理する役割を果たすため、デスクトップ コンポジションを無効化すると、想定外の動作が起こる可能性があります。さらに、デスクトップ コンポジションは、新しいスタート メニューの合成のほか、新しい Windows オペレーティング システムの中核的要素である、新規導入されたウィンドウ アニメーションにも使用されています。

デスクトップ コンポジションの制御

Windows Vista および Windows 7 では、多くのシナリオでデスクトップ コンポジションが無効化されています。Windows 8 では、DWM デスクトップ コンポジションはオペレーティング システムのコア コンポーネントであるため、無効化することはできません。いくつかの例外を除いて、デスクトップ コンポジションは常時有効化され、ユーザーがログオンする前に起動された後、セッションの有効期間中はアクティブな状態で維持されます。以降では、Windows 7 でデスクトップ コンポジションが無効化されるシナリオに、Windows 8 がどのように対応するかを説明します。

サーバー SKU および特定のクライアント SKU

Windows 8 では、すべてのサーバー SKU およびクライアント SKU でデスクトップ コンポジションが有効化されます。そのため、サーバー管理者およびユーザーが、デスクトップ コンポジションによって有効化されるエクスペリエンスを確実に活用できます。

デスクトップ コンポジションの基本要件

Windows 8 がサポートする WDDM ドライバーおよびシステムの色深度により、グラフィック アダプターとシステムの色深度の要件は確実に満たされます。

WDDM ドライバーのサポート

システムに WDDM 対応のグラフィック ドライバーがない場合、Windows 8 は既定のアダプターとして Microsoft ベーシック ディスプレイ アダプターを使用します。DWM は常に既定のアダプター上で実行されるため、システム上で WDDM 対応のグラフィック ドライバーを (装備されていないか、無効化されているために) 利用できない場合は、Microsoft ベーシック ディスプレイ アダプターを使用してデスクトップを合成します。

Microsoft ベーシック ディスプレイ アダプターは、GPU ではなく CPU を使用してすべての描画を実行するソフトウェア ラスタライザーです。Microsoft ベーシック ディスプレイ アダプター上でデスクトップ コンポジション (特にアニメーション) を実行した場合、GPU 上で実行するときほど処理がスムーズではない場合がある点に注意してください。

システムの色深度

デスクトップ コンポジションは、色深度が 32 bpp に設定されていないと実行されません。Windows 7 では次のシナリオで、システムの色深度を変更できます。

- エンド ユーザーが、Windows ディスプレイ コントロール パネルまたはサード パーティ製コントロール パネルを使ってシステム カラーを変更する場合
- エンド ユーザーが、パブリック API を通じてシステムの色深度を変更するアプリを実行する場合

Windows 7 とは異なり、Windows 8 は 32 bpp 以外の色深度をサポートしません。コントロール パネルを使用して、ユーザーがシステムの色深度を変更することはできなくなります。

さらに、アプリケーション開発者が、API を使用してシステムの色深度を変更することもできません。Windows 8 は、システムの色深度を 32 bpp 未満に変更しようとするアプリを検知し、対象のアプリを実行するには、アプリ互換性 shim の適用が必要であることをユーザーに通知します。ユーザーの確認後、アプリ互換性 shim が適用されます。この shim によって、システムを引き続き 32 bpp で運用しながら、アプリに対しては低カラー モードがエミュレートされます。

WinSAT

Windows 8 では、デスクトップ コンポジションが WinSAT スコアに依存しません。さらに、WinSAT には DWM 評価が含まれなくなります。

アプリの互換性とユーザーのアクション

Windows 8 の場合:

- Windows 7 にあった、デスクトップ コンポジションを無効化するためのオプションがすべて廃止される
- デスクトップ コンポジションが全テーマの合成を担う
- アプリから、**DwmEnableComposition** を使ってデスクトップ コンポジションを無効化できない。下位互換性を維持するため、この API の呼び出しの成功が返されるが、デスクトップ コンポジションは無効化されない
- "デスクトップ コンポジションを無効にする" shim が廃止される
- [アプリケーションのプロパティ] ダイアログ ボックスの [互換性] タブから、"デスクトップ コンポジションを無効にする" ためのオプションが削除される

リモート処理にミラーリング ドライバーを使用するアプリ

Windows 8 の場合:

- リモート処理シナリオではミラー ドライバーをサポートしない。ミラー ドライバーを使用する、既存のアプリのほとんどを引き続き利用しなければならない一方で、DWM が有効な Windows 8 で、既存のミラー ドライバーをサポートするにはインフラストラクチャの変更が必要となるため、ミラー ドライバーを使用する一部の機能やアプリは動作しない可能性がある
- リモート処理シナリオにミラー ドライバーを使用するアプリケーション開発者のために、各種のデスクトップ複製 API をサポートする
- 既存のアクセシビリティ ミラー ドライバーはサポートしない
- Windows 8 との互換性を確保するため、既存のミラー ドライバーを更新する必要がある

リモート デスクトップ接続

Windows 8 では、リモート デスクトップ接続の際、デスクトップ コンポジションが常に有効化されます。Windows 8 リモート コンピューターに接続するクライアント コンピューターでは、Windows クライアントのバージョンにかかわらず、そのリモート デスクトップ セッションに関して常にデスクトップ コンポジションが有効化されます。デスクトップ コンポジションは、クライアント マシン上の複数のモニターのほか、リモート アプリ セッションにも対応します。

加えて、Windows 8 リモート コンピューターに接続する場合、リモート デスクトップ接続クライアントでの次の設定値は効果を持ちません。

- 色深度
- "コンポジションを有効にする" ためのチェック ボックス

この接続の色深度は常に 32 bpp に設定されています。また、デスクトップ コンポジションは常に有効になっています。

Internet Explorer 9 では Direct2D レンダリングによる "リッチ" メタファイルのレンダリングがサポートされない

プラットフォーム

クライアント – Windows Vista | Windows 7 | Windows 8

サーバー – Windows Server 2008 | Windows Server 2008 R2 | Windows Server 2012

説明

Internet Explorer 9 内部のレンダリング方法の変更により、IHTMLElementRender::DrawToDC API と IViewObject::Draw API は、テキストおよびベクター情報を含むリッチ メタファイルの代わりに、Web コンテンツを表す単一のビットマップを含んだメタファイルを作成するようになります。これは、GDI レンダリングから、ハードウェア アクセラレーション対応の Direct2D (D2D) レンダリングへの移行に伴う変更です。

この変更は、上記の API を使って、従来のメタファイルのテキストまたはベクター情報に依存するアプリに影響を及ぼします。

影響

この変更の影響を受けるアプリによっては、アプリの動作障害や誤動作が生じる可能性があります。

軽減策

Web ドキュメントのテキスト情報の抽出のみを必要とするアプリ (情報の配置が必要ない) は、innerText プロパティを使うとテキストを抽出できます。

IViewObject::Draw を使ったアプリでは、ドキュメント モードが以下の条件を満たす場合、FEATURE_IVIEWOBJECTDRAW_DMLT9_WITH_GDI 機能制御キー (FCK) を使って GDI レンダリングに戻すことができます。

- ドキュメント モードが 8 以下
- FCK で、ホストによる GDI の仕様が承認されている
- メタファイル DC が API に渡される

参考資料

IHTMLElementRender::DrawToDC
http://go.microsoft.com/fwlink/p/?LinkId=325476

IViewObject::Draw
http://go.microsoft.com/fwlink/p/?LinkId=325477

IHTMLElement::innerText Property
http://go.microsoft.com/fwlink/p/?LinkId=325478

インターネット機能コントロール (I..L)
http://go.microsoft.com/fwlink/p/?LinkId=325479

DX9 レガシ ハードウェアのサポートに関する変更点

プラットフォーム

クライアント – Windows 8

説明

Intel と AMD/ATI は DX9 グラフィック パーツのサポートを取りやめ、Windows 8 では対象デバイスのドライバーが更新されません。さらに Windows 8 では、前述のデバイスが標準ではサポートされません。こうしたデバイスの最新のドライバーは、WU および OEM/IHV の Web サイトを通じて入手できるものです。Vista 対応の古いドライバーのため、こうした最終バージョンのドライバーの多くは、Windows 8 で使うと問題が発生します。さらに、nVidia は最近、Windows 8 では同社の DX9 (Vista 以前に対応) のモバイル (ノート PC 用) パーツをサポートしないと発表しました。デスクトップ用の DX9 パーツは引き続きサポートされます。

こうしたドライバーとパーツの組み合わせはすべて、Internet Explorer 9 のソフトウェア フォールバック リストに挙げられています。つまり、これらのデバイスについては、パフォーマンスまたは安定性上の理由から、Internet Explorer 9 によってソフトウェア レンダリングが使われます。このことは、Windows ストア アプリによるエクスペリエンスが、こうしたドライバーやパーツでは十分発揮されないことを示しています。

影響

前述のデバイスは標準でサポートされないため、対象のパーツを所有する多くのユーザーは、運用に Microsoft Basic Display Driver を使用することになります。これは WARP ベースの WDDM 1.2 ソフトウェア GPU で、実際にはこのクラスの一部のパーツ (たとえば、Intel GMA500 シリーズ) よりも高速に動作します。Microsoft Basic Display Driver は Aero グラスおよび DWM をサポートし、Windows ストア アプリを実行できます。

ただし、次の制約があるため注意が必要です。

- 複数のモニターやプロジェクターはサポートされない
- 休止状態はサポートするが、スリープには対応していない
- 多くの場合、画面の解像度を変更できない

軽減策

テストによって、許容可能なユーザー エクスペリエンスが実現されないとわかった場合は、提供製品のハードウェア要件から、こうした DX9 対応の古い Intel パーツおよび AMD パーツを除外することも可能です。また、必要に応じて、これらのパーツを使用すると、エクスペリエンスが大幅に低下する場合があることを利用者に警告します。

テスト

上記のデバイスについて、パフォーマンスとエクスペリエンスを評価します。

- 最終バージョンのドライバーを使ってエクスペリエンスを評価
- MSBDD を使ってエクスペリエンスを評価

Windows ストア アプリでは MSAA を利用できない

プラットフォーム

クライアント – Windows 8

説明

Windows 8 では、Windows ストア アプリからアクセス可能なデータについて、MSAA (Microsoft Active Accessibility) による読み取りまたはオートメーションをサポートしません。代替策として、Windows 7 以降で提供される UI オートメーション (UIA) API を使う必要があります。既に使用されている Windows ストア アプリの機能はないため、この変更によって既存の実装が影響を受けることはありません。影響を受けるのは、Windows 8 用に新たに記述されるアプリと機能のみです。開発者は引き続き、MSAA を使ってアクセシビリティ データを提供可能です。Windows ストア アプリ プロバイダーによって MSAA データが提供されている場合、UIA クライアントでもそのデータの読み取りとオートメーションを実行できます。

Windows ストア アプリ用の API をシンプルにするため、マイクロソフトはこうしたアプリのクライアントとして、UI オートメーションのみをサポートすることにしました。UIA クライアントはネイティブの UIA プロバイダーを読み取り可能で、変換の必要はありません。また、UIA クライアントは、UIA-to-MSAA プロキシを通じて変換された MSAA プロバイダーを読み取ることができます。これにより、Windows ストア アプリの開発者が、テストを実行する際の負荷が軽減されます。

MSAA プロバイダーのサポートをやめれば、テスト項目をさらに減らすことができますが、まだ MSAA をベースとしている主要アプリもあり、それらにアクセスできなくなるのは避けたいと考えています。

影響

Windows ストア アプリの開発者は、アプリ モデル内の MSAA の詳細にアクセスできなくなります。

解決策

オートメーションを利用する新たな開発案件では、Windows ストア アプリの機能について MSAA を使わないようにします。

Windows ストア アプリの対話的操作を必要とする場合は、MSAA を使った既存の標準ツールをすべて UIA に切り替えます。Windows ストア アプリの開発者は、UI オートメーション API を使う必要があります。

参考資料

UI オートメーションの基礎
http://go.microsoft.com/fwlink/p/?LinkId=325480

NDIS 6.30 ドライバーでのポート 3 の廃止

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows 8 では、IHV 機能拡張ポート (第 3 ポートとも呼ばれる) を作成または起動するための、NDIS ミニポート WLAN ドライバーのプラットフォームがサポートされなくなります。この機能は Wi-Fi Alliance (WFA) によって Wi-Fi Direct の仕様が策定されるまでの応急手段として、Windows 7 で導入されました。この仕様が完成し、Wi-Fi Direct を使用した認定製品が出始めたため、Windows 8 にはネイティブの Wi-Fi Direct スタックを導入しています。

影響

特になし (第 3 ポートは、必要に応じて WLAN ミニポートドライバーが起動するプラットフォームの機能であるため)。

解決策

レポートされたバージョンが NDIS 6.30 以外の場合は、デバイスの互換性を維持するため、既存ドライバーに対する機能拡張ポートの機能提供は継続されます。ただし、バージョンが NDIS 6.30 の新しい WLAN ドライバーでは、このポートを起動することはできません。こうしたドライバーの開発者は、代わりに Wi-Fi Direct を使う必要があります。

新機能と機能強化

ここでは、ユーザーと開発者の両方のエクスペリエンスを向上させることのできる、機能強化と新機能について説明します。ただし、これらのいくつかは、既存製品および新製品に影響を及ぼすため、開発者は注意が必要です。以下に、機能領域別の新機能と機能強化のリストを示します。

セキュリティ

- 起動時マルウェア対策
- カーネルモードドライバーに関するセキュア ブート機能の署名要件
- メジャー ブート

パフォーマンス

- スタートアップ アプリ
- 自動メンテナンス

ユーザビリティ

- サード パーティ製の入力方式エディター (IME)

記憶域とファイル システム

- アプリによる記憶域メディアへの "TRIM および UnMap" ヒントの送信を実現する新しい API
- 拡張 SRB (Storage Request Block) をサポートするようになったマルチパス I/O
- Resilient File System
- ファイル サーバー API サポート
- 新しいファイル履歴機能
- オペレーティング システムで制御可能となった光学ディスクドライブへの電力供給

USB

- USB 3.0 のサポート

起動時マルウェア対策

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

マルウェア対策 (AM) ソフトウェアは、マルウェアをよりよく検知できるようになっていますが、一方で攻撃者による検知から逃れるルートキットの作成がより巧妙になっています。大部分の AM ベンダーはブート サイクルの早い段階で起動するマルウェアを検出することに、懸命に挑戦しています。一般的に、それらはホスト オペレーティング システムではサポートされていないシステム ハックを作成し、実際にコンピューターが不安定な状態に置かれる可能性があります。これまでに Windows は AM がこれらの初期のブート脅威を見つけて解決する良い方法を提供していませんでした。

Windows 8 には、セキュア ブートと呼ばれる新しい機能が導入されています。この新しい機能では、Windows のブート構成およびコンポーネントを保護し、起動時マルウェア対策 (ELAM) ドライバーをロードします。このドライバーは、他のブート開始ドライバーの前に開始され、これらのドライバーの評価を可能にすることで、Windows カーネルがそれらのドライバーが初期化されるべきかどうかを決定できるようにします。

影響

カーネルによって最初に起動されることにより、ELAM は、任意のサード パーティ製のソフトウェアの前に起動されることが保証されるため、ブート プロセスでマルウェアを検出して初期化されることを防止することができます。

軽減策

ブート ドライバーは、初期化ポリシーに従って ELAM ドライバーから返される分類に基づいて初期化されます。既定では、ポリシーは既知の良質なドライバーと不明なドライバーを初期化しますが、既知の粗悪なドライバーは初期化しません。システム管理者は、グループ ポリシーを通じてカスタム ポリシーを指定することで不明なドライバーからの初期化を防止することができ、ブート プロセスに不可欠なドライバーを有効にすることができます。しかしそれらは改ざんされており、初期化されるおそれがあります。

解決策

ELAM ドライバーは、初期化されるたびにブート開始ドライバーに関する情報を取得するために、カーネル コールバックを登録する必要があります。ELAM ドライバーは、個々のドライバーの分類を返すことができます。これには以下の関数が必要です。

IoRegisterBootDriverCallback() and IoUnRegisterBootDriverCallback()

ELAM ドライバーは、レジストリ コールバックを登録することもできます。これによって個々のブート開始ドライバーによって使用される構成データを検査する ELAM ドライバーを有効にできます。

ELAM ドライバーは必要に応じて、ブート開始ドライバーが使うよりも前に、データをブロックまたは修正することができます。これには以下の関数が必要です。

CmRegisterCallbackEx() and CmUnRegisterCallback()

ELAM ドライバーの要件および API の使用方法の詳細な説明は、起動時マルウェア対策と呼ばれる MSDN のホワイト ペーパーとしてドキュメント化されています。

テスト

ELAM ドライバーは、ブート プロセスの初期段階で Windows カーネルによって開始されるようにするために、特別にマイクロソフトによって署名されている必要があります。署名を取得するために、ELAM ドライバーは、パフォーマンスとその他の動作を検証するための一連の認定テストにパスしなければなりません。これらのテストは、Windows ハードウェア認定キットに含まれます。

参考資料

起動時マルウェア対策のホワイトペーパー
http://go.microsoft.com/fwlink/p/?LinkId=325481

CmRegisterCallbackEx()
http://go.microsoft.com/fwlink/p/?LinkId=325482

CmUnRegisterCallback()
http://go.microsoft.com/fwlink/p/?LinkId=325483

IoRegisterBootDriverCallback()
http://go.microsoft.com/fwlink/p/?LinkId=325484

IoUnRegisterBootDriverCallback()
http://go.microsoft.com/fwlink/p/?LinkId=325485

Windows ハードウェア認定キットによるハードウェア認定についての//build/ でのプレゼンテーション
http://go.microsoft.com/fwlink/p/?LinkId=325486

ハードウェア キットとツール
http://go.microsoft.com/fwlink/p/?LinkId=325487

カーネルモード ドライバーに関するセキュア ブート機能の署名要件

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows 8 と Windows Server 2012 (WinPE を含む) では、署名されたドライバーに関する Windows オペレーティング システムのセキュリティ要件が、ブート キットまたはルート キットを通じて侵入したマルウェアによって迂回されるのを防ぐため、カーネルがロックダウンされています。

この変更は、Unified Extensible Firmware Interface (UEFI) セキュア ブートをサポートするデバイスの、すべてのカーネルモード ドライバーに影響します。UEFI セキュア ブートは、クライアント マシンの Windows 8 認定取得に必要であり、サーバーへの実装は任意です。ユーザー モード ドライバーへの影響はありません。

カーネルモード ドライバーの標準的な開発には、カーネル モード デバッグの使用またはブート構成データ (BCD) の <testsigning> 設定のいずれかが含まれます。セキュア ブートが有効な場合、これらの両方が無効化されます。

影響

カーネルモード ドライバーは、信頼された証明機関 (CA) によって正しく署名されていないと動作しません。オペレーティング システムは信頼できないドライバーの実行を許可しないため、カーネルのデバッグおよびテストといった標準のメカニズムを使用できなくなります。

軽減策

ドライバーをテストする場合は、BIOS でセキュア ブートを無効化すると同時に、その他のテスト署名要件 (以下を参照) を満たす必要があります。

より広範に配布するドライバーは、信頼された CA によって正しく署名されている必要があります。

参考資料

Windows のドライバー署名の要件
http://go.microsoft.com/fwlink/p/?LinkId=325488

メジャー ブート

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

マルウェア対策 (AM) ソフトウェアは、マルウェアをよりよく検知できるようになっていますが、一方で攻撃者による検知から逃れるルートキットの作成がより巧妙になっています。大部分の AM ベンダーはブート サイクルの早い段階で起動するマルウェアを検出することに、懸命に挑戦しています。一般的に、それらはホスト オペレーティング システムではサポートされていないシステム ハックを作成し、実際にコンピューターが不安定な状態に置かれる可能性があります。これまでに Windows は AM がこれらの初期のブート脅威を見つけて解決する良い方法を提供していませんでした。

Windows 8 には、メジャー ブートと呼ばれる新しい機能が導入されています。この新しい機能では、ファームウェアからブート スタート ドライバーを通じて各コンポーネントを測定し、マシン上の Trusted Platform Module (TPM) に測定値を保存してログを作成し、リモートでクライアントの起動状態をテストできるようにします。

影響

メジャー ブート機能は、AM ソフトウェアの前に開始したすべてのブート コンポーネントの信頼された (なりすましに対する耐性や改ざんの) ログを、AM ソフトウェアに提供します。AM ソフトウェアでは、このログを使用して、以前に動作したコンポーネントが信頼できるかどうか、それらがマルウェアに感染していないかどうかを確認することができます。ローカル マシンの AM ソフトウェアは、ログを評価するためにリモート サーバーに送信することができます。リモート サーバーは、必要に応じてクライアント上のソフトウェアと相互作用によって、または帯域外のメカニズムによって、修復アクションを開始することができます。

軽減策

企業シナリオでは、システム管理者はメジャー ブート情報がどのように使われるかをコントロールしています。エンドユーザー シナリオ (たとえばオンライン バンキング) では、消費者は特定のサービスのためにメジャー ブートの使用を選択する必要があります。

解決策

ホワイトペーパーでは、さまざまなメジャー ブートのシナリオを作成するための API と関数呼び出しについての詳細情報を準備しています。

スタートアップ アプリ

プラットフォーム

クライアント – Windows 8

説明

Windows で最も力を入れている点の 1 つが、従来のデスクトップおよびノート PC から、低消費電力型タブレット PC まで、さまざまなフォーム ファクターをカバーできるようにすることです。Windows を使用するどのフォーム ファクターを選択した場合でも、ユーザーと開発者の両方の優れたエクスペリエンスを確保するには、バッテリ寿命の延長および卓越した PC の応答性という 2 つの基準を満たすことが成功の鍵となります。Windows ではこれらの基準を満たすため、プロセスのライフ サイクル、スリープ状態、スタートアップ アプリ (マシンのブート後に自動で起動されるアプリ) など、複数の領域に関して改善が図られています。このトピックでは、スタートアップ アプリが Windows デバイスに与える影響のいくつかを紹介します。また、開発者 (ISV/IHV) および OEM がスタートアップ アプリの使用パターンを再考し、ユーザーおよび開発者に、優れたバッテリ寿命と応答性を提供するためのガイダンスを提供します。さらに、実際に実行するスタートアップ アプリをユーザーが決定できるようになった、Windows の変更点についても説明します。

Windows ストア アプリは、バッテリ消費と応答性の新標準を忠実に満たすように設計されており、アプリの中断または終了によって Windows がそのライフ サイクルを管理します。ただし、以前のバージョンの Windows 用に設計されたデスクトップ アプリは、必ずしもバッテリ寿命の延長や、ユーザーのアクティビティへの配慮がなされているわけではなく、システムの応答性に影響を及ぼす可能性があります (たとえば、更新確認のためにアプリが 1 秒周期でハートビートを送信する場合や、後から必要となる場合に備えてメモリを事前に割り当てる場合など)。長いバッテリ寿命や数週間持続するスタンバイを期待して Windows タブレット PC を購入したのに、タブレット PC のバッテリ寿命ではそれを達成できないとなると、ユーザーを失望させてしまいます。また、スタートアップ アプリはバックグラウンドで動作するため、ユーザーが認識しているよりもはるかに多くのアプリがシステム上で実行され、システムの応答性に影響を及ぼす場合もあります。スタートアップ アプリには、以下のメカニズムを活用して起動するものが含まれます。

- **Run** レジストリ キー (HKLM、HKCU、wow64 のノードを含む)
- **RunOnce** レジストリ キー
- ユーザーごとの場所と共有の場所にある、スタート メニュー配下のスタートアップ フォルダー

エンド ユーザーがシステム上で稼働するアプリを常に管理できるようにするため、Windows に新しい機能が追加されました。タスク マネージャーの [スタートアップ] タブには、スタートアップ アプリの一覧のほかに、スタートアップ アプリを無効化するためのコントロールが表示されます。

タスク マネージャーは、無効化対象をユーザーが判断しやすいよう、各スタートアップ アプリの影響度の目安を表示します。影響度は、スタートアップ時にアプリが使う CPU とディスクの量に基づいて評価されます。影響度値は、次の基準を適用して決定されます。

- **影響度高** –スタートアップ時に、1 秒間を超える CPU 時間または 3 MB を超えるディスク I/O を使うアプリ

- **影響度中** – 300 ～ 1,000 ミリ秒の CPU 時間または 300 KB ～ 3 MB のディスク I/O を使用するアプリ
- **影響度低** – 300 ミリ秒未満の CPU 時間および 300 KB 未満のディスク I/O を使用するアプリ

Microsoft では、アプリ開発者がアプリのスタートアップ時の影響を評価、分析し、影響の軽減措置を講じてユーザー エクスペリエンスを向上させるための各種ツールを提供しています。Windows アセスメント & デプロイメント キットは、起動パフォーマンスを評価し、スタートアップ時に実行されるアプリの影響度を測定する機能を備えています。評価結果には、Windows のスタートアップ時に大きな影響を及ぼすコンポーネントの詳しい分析に加え、可能な場合には修復方法が含まれます。Windows パフォーマンス アナライザーを使用すると、アプリ開発者が詳細な分析を実行し、パフォーマンスへの影響の根本原因を突き止めて、Windows のスタートアップ パフォーマンスを改善できます。Windows ADK のインストール方法については、こちらをご覧ください。

ガイダンス

下表に示すように、スタートアップ アプリは複数のカテゴリに分けられます。前述した Windows の機能変更に対応するための開発者向け推奨事項を、カテゴリ別に記載しています。

スタートアップ アプリのカテゴリ		説明	推奨事項
アップデーター		オンライン更新の有無を監視し、ユーザーに通知する	**メンテナンス タスク** 注: すべての更新は、UI による対話的操作を必要とせずに、メンテナンス タスクとして処理される必要があります。アプリによって自動的に更新処理が行われ、障害が起きた場合はロールバックされるようにします。
ハードウェアの補助	代替アクセス ポイント	Windows の他のアクセス ポイントからアクセス可能な、Windows の機能およびアプリへのアクセス手段を提供する	**削除** 注: Windows 内に存在する重複した機能を削減することが重要です。
	通知	ユーザーに使用中のデバイスに関する情報を通知する	**削除** 注: Windows はユーザーに対し、ユーザーが所有するデバイスについて Windows 8 通知を提供します。
予備ランチャー		アプリで必要な予備作業の一部が、ユーザー ログインの間にスタートアップ アプリにオフロードされる	**削除** 注: Windows 8 はアプリの高速な起動を実現するために最適化されています。
ユーティリティ	PC の同期	複数のシステム間の同期機能を提供する	**スタートアップ** (Beta で更新の見込み)
	バックアップと復元	ファイル、設定、またはシステム全体の状態を保存、復元するためのエントリ ポイント	**Windows ストア アプリをユーザーとのインターフェイスに使用**
	製品利用統計情報	ユーザー エクスペリエンスと環境の情報収集と送信を行う	**メンテナンス タスク**
	PC の監視	既存の標準機能と重複する、システム状態の監視および通知の機能をユーザーの求めなく提供する	**削除** 注: Windows 内に存在する重複した機能を削減することが重要です。

スタートアップ アプリのカテゴリ		説明	推奨事項
セキュリティ	保護者による制限とフィルター	インターネットへのアクセスと使用方法について、設定されたルールと制限を適用する	**スタートアップ**
	構成と管理	システムのセキュリティ監視について、診断と修復のオプションをユーザーが制御可能にする ユーザーに調査結果とセキュリティ対策を通知する	**Windows ストア アプリをユーザーとのインターフェイスに使用**
通信とインターネット (IM と VoIP)		メッセージと電話の発着信を行う	**Windows ストア アプリ**
音楽と MP3		音楽の再生、保存、管理	**Windows ストア アプリ**
写真とビデオ		写真とビデオの検出、記録、表示、保存、管理	**Windows ストア アプリ**
PC ゲーム		さまざまなドメインにわたってゲームを起動	**Windows ストア アプリ**
アップセルと広告		利用可能な商品とサービスの購入を勧める	**削除**

注: アクセシビリティ アプリのガイドライン については、別途 ISV から直接説明があります。詳しくは、「**コンピューターの簡単操作のためのプログラミング**」(http://msdn.microsoft.com/ja-jp/windows/bb879984.aspx) をご覧ください。

Windows ストア アプリ

Windows ストア アプリは、新たなアプリ モデル、新たなユーザー インターフェイス、Windows ストアから成る新たな組み合わせを Windows の世界に導入することで、高度なユーザー エクスペリエンスを実現します。Windows ストア アプリの開発では、以下の言語と表示フレームワークを選択できます。

- HTML/JavaScript/CSS
- XAML/C#
- XAML/C++

Windows ストア アプリの開発に関する総合的な情報については、Windows デベロッパー センターの http://dev.windows.com と http://go.microsoft.com/fwlink/?LinkId=221445 をご覧ください。

例:

Windows ストアのゲーム開発:
http://go.microsoft.com/fwlink/?LinkId=228452

メディアを使った Windows ストア アプリの開発:
http://go.microsoft.com/fwlink/?LinkId=228453

自動メンテナンス タスク

定期的なバックグラウンド アクティビティは、自動メンテナンス タスクとして設計する必要があります。こうしたタスクは、Windows PC の応答性とエネルギー効率の向上のため、システムのアイドル時間に実行するようにスケジューリングされます。メンテナンス タスクをデスクトップ アプリのイン

ストール時に作成および設定できるようにするには、デスクトップ SDK を使用します。詳細については、「自動メンテナンス」の項を参照してください。

参考資料

コンピューターの簡単操作のためのプログラミング
http://go.microsoft.com/fwlink/p/?LinkId=325489

Windows ADK
http://go.microsoft.com/fwlink/?LinkId=234980

Windows デベロッパー センター
http://go.microsoft.com/fwlink/?LinkId=221445

自動メンテナンス

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows の付加価値のほとんどは、Windows Update、ディスクの自動最適化、ウイルス対策の更新プログラム、ウイルス スキャンなど、標準およびサード パーティ製のメンテナンス アクティビティを実行することでもたらされています。さらに企業では、ネットワーク アクセス保護 (NAP) スキャニングなどのメンテナンス アクティビティを頻繁に利用し、社内の全ワークステーションにセキュリティ標準を適用するよう努めています。

Windows のメンテナンス アクティビティは、ユーザーによる操作をほとんど必要とせずにバックグラウンドで動作し、パフォーマンスおよびエネルギー効率には最小限の影響しか及ぼさないように設計されています。ただし、Windows 内の複数のメンテナンス アクティビティは、不確定で多岐にわたるスケジュール設定がされているために、Windows 7 以前のバージョンではパフォーマンスとエネルギー効率への影響が解消されていません。ユーザーがコンピューターを頻繁に利用している間にメンテナンス アクティビティを実行すると、ユーザーへの応答性が損なわれます。アプリもまた、頻繁にソフトウェアの更新を要求し、バックグラウンドでメンテナンスを実行するほか、アクション センター、コントロール パネル、Windows Update、タスク スケジューラの MMC スナップイン、サード パーティ製の制御機能といった、複数のエクスペリエンスにユーザーを誘導します。

自動メンテナンスの目的は、バックグラウンドで実行される Windows 内のすべてのメンテナンス アクティビティを統合し、パフォーマンスとエネルギー効率に悪影響を及ぼすことなく、サード パーティの開発者が自社製品のメンテナンス アクティビティを Windows に追加できるようにすることです。自動メンテナンスはさらに、ユーザーおよび企業による、メンテナンス アクティビティのスケジューリングおよび構成の管理を可能にします。

主な問題点

自動メンテナンスは、Windows 内のメンテナンス アクティビティに関連した以下の問題点に対応できます。

- 期限のスケジューリング
- リソースの使用上の競合
- エネルギー効率
- ユーザーに対する透明性

機能

自動メンテナンスは、アイドル時間の効率的利用を促進し、すべてのアクティビティを優先度に応じた適切なタイミングで実行します。また、メンテナンス アクティビティの一元的な監視と管理を可能にし、パフォーマンスやエネルギー効率に悪影響を及ぼすことなく、サード パーティの開発者が自社製品のメンテナンス アクティビティを Windows に追加できるようにします。自動メンテナンス

は上記を達成するため、完全自動モード、ユーザー開始モード、自動停止、期限と通知、企業管理に対応しています。それぞれについて以下で説明します。

完全自動モード

PC のアイドル時間と指定済みの時間にメンテナンス アクティビティの実行と一時停止を高度にスケジューリング可能な既定のモードで、ユーザーによる操作は一切必要ありません。ユーザーは週次または日次のスケジュールを設定できます。すべてのメンテナンス アクティビティは、操作が不要で自動的に実行されます。

システムが使用されていないと思われる場合には、コンピューターのスリープが自動的に解除されます。ただし、ノート PC の場合は電源管理ポリシーの既定値が優先され、AC 電源に接続されている場合のみスリープが解除されます。すべてのシステム リソースを最大限に活用し、最短の時間でメンテナンス アクティビティを完了します。自動メンテナンスのためにシステムのスリープが解除された場合は、スリープ状態への復帰が要求されます。

設定などのアクティビティに関してユーザーの操作が必要となる場合は、自動メンテナンスの実施とは別に行います。

ユーザー開始モード

ユーザーが出張の準備として、バッテリの電源を長持ちさせたい場合や、パフォーマンスと応答性の最適化を図る場合には、自動メンテナンスを手動で開始することもできます。ユーザーは自動実行スケジュールなど、自動メンテナンスの属性設定を変更できます。また、ユーザーは自動メンテナンスの現在の実行ステータスを確認するほか、必要に応じて自動メンテナンスを停止することも可能です。

自動停止

ユーザーがコンピューターの操作を開始すると、自動メンテナンスは、そのとき実行しているメンテナンス アクティビティを自動的に停止させます。メンテナンス アクティビティは、システムがアイドル状態に戻ってから再開されます。

注: 自動メンテナンスのアクティビティはすべて、2 分以内の停止が可能である必要があります。アクティビティを停止した場合は、ユーザーへの通知が必要です。

期限と通知

重要なメンテナンス アクティビティは、あらかじめ決められた時間枠内で実行する必要があります。指定した時間内に重要なタスクを実行できなかった場合、自動メンテナンスは、次にシステムがアイドル状態になった機会に、自動的にタスクの実行を開始します。ただし、タスクの期限が超過した状態のままにしておくと、自動メンテナンスはユーザーにそのアクティビティについて通知し、自動メンテナンスを手動で実行するかどうかを確認します。メンテナンスのためにスケジューリングされたタスクはすべて実行されますが、最も負荷の高いタスクが優先的に処理されます。このアクティビティはシステムの応答性とパフォーマンスに影響を及ぼすことがあるため、自動メンテナンスは、重要なメンテナンス アクティビティを実行中であることをユーザーに通知します。

企業における管理

企業の IT 担当者は、社内の Windows システム上でいつ自動メンテナンスを実行するかについて決定し、標準化された管理インターフェイスを通じてそのスケジュールを適用すると共に、実行を試みた自動メンテナンスのステータスに関するイベント データを取得できなくてはなりません。さらに、IT 担当者は標準の管理インターフェイスを介して、自動メンテナンスの特定のアクティビティをリモートから呼び出せなければなりません。自動メンテナンスの実行時には、毎回進捗状況が報

告されます。たとえば、ユーザーが手動でアクティビティを停止したため、自動メンテナンスを実行できなかったことなどが通知されます。IT 担当者は、ユーザーのログオン時間を短縮するため、ログオン スクリプトを自動メンテナンスに移行することを検討する必要があります。

自動メンテナンス タスクの作成

以降では、XML または C 言語によるタスク定義を使って、開発者がタスクを作成する方法を詳しく説明します。注意点として、自動メンテナンスは完全に自動化され、ユーザーがいないときに実行されるため、メンテナンス アクティビティでは、ユーザーの操作を必要とするユーザー インターフェイスを起動するべきではありません。実際には、自動メンテナンスの実行中にユーザーがコンピューターを操作すると、進行中の全タスクが中断され、次のアイドル時間まで保留されます。

XML の使用

タスク スケジューラに含まれる、ビルトインのコマンドライン ツール (schtasks.exe) を使用すると、XML 形式のタスク定義をインポートできます。タスク定義のスキーマについては、http://msdn.microsoft.com/library/aa383609(v=VS.85).aspx. で説明しています。次に、XML で定義された自動メンテナンス タスクの例を示します。

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.4" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>2011-07-01T11:34:31</Date>
    <Author>IT Deptartment</Author>
  </RegistrationInfo>
  <Principals>
    <Principal id="Author">
      <RunLevel>LeastPrivilege</RunLevel>
      <GroupId>NT AUTHORITY\SYSTEM</GroupId>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>false</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <MaintenanceSettings>
      <Period>P2D</Period>
      <Deadline>P14D</Deadline>
    </MaintenanceSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <DisallowStartOnRemoteAppSession>false</DisallowStartOnRemoteAppSession>
    <UseUnifiedSchedulingEngine>true</UseUnifiedSchedulingEngine>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>P3D</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>cmd</Command>
      <Arguments>/c timeout -t 60</Arguments>
    </Exec>
  </Actions>
</Task>
```

タスクを Windows コンピューターに保存するには、上記の XML をテキスト ファイルとして保存し、次のコマンド ラインを実行します。

```
Schtasks.exe /create /tn <task name> /xml <text file name>
```

C 言語の使用

自動メンテナンス タスクは C 言語を使って作成することもできます。タスクの自動メンテナンスの設定値を構成する際に利用できるコード サンプルを次に示します。

```
/*****************************************************************
This sample creates a maintenance task to start cmd window during maintenance
opportunities with periodicity of 2 days and deadline Of 14 days.
*****************************************************************/

#define _WIN32_DCOM

#include <windows.h>
#include <iostream>
#include <stdio.h>
#include <comdef.h>
#include <wincred.h>
//  Include the task header file.
#include <taskschd.h>
//#pragma comment(lib, "taskschd.lib")
//#pragma comment(lib, "comsupp.lib")

int __cdecl
MainteanceTask( )
{
    //  ------------------------------------------------------
    //  Initialize COM.
    HRESULT hr;

    //  ------------------------------------------------------
    //  Create a name for the task.
    LPCWSTR wszTaskName = L"MaintenanceTask";

    ITaskService *pService = NULL;
    ITaskFolder *pRootFolder = NULL;
    ITaskDefinition *pTask = NULL;
    ITaskSettings *pSettings = NULL;
    IRegistrationInfo *pRegInfo= NULL;
    IPrincipal *pPrincipal = NULL;
    ITaskSettings3 *pSettings3 = NULL;
    IMaintenanceSettings* pMaintenanceSettings = NULL;
    IActionCollection *pActionCollection = NULL;
    IAction *pAction = NULL;
    IExecAction *pExecAction = NULL;
    IRegisteredTask *pRegisteredTask = NULL;

    wprintf(L"\nCreate Maintenance Task %ws", wszTaskName );

    hr = CoInitializeEx( NULL, COINIT_MULTITHREADED);
    if( FAILED(hr) )
    {
        wprintf(L"\nCoInitializeEx failed: %x", hr );
        return 1;
    }

    //  Set general COM security levels.
    hr = CoInitializeSecurity( NULL,
```

```
        -1,
        NULL,
        NULL,
        RPC_C_AUTHN_LEVEL_PKT_PRIVACY,
        RPC_C_IMP_LEVEL_IMPERSONATE,
        NULL,
        0,
        NULL);

    if( FAILED(hr) )
    {
        wprintf(L"\nCoInitializeSecurity failed: %x", hr );
        goto CleanUp;
    }

    //  ------------------------------------------------------
    //  Create an instance of the Task Service.
    hr = CoCreateInstance( CLSID_TaskScheduler,
                           NULL,
                           CLSCTX_INPROC_SERVER,
                           IID_ITaskService,
                           (void**)&pService );
    if (FAILED(hr))
    {
        wprintf(L"\nFailed to create an instance of ITaskService: %x", hr);
        goto CleanUp;
    }

    //  Connect to the task service.
    hr = pService->Connect(_variant_t(), _variant_t(), _variant_t(),
_variant_t());
    if( FAILED(hr) )
    {
        wprintf(L"\nITaskService::Connect failed: %x", hr );
        goto CleanUp;
    }

    //  ------------------------------------------------------
    //  Get the pointer to the root task folder.  This folder will hold the
    //  new task that is registered.
    hr = pService->GetFolder( _bstr_t( L"\\") , &pRootFolder );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot get Root folder pointer: %x", hr );
        goto CleanUp;
    }

    //  If the same task exists, remove it.
    ( void ) pRootFolder->DeleteTask( _bstr_t(wszTaskName), 0  );

    //  Create the task definition object to create the task.
    hr = pService->NewTask( 0, &pTask );
    if (FAILED(hr))
    {
        wprintf(L"\nFailed to CoCreate an instance of the TaskService class: %x",
hr);
        goto CleanUp;
    }

    //  ------------------------------------------------------
    //  Get the registration info for setting the identification.
    hr = pTask->get_RegistrationInfo( &pRegInfo );
```

```
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot get identification pointer: %x", hr );
        goto CleanUp;
    }

    hr = pRegInfo->put_Author( _bstr_t(L"Author Name") );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot put identification info: %x", hr );
        goto CleanUp;
    }

    // The task needs to grant explicit FRFX to LOCAL SERVICE (A;;FRFX;;;LS)
    hr = pRegInfo-
>put_SecurityDescriptor( _variant_t(L"D:P(A;;FA;;;BA)(A;;FA;;;SY)(A;;FRFX;;;LS)")
);
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot put security descriptor: %x", hr );
        goto CleanUp;
    }

    //  ----------------------------------------------------
    //  Create the principal for the task - these credentials
    //  are overwritten with the credentials passed to RegisterTaskDefinition
    hr = pTask->get_Principal( &pPrincipal );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot get principal pointer: %x", hr );
        goto CleanUp;
    }

    //  Set up principal logon type to interactive logon
    hr = pPrincipal->put_LogonType( TASK_LOGON_INTERACTIVE_TOKEN );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot put principal info: %x", hr );
        goto CleanUp;
    }

    //  ----------------------------------------------------
    //  Create the settings for the task
    hr = pTask->get_Settings( &pSettings );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot get settings pointer: %x", hr );
        goto CleanUp;
    }

    hr = pSettings->QueryInterface( __uuidof(ITaskSettings3), (void**)
&pSettings3 );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot query ITaskSettings3 interface: %x", hr );
        goto CleanUp;
    }

    hr = pSettings3->put_UseUnifiedSchedulingEngine( VARIANT_TRUE );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot put_UseUnifiedSchedulingEngine: %x", hr );
```

```
        goto CleanUp;
    }

    hr = pSettings3->CreateMaintenanceSettings( &pMaintenanceSettings );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot CreateMaintenanceSettings: %x", hr );
        goto CleanUp;
    }

    hr = pMaintenanceSettings->put_Period ( _bstr_t(L"P2D") );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot put_Period: %x", hr );
        goto CleanUp;
    }

    hr = pMaintenanceSettings->put_Deadline ( _bstr_t(L"P14D") );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot put_Period: %x", hr );
        goto CleanUp;
    }

    //  ------------------------------------------------------
    //  Add an action to the task. This task will execute notepad.exe.
    //  Get the task action collection pointer.
    hr = pTask->get_Actions( &pActionCollection );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot get Task collection pointer: %x", hr );
        goto CleanUp;
    }

    //  Create the action, specifying that it is an executable action.
    hr = pActionCollection->Create( TASK_ACTION_EXEC, &pAction );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot create the action: %x", hr );
        goto CleanUp;
    }

    //  QI for the executable task pointer.
    hr = pAction->QueryInterface( IID_IExecAction, (void**) &pExecAction );
    if( FAILED(hr) )
    {
        wprintf(L"\nQueryInterface call failed for IExecAction: %x", hr );
        goto CleanUp;
    }

    //  Set the path of the executable to notepad.exe.
    hr = pExecAction->put_Path( _bstr_t(L"cmd") );
    if( FAILED(hr) )
    {
        wprintf(L"\nCannot put action path: %x", hr );
        goto CleanUp;
    }

    //  ------------------------------------------------------
    //  Save the task in the root folder.
    hr = pRootFolder->RegisterTaskDefinition(
            _bstr_t(wszTaskName),
```

```
            pTask,
            TASK_CREATE_OR_UPDATE,
            _variant_t(),
            _variant_t(),
            TASK_LOGON_INTERACTIVE_TOKEN,
            _variant_t(L""),
            &pRegisteredTask);
    if( FAILED(hr) )
    {
        wprintf(L"\nError saving the Task : %x", hr );
        goto CleanUp;
    }

    wprintf(L"\nSuccess!\n----------------------------------" );

CleanUp:

    if ( pService != NULL ) pService->Release();
    if ( pRootFolder != NULL ) pRootFolder->Release();
    if ( pTask != NULL ) pTask->Release();
    if ( pSettings != NULL ) pSettings->Release();
    if ( pRegInfo != NULL ) pRegInfo->Release();
    if ( pPrincipal != NULL ) pPrincipal->Release();
    if ( pSettings3 != NULL ) pSettings3->Release();
    if ( pMaintenanceSettings != NULL ) pMaintenanceSettings->Release();
    if ( pActionCollection != NULL ) pActionCollection->Release();
    if ( pAction != NULL ) pAction->Release();
    if ( pExecAction != NULL ) pExecAction->Release();
    if ( pRegisteredTask != NULL ) pRegisteredTask->Release();

    CoUninitialize();
    return SUCCEEDED ( hr ) ? 0 : 1;
}
```

検証タスク

タスクが正しく作成され、メンテナンスの一部として実行されることを検証します。

タスク作成の検証

次のコマンド ラインを使い、タスク定義をファイルにエクスポートして、タスク定義が目的に沿っていることを確認します。

```
Schtasks.exe /Query /tn<task name> /xml <text file name>
```

タスクの実行検証

次のコマンド ラインを実行してタスクを起動し、タスク スケジューラ (taskschd.msc) の UI に、タスクを実行したことが表示されるのを確認します。

```
Schtasks.exe /Run /tn<task name>
```

参考資料

タスク スケジューラ

http://go.microsoft.com/fwlink/p/?LinkId=325490

https://msdn.microsoft.com/en-us/library/bb756903.aspx

サード パーティ製の入力方式エディター (IME)

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

入力方式エディター (IME) は、キーボードで表現することができる文字よりも多くの文字を持つ言語のテキストの入力を可能にするソフトウェア コンポーネントです(これは一般的には東アジア言語ですが、この限りではありません)。 1 つのキーで表現される 1 つの文字の代わりに、ユーザーは IME によって解釈されるキーの組み合わせを入力します。IME は、ユーザーに選択可能な文字のリストを提示し、ユーザーのアプリの編集コントロール ウィンドウに、キー ストロークのセットと一致した文字を挿入する文字を生成します。

過去に Windows は サード パーティ製の IME が Windows システムで実行されるのを許可しており、Windows 8 でもこの機能を引き継いでいます。ユーザーはサード パーティ製の IME をインストールして使用することができます。さらに悪意のある IME を防止するためにシステムとプロセスを強化してセキュリティを向上し、ユーザー エクスペリエンスを強化しています。

Windows 8 では、次の点にご注意ください。

- ハードウェア キーボードとタッチ キーボード用に、サード パーティ製の IME をサポートします。
- サード パーティの IME ベンダーは、Windows 8 に対応した IME を開発するために、Microsoft のガイドラインに従う必要があります。
- サード パーティ製の IME はデジタル署名を行う必要があります。
- サード パーティ製の IME はテキスト サービス フレームワーク (TSF) に準拠し、Windows 8 で正しく実行されるために適切な IME フラグを設定する必要があります。
- 従来のサード パーティ製の IME はデスクトップ アプリで実行できますが、Windows ストア アプリでは拒否されます。
- ユーザーがタッチ キーボードで IME を使えるように、サード パーティ製の IME では Windows によって提供されるタッチ キーボード レイアウトを使って、IME にリンクすることができます。ただし、サード パーティ製の IME には、タッチ キーボード用の付属の IME において特定の機能が提供されません。
- Windows Defender は、Windows システムから悪意のある IME を取り除きます。

影響

入力言語と入力方式の変更を切り替える

すべての IME モード アイコンを IME 商標アイコンと共に表示する代わりに、IME 商標アイコンと 1 つの IME モード アイコンが表示されます。IME **商標**アイコンをクリックすると、入力方式を切り替え

ることができます。IME **モード** アイコンをクリックすると、別の IME モードに切り替えることができます。

日本語の IME が現在の入力方法として表示された Windows 8 の入力ポップアップと IME ポップアップ

IME が Windows 7 上で IME モード アイコンを表示する言語バーに依存している場合は、IME 商標アイコンと IME モード アイコンを Windows 8 上 の入力インジケーターで表示するための変更を行う必要があります。

注: IME がデスクトップ タスク バーのシステム トレイで IME 商標アイコンと IME モード アイコンを表示する方法について詳しくは、Windows 8 IME ガイドラインで文書化されて掲示される予定です。

新しい Windows 環境

Windows 8 の環境では、IME の見た目が変わります。Windows ストア アプリ、ローカル コンテキストのアプリ コンテナー、IME の API 制限の概念は、Windows 7 では提示されていませんでした。いくつかの既存の Windows 7 の IME は、Windows ストア アプリ内で動作するときに応答を停止するため、Windows ストア アプリで動作するレガシ IME を許可しません。これは、新しいバージョンの IME が Windows ストア アプリ内で実行される前に、新しい UI 環境との互換性確保について確認する必要があることを示します。

軽減策

デスクトップ互換の IME をシステムで使うことができます。主にデスクトップ アプリを使う場合や、レガシ IME を優先して使い続ける場合は、これは最良の選択肢となるでしょう。この限りでない場合には Windows 8 IME を使って、レガシ / 非認定の IME の使用をやめることをお勧めします。デスクトップ互換の IME を使う影響を警告するための通知機能は言語 CPL と入力スイッチによって提供されます。

デスクトップ IME がシステムで動作しない場合は、次の事象が発生します。

- 言語 CPL UI は、デスクトップ互換の IME にラベルをつけて、デスクトップ互換の IME がデスクトップアプリだけで動作するという旨のメッセージを表示します。
- ユーザーが Windows ストア アプリを使っているとき、入力フライアウトはデスクトップ互換の IME をグレー アウトします。これは IME がこのアプリで動作しないことを示しています (デスクトップでは、デスクトップ互換の IME はグレーアウトされません)。デスクトップ互換の IME と共に Windows ストア アプリに切り替えて IME をオフにする場合は、Windows ストア アプリと互換性のある IME に変更するために入力インジケーターを使います。

レガシまたはデスクトップ互換の IME の使用は、以下の状況に限られています。

- システム上でサード パーティ製の IME とともに、Windows 7 から Windows 8 へアップグレードします。
- ベンダーは Windows 8 と互換性のない IME に対応しておらず、ユーザーはその間に既存の Windows 7 版を使おうとします。

解決策

全般

IME ロジックおよび UI のための Windows ストア アプリの共通コントロールを実装するために、既存のテキスト サービス フレームワーク (TSF) インフラストラクチャを使ってください。UI をホストするための Owned Window を作成してください。

検索予測を改善し、UI でより無駄のない検索エクスペリエンスを提供するために、新しい検索 API が追加されています。

UI がタッチ キーボードで隠れてしまうのを保護するためにタッチ キーボードが起動されたことをサード パーティ製の IME に通知する API も追加されています。既定のクラシック タッチ キーボード レイアウトは、自動的にサード パーティ製の IME 用に読み込まれます。このクラシック タッチ キーボード レイアウトを統合するために追加の作業は必要ありません。ただし、サード パーティ製の IME は、代替のタッチ レイアウトを要求することもできます。

IME において Windows ストア ユーザー エクスペリエンスの主要原則を促進することができるように、Windows 8 の IME ガイドラインに精通してください。ガイドラインを厳守している IME は、IME が Microsoft のデザインと互換性を持つことを示すためのフラグをセットしなければなりません。Windows ストア アプリでは、デスクトップ IME が実行できないようにすべてブロックされます。

デジタル署名は、悪意のある IME が Windows Defender によって取り除かれることに加えて、Windows 8 システム上にインストールされるのを防ぎます。本人確認時に、サード パーティ製の IME の dll がデジタル署名されます。このデジタル署名がある IME だけを重要な警告メッセージをユーザーに表示することなくシステム上にインストールすることができます。ユーザーは、悪意のある IME を報告することができます。悪意がある IME と判断されると、Windows Defender は、その IME を Windows システム上から削除します。

Text Services Framework

IME は、Windows 8 上で実行できるようにするために TSF に対応する必要があります。Windows 8 は、TSF 非対応の IME が Windows ストア アプリで実行されるのをブロックします。アプリが開始されると、TSF は、ユーザーがアプリ処理中に選択した IME の IME.dll を読み込みます。

注: Windows ストア アプリとデスクトップ アプリの間で別々の機能または UI を提供するために、TSF によって読み込まれる dll は、それがどの種類のアプリによってロードされているのかを確認することができます。IME は ITfThreadMgrEx::GetActiveFlags メソッドを呼び出し、TF_TMF_IMMERSIVEMODE フラグをチェックして、結果に応じて異なるアプリ ロジックをトリガーすることができます。

IME を Windows ストア アプリに読み込むときは、アプリ自体と同じアプリ コンテナーの制限が適用されます。この動作は、IME がデスクトップ SDK にアクセスできるにもかかわらず、(Windows ストアで配布または認定されないため) Windows ストア アプリのセキュリティ契約に違反できないことを保証します。IME が現在実行するいくつかの機能は、アプリ コンテナーの内部で影響を受けます。それらの機能には以下が含まれます。

- 辞書ファイル
- インターネット更新
- オンザフライの学習
- プロセス間での情報共有

詳しくは、Windows 8 IME ガイドラインをご覧ください。

システム停止を含む悪いユーザー エクスペリエンスの可能性を避けるために、レガシ IME は、Windows ストア アプリでは動作しません。Windows ストア アプリと互換性を持つ IME は、フラグをセットすることで、この互換性を示していることを自己宣言しなければなりません。このフラグは、TSF によって TF_INPUTPROCESSORPROFILE 構造体の中で提供されます。このフラグを使って、Windows ストア アプリ互換としてサード パーティ製の IME を宣言する方法の詳細を文書化した Windows 8 IME ガイドラインが公開されています。

Windows ストア アプリと互換性を持つ IME は、デスクトップ アプリと Windows ストア アプリのどちらでも動作します。互換性を持たない IME は、デスクトップ プロセス内のみで動作します。

ユーザー インターフェイス

サード パーティ製の IME は desktop windowing API にアクセスしますが、それらが動作しているアプリと同じ Windows API 制限に従わなければなりません。たとえば、IME はデスクトップ アプリ内でアクティブな間、Windows ストア アプリ上には表示できません。API 制限は、以下のシナリオを防ぐための対象となります。

- Windows ストア アプリからフォーカスを取得しているデスクトップ アプリ
- Windows ストア アプリで描画しているデスクトップ アプリ
- Windows ストア アプリに干渉しているデスクトップ アプリ

タッチ キーボードのサポート

サード パーティ製の IME ベンダーがタッチ キーボード (TKB) サポートを利用できる間は、Windows 8 ではカスタマイズ可能で統合されたタッチ キーボード エクスペリエンスは完全には提供されません。ただし、サード パーティ製の IME は タッチ キーボードのために、最適化されたキーボード レイアウトで IME をマッピングできます。Windows Soft Input Panel (SIP) は、サード パーティ製の IME 用のデフォルトとしてクラシック キーボード レイアウトを提供します。ハードウェア キーボードと同様の方法でクラシック キーボードが重要なイベントを生成しているため、現在タッチ キーボードで動作するサード パーティ製の IME の特別な実行要件はありません。ハードウェア キー イベントのための入力処理は、クラシック タッチ レイアウトからの重要なイベントも処理します。

注: IME は TKB サポートなどの最適化されたキーボード レイアウトを含むように拡張された場合、Unicode 入力イベントを処理する必要があるでしょう。

サード パーティ製の IME は、それらの IME のために最適化されたキーボード レイアウトを使うことができます。詳しくは、サード パーティ製の IME ガイドラインをご覧ください。

候補ウィンドウ の UI (と他の UI 要素) は、タッチ キーボードの下には描画されないことを確認してください。ほとんどの場合、アプリはタッチ キーボードを考慮してウィンドウの大きさを変更しなければなりません。しかし、アプリがこのような動きをしない場合は、IME はタッチ キーボードの位置を取得するための InputPaneFramework API を使用することができます。サード パーティ製の IME は、候補 (あるいはその他) の UI を描画する前に、タッチ キーボードによって画面スペースを確保するためにこの API を使うことができ、それらの UI がタッチ キーボードの下に描画されないように調整することができます。

検索

Windows 8 では、Windows ストア アプリにおいて検索コントラクトを実装して検索ペインと統合することで、ユーザーに検索機能を簡単に提供することができます。検索ペインは、すべてのアプリで検索を行うユーザーのために中央に配置されています。Windows は、検索ペインを使用しているアプリで、ユーザーができるだけ早く検索ペインに辿り着けるようにしています。特に、より高い効率とユーザビリティのために Windows 8 と互換性のある IME を統合することで、IME ユーザーに独自の検索エクスペリエンスを提供します。

IME は、次の条件を満たす場合に、統合された検索エクスペリエンスと連携できます。

- Windows ストア アプリ環境と互換性がある
- TSF UILess mode API を実装している
- TSF search integration API を実装している
    - ItfSearchCandidateProvider
    - ItfSearchHardwareKeyboardBehaviors

検索ペイン内でアクティブになった互換性のある IME は、UI レス モードに置かれ、その UI を表示することはできません。代わりに、Windows に変換候補を送信します。その後、それらの変換候補はインラインの候補リスト コントロールに表示されます。

IME は、現在の検索の実行において使う必要のある候補を Windows に送信します。これらの候補は、変換候補と同じにするか、あるいは検索に合わせて調整することができます。検索に適した候補の条件を次に示します。

- プレフィックスが重複していない
- 予測候補がない (完成形のみ)

基準を満たさず、検索と互換性がない IME は、他の Windows ストア アプリコントロールと同じように表示され、UI の統合と検索候補を活用することができません (ユーザーが文章を構成し終わった後でしか、アプリはクエリを受け取ることができません)。

検索コントラクトをサポートするアプリがクエリを受け取る際には、クエリ イベントには、最も関連が多い (関連がありそうな) ものから最も関連が少ない (関連がありそうもない) ものまでがランク付けされたすべての既知の選択肢を含む "queryTextAlternatives" 配列が含まれます。選択肢が提供

されるたびに、アプリはクエリと同じように各選択肢を扱わなければなりません。そして、基本的には "or" のクエリを、結果を提供するサービスに送り、(ユーザーが多数のクエリを同時に出したかのように) 選択肢のいずれかに一致する結果を返さなければなりません。パフォーマンスを強化するために、アプリは多くの場合、10 個の最も関連性の高い選択肢に一致するように制限を設けます。

IME デジタル署名

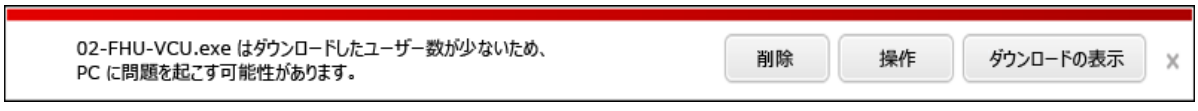
すべてのサード パーティ製の IME は、IME として Windows 8 システム上にインストールされるためにデジタル署名が必要とされます。SmartScreen を使用して、ユーザーが Web から署名のない IME をダウンロードする際には警告メッセージが表示されます。証明書を取得して署名する方法は以下のとおりです。

- **プログラムにデジタル署名するために Authenticode 署名を使用します。**
    - Windows によってサポートされている多くの証明機関の 1 つから、有効な Authenticode コード署名証明書を取得してください。
    - 配布する前にアプリを登録するための開発ツール (signtool.exe など) を使ってください。
    - 詳しい情報と署名処理の段階的な説明については、ブログ記事「Everything you need to know about Authenticode code signing (Authenticode コード署名について知っておくべきこと)」をご覧ください。
- **ダウンロードがマルウェアとして検出されないことを確認します。**
    - マルウェアとして検出され、確認されたプログラムは、ダウンロードの評価およびファイルに署名するのに使用されるデジタル証明書の評価に影響を及ぼします。
- **Windows 証明書を申請します。**
    - MSDN の Windows アプリの認定に関するページをご覧ください。

デジタル署名とコード署名について詳しくは、以下の記事をご覧ください。

- Authenticode の概要
- 整合性と信頼性の確保
- コード署名のベスト プラクティス
- デジタル証明書とは

IME が**署名されていない**場合、ユーザーが IME をダウンロードしようとすると以下の警告メッセージが表示されます。



IME が**署名されている**場合、代わりに以下のメッセージが表示されます。

これらの通知に従って、ユーザーはファイルを削除するのか、警告を無視してダウンロードされたプログラムを実行するのかを選択することができます。

IME の取り消し

悪意がある、あるいは Windows 8 IME ガイドラインに従っていない IME は、Windows Defender を使ってシステムから削除することができます。悪意のある IME について詳しくは、「Windows 8 のサード パーティ製の IME」をご覧ください。

参考資料

ITfThreadMgrEx::Get Active Flags メソッド
http://go.microsoft.com/fwlink/p/?LinkId=325492

SignTool
http://go.microsoft.com/fwlink/p/?LinkId=325493

Authenticode コード署名について知っておくべきこと
http://go.microsoft.com/fwlink/p/?LinkId=325494

Windows アプリのコントラクトと拡張
http://go.microsoft.com/fwlink/p/?LinkId=325495

Windows 8 デスクトップ アプリの認定要件
http://go.microsoft.com/fwlink/p/?LinkId=325496

Windows アプリの認定要件
https://msdn.microsoft.com/ja-jp/library/windows/apps/dn764944.aspx

Windows アプリ認定キットを使用する
https://msdn.microsoft.com/windows/uwp/debug-test-perf/windows-app-certification-kit-tests

Authenticode の概要
http://go.microsoft.com/fwlink/p/?LinkId=325498

整合性と信頼性の確保
http://go.microsoft.com/fwlink/p/?LinkId=325499

コード署名のベスト プラクティス
http://go.microsoft.com/fwlink/p/?LinkId=325500

デジタル証明書とは
http://go.microsoft.com/fwlink/p/?LinkId=325501

アプリによる記憶域メディアへの "TRIM および UnMap" ヒントの送信を実現する新しい API

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

TRIM ヒントは、以前に割り当てられた特定のセクターが、アプリで必要なくなったために回収可能であることをドライブに通知します。これは通常、アプリがファイルを介して大量のスペースを割り当て、そのファイル (たとえば、仮想ハード ディスク ファイルなど) への割り当てを自己管理する場合に使用される方法です。

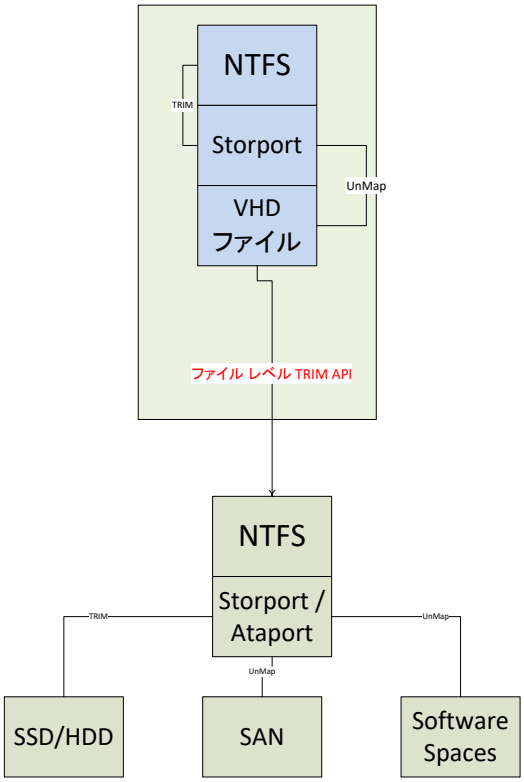
TRIM とは

ソリッド ステート ドライブ (SSD) は通常、フラッシュ メモリ ベースのブロック消去デバイスです。つまり、SSD に書き込まれたデータは直接上書きできず、そのブロックがガベージ コレクションの対象になるまで、別の場所に書き込んでおく必要があります。SSD は、特定のブロックが削除され、別のブロックが必要とされていることを判断するための内部メカニズムを備えていません。SSD はセクターが上書きされた場合のみ、そのセクターを "ダーティ" としてマークできます。その他 (ファイルの削除など) の場合、SSD は対象のセクターを保持します。これは、削除処理がマスター ファイル テーブル (MFT) のみの変更として実行され、ファイルの全セクターに対する処理とは見なされないためです。Windows 7 には、不要になったセクターに関する情報を SSD とやり取りするための標準の手段が導入されています。このコマンドは、T13 仕様 (http://go.microsoft.com/fwlink/p/?LinkId=325502) で TRIM コマンドとして定義されています。NTFS は "deletefile" など、一部の通常のインライン処理のためにこの TRIM コマンドを送信します。

記憶域に関連した TRIM のその他の利用方法

記憶域ネットワーク (SAN)、および Windows 8 の新機能である Software Spaces の実装では、SSD と同様に、TRIM コマンドのヒントを使用して仮想プロビジョニング環境のスペースを管理します。SAN および Software Spaces は、記憶域の領域を、セクターまたはクラスターよりも大きいサイズ (1 MB ～ 1 GB の間) で割り当てます。割り当てた (または割り当てサイズよりも大きい) サイズについて TRIM ヒントを受信した場合、SAN や SSD は領域の割り当てを解除し、別のファイルのスペースとして再利用できます。一般に、すべての TRIM ヒントを下層メディア (SSD または HDD) にパス スルーすることで、随時空いたスペースを使用できるようになります。通常、領域の割り当てを解除する場合、データを移動することはなく、TRIM 領域を追跡することもありません (その領域が空の場合)。

仮想プロビジョニングされた SAN では、渡された TRIM ヒントを使用して全体的な物理記憶域の容量を削減し、コスト削減に活用します。SCSI (T10: http://go.microsoft.com/fwlink/p/?LinkId=325503) 仕様には、"UnMap" コマンド (TRIM コマンドに類似) が定義されています。このコマンドは、HDD や SSD など、すべての種類の記憶域に適用可能です。UnMap コマンドは、SAN の割り当てから物理ブロックを削除するために使われます。



新しい API の使用方法

```
#define FSCTL_FILE_LEVEL_TRIM CTL_CODE(FILE_DEVICE_FILE_SYSTEM, 130,
METHOD_BUFFERED, FILE_WRITE_DATA)

Where
typedef struct _EXTENT_PAIR {
    ULONGLONG Offset;
    ULONGLONG Length;
} EXTENT_PAIR, *PEXTENT_PAIR;

typedef struct _FILE_LEVEL_TRIM {
    //
    // A count of how many Offset:Length pairs are given
    //
    DWORD PairCount;
    //
    // All the pairs.
    //
    EXTENT_PAIR Pairs[1];
} FILE_LEVEL_TRIM, *PFILE_LEVEL_TRIM;
```

File TRIM は、可能な場合はバッファーを介して、それ以外の場合は非同期で (バッファーを介さずに)、Device IOCTL DSM コマンドの TRIM に渡されます。これは、ATA デバイスでは TRIM コマンドに、SCSI デバイスでは UnMap コマンドにマッピングされます。File TRIM コードは、上記のエクステントによって指定された領域を 1 つずつ送信します。

File TRIM はデバイスからの応答を待ちません。これは、TRIM コマンドおよび Unmap コマンドが下層記憶域メディアに対するヒントとして定義されており、リターン コードが想定されていないためです。

エンド ツー エンドのワークフロー

1. File TRIM を呼び出す
    a. File TRIM が入力値にエラーがないか検証する
    b. File TRIM がエクステントを LCN 領域に分割する
    c. TRIM に渡すため、File TRIM が切り上げおよび切り捨てを実行して領域を調整する
    d. File TRIM が TRIM 領域に関連するキャッシュからエントリを削除する
    e. File TRIM が領域ごとに IOCTL_DSM (TRIM) を渡す
2. File TRIM が結果またはエラーを返す
    a. エラー チェックを実行し、入力の有効性を調べる
    b. エクステントの一部のみが有効な場合、API の呼び出し全体に対してエラーが 1 回返される

ユース ケース

SSD 上にマウントされたコンシューマー仮想ハード ディスク (VHD)

VHD は当初、未使用の "クリーンな" メディアにマウントされます。VHD を使用する間、VHD は記憶域メディアを部分的に消費してファイルなどを格納します。記憶域メディア内のファイルを削除しても、対象のファイルは SSD から取り除かれません。これは、VHD 全体が、1 つのファイルとして SSD 上に格納されているためです。VHD 環境でファイルが削除されると、Hyper-V 環境によって、削除された全領域に対する File TRIM が呼び出されます。こうした File_TRIM を SSD に伝達することで、SSD のパフォーマンスを最適化できます。

仮想プロビジョニングの SAN にマウントされたコンシューマー VHD

VHD は当初、仮想プロビジョニング環境の最小スラブの 1 つにマウントされます。VHD にファイルを格納するにつれて、VHD で占有する記憶域が複数のスラブに拡大します。VHD でファイルを削除すると、Hyper-V によって、仮想プロビジョニングされた下層の SAN に対する File_TRIM が呼び出されます。TRIM がスラブの容量よりも大きい場合、SAN はスラブを削除し、VHD が SAN 上で占有する容量を削減できます。

VHD が Windows 8 ベースのサーバー上にある場合、VHD のスラブ占有量を減らすため、マウントされた VHD 内から、Storage Optimizer も TRIM を送信します。

テスト

以前リリースされたオペレーティング システムには、同等の API はありません。実際の API 自体からパフォーマンス上の影響を受けることはないはずですが、記憶域メディア (正しく実装されている場合) の書き込みパフォーマンスは向上する可能性があります。この API はきわめて慎重に使用する必要があります。対象のエクステントは記憶域メディアから永続的に削除されるため、不要となったエクステントのみを伝達することが重要です。

参考資料

T13 仕様
http://go.microsoft.com/fwlink/p/?LinkId=325502

T10 SCSI 仕様
http://go.microsoft.com/fwlink/p/?LinkId=325503

拡張 SRB (Storage Request Block) をサポートするようになったマルチパス I/O

プラットフォーム

サーバー – Windows Server 2012

説明

Windows Server 2012 では、コア記憶域スタック内の SCSI_REQUEST_BLOCK (レガシ SRB) が、新しい構造体の STORAGE_REQUEST_BLOCK (拡張 SRB) に置き換えられます。拡張 SRB にはレガシ SRB の機能がそのまま引き継がれますが、拡張性とスケーラビリティも兼ね備えています。

マルチパス I/O (MPIO) は拡張 SRB をサポートし、デバイス固有モジュール (DSM) でも拡張 SRB サポート を使用できます。ただし、マルチパス デバイスの記憶域スタックで拡張 SRB を使用するには、**DSM を含めた、スタック内のすべてのコンポーネントが拡張 SRB をサポートしている必要があります**。マイクロソフトの標準の DSM と MSDSM は、拡張 SRB をサポートしている点に注意してください。

コマンド ライン デバッガー (CDB) によって、拡張 SCSI パス スルーのサイズは変化する可能性があるため、SCSI_PASS_THROUGH_EX 構造体は、拡張 MPIO パス スルー構造体の一部ではありません。代わりに、拡張 MPIO パス スルーには、拡張 MPIO パス スルー構造体の開始から SCSI_PASS_THROUGH_EX 構造体までのオフセットを定義するためのフィールドがあります。拡張 MPIO パス スルーを呼び出す場合は、必ず適切なサイズのバッファーを割り当て、SCSI パス スルーのオフセットを正しく設定しておく必要があります。拡張 SCSI パス スルーのルールに従って呼び出す以外、拡張 MPIO パス スルーを使用するにあたってその他の作業は必要ないはずです。

注: DSM が拡張 SRB をサポートしていない場合、MPIO は、MPIO_IOCTL_FLAG_INVOLVE_DSM フラグ セットを持つ拡張 MPIO パス スルー要求を拒否します。

影響

DSM も含め、デバイスの記憶域スタックの一部が拡張 SRB をサポートしていない場合、記憶域スタックはレガシ SRB を使用します。

解決策

DSM で STORAGE_REQUEST_BLOCKS をサポートするための MPIO 要件は次の 2 点のみです。

- レポートされる DSM のタイプが **DsmType6** である
- DSM が **DSM_ADDRESS_TYPE_SUPPORTED** 機能を提供する

DSM のタイプが DsmType6 ではない場合 や、タイプが DsmType6 でも、DSM_ADDRESS_TYPE_SUPPORTED 機能を提供していない場合には、MPIO は DSM が拡張 SRB をサポートしていないと見なします。

DSM_ADDRESS_TYPE_SUPPORTED 機能は 2 つのパラメーターをとり、そのうちの 1 つがアドレス タイプです。このアドレス タイプは、srb.h で定義されている STORAGE_ADDRESS_TYPE_* 値のいずれかでなくてはなりません。DSM は、所定のアドレス タイプをサポートしている場合には TRUE を、サ

ポートしていない場合には FALSE を返す必要があります。"サポート" の定義は、最終的には DSM に応じて異なります。MPIO はこの機能を使用することで、所定のタイプの STOR_ADDRESS 構造体を持つ拡張 SRB を渡された場合に、DSM が誤動作しないようにします。上記に基づいて、MPIO は通常、マルチパス デバイスをリストアップする際にこの機能を呼び出しますが、この機能は随時呼び出し可能です。

DSM が拡張 SRB の STOR_ADDRESS をまったく利用しない場合、DSM は STORAGE_ADDRESS_TYPE_* 値のすべてに対して TRUE を返すことができます。詳しくは、WDK のサンプル DSM を確認してください。

その他の重要事項

- 拡張 SRB をサポートする DSM は、SCSI_REQUEST_BLOCK 構造体と STORAGE_REQUEST_BLOCK 構造体の両方を処理できる必要があります。拡張 SRB をサポートすることが DSM によってレポートされていても、その DSM が拡張 SRB のみを受信するわけではありません。DSM は拡張 SRB に加え、引き続き任意の数のレガシ SRB を受信する可能性があるため、両方を処理できなくてはなりません。
- マイクロソフトの WDK では、srbhelper.h というヘッダー ファイルを提供しています。このファイルには、拡張 SRB とレガシ SRB の両方を処理する必要のあるドライバーで利用可能な、インライン関数が含まれています。マイクロソフトのサンプル DSM ではこのヘッダーを使用しているため、こうした関数の使用方法の例として役立ちます。
- 拡張 SRB をサポートしない DSM では、拡張 SRB を処理する必要はありません。
- DSM が所定の STORAGE_ADDRESS_TYPE_* をサポートしない場合、MPIO はレガシ SRB を使用するように記憶域スタックをフォールバックできます (それ以外の場合は、拡張 SRB をサポートします)。
- "BTL8" は、現在 Windows 8 用に定義されている唯一の STORAGE_ADDRESS_TYPE_* です。

Resilient file system

プラットフォーム

サーバー – Windows Server 2012

説明

Resilient File System (ReFS) は、新しいローカル ファイル システムです。それは履歴データの損失やダウンタイムの原因となるエラーが発生した場合でも、データの可用性を最大にします。ビジネスで重要なデータをエラーから保護し、必要なときに利用できるようにデータの保全性を保証しています。そのアーキテクチャは、絶えず増大するデータ セット サイズと動的ワークロードの時代においてスケーラビリティとパフォーマンスを提供するように設計されています。

ReFS の重要な機能は次のとおりです。

- **完全性**: ReFS はデータ損失の原因となる一般的なエラーの多くから保護されるようにデータを保存します。ファイル システム メタデータは、常に保護されています。必要に応じてユーザー データをボリューム、ディレクトリ、ファイルごとに保護することができます。破損が発生した場合でも、記憶域スペースで構成された ReFS は破損を検知し自動的に修正します。システム エラーのイベントにおいて、ReFS は迅速にユーザー データの損失を回復するように設計されています。
- **可用性**: ReFS はデータの可用性を優先するように設計されています。ReFS は破損が発生し、それを自動的に修復することができない場合は、長時間のダウンタイムなしでオンライン回収処理を破損領域にローカライズします。要するに、破損が発生した場合でも、ReFS はオンラインのままです。
- **スケーラビリティ**: ReFS は今日と明日のデータ セットのサイズのために設計されており、それは、高いスケーラビリティのために最適化されています。
- **アプリの互換性**: アプリの互換性を最大にするために、ReFS は広く採用されている Win32 API に加えて、NTFS 機能のサブセットをサポートします。
- **積極的なエラー識別**: ReFS の整合性機能は、ボリュームを定期的にスキャンして潜在的な破損を識別し、積極的に破損したデータの修復をトリガーするデータ整合性スキャナー (スクラブ) で活用されています。

参考資料

Building Windows 8 ブログ記事「Windows の次世代のファイル システムを構築する: ReFS」
http://go.microsoft.com/fwlink/p/?LinkId=325504

アプリケーションの互換性と ReFS
http://go.microsoft.com/?linkid=9797693

ファイル サーバー API サポート

プラットフォーム

サーバー – Windows Server 2012

説明

Server Message Block 2.2 (SMB 2.2) プロトコルと Resilient/Cluster Shared Volumes (ReFS/CSVFS) ファイル システムのためにサポートされる API ドキュメントは、以下の参考資料として開発者に提供されます。ドキュメントは、次のカテゴリに分類されます。

- ファイル管理機能
- ディレクトリ管理機能
- ボリューム管理機能
- セキュリティ機能
- ファイルとディレクトリ サポート コード
- ボリューム コントロール コード
- メモリにマップされたファイル

使用方法

特定の API は、すべてのプロトコルやすべてのシナリオでサポートされません。一方、その他の API のなかには Windows Server 2012 で廃止されたものもあります。アプリの継続的なサポートを保証するために、サポートされている API のリストを確認してください。

参考資料

API サポートと互換性マトリックス
http://go.microsoft.com/?linkid=9797692

新しいファイル履歴機能

プラットフォーム

クライアント – Windows 8

説明

新しい "ファイル履歴" 機能は、既にある "バックアップと復元" の後継機能で、ユーザーのライブラリに格納されたユーザー ファイルを保護する役割を果たします。開発者には、プログラムを通じて "ファイル履歴" を有効化し、対象となる特定の記憶域を選択できる一連の API が提供されます。これらの API に関するドキュメントは、MSDN で公開しています。

この変更は、データ保護に関するワークフロー、および Windows の "バックアップと復元" 機能に依存するドキュメントに影響を及ぼす可能性があります。

両機能を同時に使用するのは控えてください。"ファイル履歴" はアクティブなバックアップ スケジュールがあるかどうかを確認し、存在する場合は、そのスケジュールを有効化できないようにします。この場合、"ファイル履歴" を使用するユーザーは、対象の Windows バックアップのスケジュールを削除する必要があります。

影響

ワークフローが中断される可能性があります。Windows の "バックアップと復元" 機能の従来の使用方法を説明しているドキュメントに、上記の変更点を反映する必要があります。

解決策

新しい "ファイル履歴" 機能によって悪影響が生じるワークフローを持ったアプリを見直し、競合がある場合は解消して、新しい API のセットを組み込みます。

テスト

開発者は API を使って、"ファイル履歴" が有効になっているかどうかを確認できます。

オペレーティング システムで制御可能となった光学ディスク ドライブへの電力供給

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

以前のバージョンの Windows では、光学ドライブが使われていない場合、その光学ドライブへの電力供給を管理できませんでした。新しいオペレーティング システムでは、光学ディスクドライブ (ODD) 内にメディアが存在しない場合、その光学ドライブへの電力供給を停止できるようになります。これは、ゼロ パワー ODD (ZPODD) と呼ばれる機能です。この機能は、Slimline SATA (Serial Advanced Technology Attachment) コネクタを使用する光学ドライブにのみ適用可能です。

影響

この新しい動作による悪影響は見つかっていませんが、メディアの書き込み用ソフトウェアが想定外の動作をする可能性もあるため、注意が必要です。

軽減策

常に電力を供給する状態に戻すには、レジストリでこの機能を無効化します。レジストリ値への絶対パスは次のとおりです。

```
HKLM\SYSTEM\CurrentControlSet\Services\cdrom\Parameters\ZeroPowerODDEnabled
```

この値の型は DWORD (32 ビット) で、値が 0 の場合は ZPODD が無効化されます。それ以外の値の場合は ZPODD が有効になります。

テスト

この新機能のために、メディアの書き込み用ソフトウェアで異常が発生しないかどうかをテストします。この新機能に関する問題点を発見した場合は、Microsoft (mailto:OptIssue@microsoft.com) にご連絡ください。

USB 3.0 のサポート

プラットフォーム

クライアント – Windows 8

サーバー – Windows Server 2012

説明

Windows 8 と Windows Server 2012 では、新たに USB 3.0 がサポートされるようになりました。そこで、eXtensible Host Controller (XHC) と呼ばれる USB 3.0 ホスト コントローラーを有効化するために、新しいソフトウェア スタックを追加しています。インターフェイスのパリティ、IRQL レベル、呼び出し元コンテキスト、エラー ステータス、再試行の頻度などは変更していません。通常、アプリの開発者がデバイスを操作するうえで、USB 2.0 で接続した場合と USB 3.0 で接続した場合とでは特に違いはありません。

ただし、新しい USB 3.0 デバイス用のドライバーを作成する際は、必ず新しい USB 3.0 の機能を選んでください。

影響

総体的にデバイスの転送速度が向上し、USB デバイスによって消費される PC の電力も減少します。USB 3.0 ポートに接続した場合、既存の USB デバイスは正しく機能しない可能性もあります。こうしたことが起こらないよう配慮していますが、USB 3.0 を有効化するソフトウェアは新しいため、確実とは言い切れません。

ツール、ベスト プラクティス、ガイダンス

このセクションには、既にあるアプリの継続した互換性の確認、または新しいアプリの設計時に最適な品質と互換性を確保するための補助となる項目が含まれます。ここでは以下のツール、ベストプラクティスとガイダンスについて説明します。

- Windows アセスメント ツールキット
- Windows ハードウェア認定キット
- Windows アプリ認定キット

Windows アセスメント ツールキット

プラットフォーム

クライアント – Windows 7 | Windows 8

説明

Windows アセスメント ツールキットと Windows パフォーマンス ツールキットは、Windows アセスメント & デプロイメント キット (ADK) を構成します。これらは、新しいコンピューターへの Windows の自動配置と性能を評価するための、完全な機能を提供します。

本トピックでは **Windows アセスメント ツールキット**を中心に説明を行います。評価結果は開発するハードウェアおよびソフトウェアの応答性、バッテリ寿命、スタートアップ パフォーマンス、シャットダウン時間のような潜在的な問題を診断するために利用できます。これらは OEM パートナーや ISV/IHV パートナー、愛好家、コミュニティ メンバーに計測、比較、品質測定の共通フレームワークを確立します。

Windows アセスメント ツールキットを利用することで、起動時間からバッテリ寿命、高解像度ストリーミングまでさまざまなシナリオ全体でのパフォーマンスを、異なる側面から計測することが可能になります。評価は、調査するべき潜在的な問題、矛盾している動作や注意箇所を特定することができます。

以前の Windows のリリースでは、品質を測定するために、Velocity Test Suite (VTS/VOS)、Fundamental Quality Tools Suite (FQTS)、System Poer and Performance Tools Suite (SPPTS) など複数のツールを利用していました。Windows アセスメント ツールキットはこれらのパフォーマンス診断ツールの機能を統合し、より使いやすく、より幅広く、より意味のある結果を提供します。

Windows アセスメント ツールキットが提供する機能は次のとおりです。

- Windows および付加価値を提供するソフトウェアおよびドライバーとハードウェアの構成の全体的なエクスペリエンス評価を支援します。
- 品質問題の根本原因を特定するのに役立つ詳細なシステム データを提供します。
- 開発期間中の問題を明らかにすることによってコストを削減します。
- 異なるコンピューターまたは同一構成の一連のコンピューターから、時間毎の結果を比較し、これらの結果セットから比較グラフを生成できます。
- 製品の品質を改善するために利用できる標準化されたフォーマットでのフィードバックを生成します。

重要なビジネス目的を達成するためにこの Windows アセスメント ツールキットを利用できます。

- **計測と比較** – 意思決定と提案、競争基準を容易にするために、類似した他のコンポーネントに対して、ソフトウェアやドライバーまたはその両方のコンポーネントを比較するためのデータを利用することができます。
- **品質改善** – 事前定義された品質判定基準に従ってソフトウェアやドライバー、またはその両方のコンポーネントを開発するために、単独またはパートナーと作業することができます。
- **品質の追跡** – コンポーネントのバージョンとイテレーション後の再帰検証に効果的な品質追跡プロセスを作成できます。

利用方法とベスト プラクティス

評価は、コンピューターの状態の特定のセット、アクティビティの計測と記録、記録された結果を維持する XML とバイナリ ファイルの組み合わせです。ジョブは一つ以上の評価と設定のコレクションで、コンピューターで一度実行されます。Assessment Platform は一貫した実行、ジョブ表示、評価、および結果を提供します。結果には、調査や対処がさらに必要な領域を判断するのに役立つ診断と修復情報が含まれていることがよくあります。

- 単一のコンピューターまたは小さなコンピューター コレクションで **Windows アセスメント コンソール** (Windows AC) を実行します。このシナリオでは 1 つまたはいくつかの異なるコンピューター構成上でのパフォーマンス特性を確認できます。Windows AC は評価のグループ化、ジョブの作成、ジョブ結果の管理を行うためのグラフィカル ユーザー インターフェイス (GUI) です。結果には推奨されるアクションが含まれることもあります。
- ラボ環境にある複数のコンピューターで **Windows アセスメント サービス** (Windows AS) を実行します。このシナリオは主に開発環境でデスクトップ、ノート PC またはスレート コンピューターに対して一貫した定性的な評価スイートを実行する必要のあるユーザー向けです。

Windows アセスメント ツールキットが提供する機能は次のとおりです。

- システムについての深い技術的な知識なしでもコンピューターを評価するのに利用できる単純なグラフィカル ユーザー インターフェイス (GUI) と評価
- システム改善を支援するための推奨事項を含む、コンソールまたは UI で確認可能な評価結果
- ワンクリックでの事前に設定されたジョブの実行
- 複数のコンピューター上で実行可能な事前に定義したジョブと評価設定および、比較可能な結果
- 利用したい評価と設定を指定できる、カスタマイズ可能なジョブ
- コマンドライン構文を利用した、スクリプティングと自動化のための Assessment Platform
- ジョブの実行、結果の確認、システム改善のための対応を行い、もう一度ジョブを実行して結果を比較するための機能

アセスメント ツールキットを利用する典型的なシナリオを以下に示します。

シナリオ	説明
"ブラック ボックス"	あらかじめ定義されたジョブを実行し、任意の異常な値や、ドライバー、メモリ利用状況またはその他の項目について結果を調査します。
結果比較	1. サポートされたオペレーティング システムですべてのコンピューターで動作可能な推奨された設定を利用し、一度評価を実行します。 2. 他のコンピューターで動作するジョブをパッケージするために、Windows AC を使います。 3. 結果を比較することができるように、結果を共有します。 違いを確認するために、任意の Windows コンピューターの結果を、他のサポートされたオペレーティング システムの結果と比較します。
クリーン コンピューター	結果のベースラインを確立するためにオペレーティング システムだけを含むクリーンなコンピューターで、評価を実行します。
ハードウェアやソフトウェアを追加したコンピューター	新しいハードウェアまたはソフトウェアをクリーンなコンピューターに追加し、クリーンなコンピューターの結果と比較するために、再度評価を実行します。

シナリオ	説明
評価の作成	評価の拡張や、独自のツールやインフラストラクチャへの評価の統合を行う場合は、パブリックな API を利用してください。

利用可能な評価を以下に示します。

評価	説明
Driver Certification Pre-validation	**Driver Certification Pre-validation** は、Windows オペレーティング システム上で実行されるドライバーが Windows 認定プログラムに適していることを確認します。結果は評価によって見つかる問題 (たとえば未署名のドライバーまたは期限切れの署名) を解決するのに役に立つ推奨事項を含みます。
Driver Verification	**Driver Verification** は、オフラインの Windows イメージまたは実行中の Windows オペレーティング システムがドライバーの正しいセットを含むかを確かめます。結果には問題を解決するための推奨事項が含まれます。問題には、見つからないドライバーや複製されたドライバー、古いドライバーや不要なドライバーが含まれることがあります。
File Handling	**File Handling** は、自動化された共通のファイル操作およびメトリックスの測定を提供します。エンドユーザーがファイルを取り扱うシナリオについてコンピューターがどの程度動作するかを確認できるようにするために、測定基準として期間とスループットを計算します。File Handling は、ユーザーがクライアント上でファイルとフォルダーについて、コピー、移動、圧縮、解凍、削除をシミュレートするためのワークロードのセットを利用します。
Photo Handling	**Photo Handling** は、写真を表示、操作しているエンドユーザーをシミュレーションすることでコンピューターの性能とバッテリの寿命を測定します。
Internet Explorer launch/tab create	**Internet Explorer Startup Performance** は、Internet Explorer を開始するために必要な時間に影響を及ぼす構成要素を特定します。評価は、完全に空白のページ (IExplore.exe プロセスとフレーム生成の時間とタブ生成間隔を含む) を表示する時間を測定します。同時に、すべての拡張、アドインとシステム上にインストールされているツール バーの影響を測定します。ネットワーク パフォーマンスまたはブラウジング パフォーマンスは含まれません。
On/Off	**On/Off Transition** は、Windows 8 のブート、スタンバイ、休止のシナリオを測定します。
Internet Explorer Browsing Performance	**Internet Explorer Browsing Performance** は、Internet Explorer のブラウジングの品質を測定し、CPU とグラフィック ハードウェア能力を評価します。3 つの別々のワークロードがさまざまな方法でブラウジングを行い、コンピューターに負荷をかけます。
Media Transcoding Performance	**Transcode Video** は、ビデオ ファイルを異なるフォーマットまたはビットレートに変更する処理を測定します。この評価は一般的な入力/出力ファイルの形式と解決方法を使った一連の変換操作を実行します。
Windows UI Performance	**Windows UI Performance** は、いくつかの基本的なエクスペリエンスについて Windows 8 ユーザー インターフェイスの範囲内で評価を行います。評価は応答性と描画品質について、検索や Windows ストア アプリからデスクトップ アプリへの遷移のようなユーザー操作をシミュレートします。応答性の結果は、ミリ秒単位で判断されます。低い値は、コンピューターがより速く、より機敏なことを意味します。描画については、フレームレートおよび発生した異常数を結果として表示します。
Memory Footprint	**Memory Footprint** は、規定のオペレーティング システム イメージに対して、定量的にもう一方のオペレーティング システム イメージを比較するために利用します。これにより、物理システムにおけるメモリ フットプリントへ影響を与えるコンポーネントを特定することができます。これらのコンポーネントにはドライバー、アドイン アプリ、プレインストール ソフトウェアおよびアンチウイルス プログラムを含めることができます。
First Boot Performance	**First Boot Performance** は、コンピューターを開始する際のブート、スタート画面の初回表示時間に対して、Windows に影響を及ぼす問題を特定します。結果として遅延を引き起こした原因とエクスペリエンスを改善する推奨事項が提供されます。

評価	説明
Media Streaming	**Streaming Media Performance** は、Internet Explorer のストリーミング メディアを利用した場合のコンピューターの構成に関するパフォーマンスの測定に役立ちます。ストリーミング メディアのエクスペリエンスを理解し、比較と改善を行うために評価結果を利用できます。
WinSAT Comprehensive	**Windows System Assessment (WinSAT)** は、CPU、メモリ、ディスク、グラフィックを含む、いくつかのシステム コンポーネントでコンピューターのパフォーマンスを評価および向上させるために利用できます。WinSAT の包括的な評価結果は数値でコンピューター ハードウェア構成の能力を表します。通常は、高いスコアで評価されたコンピューターは低いスコアで評価されたコンピューターよりもより良く、かつ速く動作することを意味します。
Energy Efficiency	**Energy Efficiency** のジョブはコンピューターのバッテリ寿命に関する評価を自動化する機能を提供します。ワークロードを利用し、Energy Efficiency ジョブはシステム コンポーネントがアイドル状態の時に利用される電力に関する診断を実行します。
MiniFilter Diagnostic Settings	**MiniFilter Diagnostic** オプションは、Internet Explorer Startup Perfomance、File Handling、Boot Performance (Windows 8) の各評価に含まれます。このオプションを選ぶことでさまざまな評価シナリオ上で MiniFilter の動作の影響を評価することができるメトリックスが提供されます。
Windows Media Player Performance and Quality	**Windows Media Player Performance and Quality** は、WMP を起動し、複数のメディアクリップを再生することで、パフォーマンスとメディア再生品質の測定を行います。

Windows パフォーマンス ツールキットなどその他のツールが ADK に含まれています。これらのツールはシステムやアプリのパフォーマンスに関してより深い診断と解析を可能とします。詳しくは、以下の参考資料をご覧ください。

参考資料

Channel9 – BUILD ADK ビデオ
https://channel9.msdn.com/Events/Build/BUILD2011/HW-147T

Windows アセスメント & デプロイメント キットについて
http://go.microsoft.com/fwlink/p/?LinkId=325506

Windows Assessment Toolkit Technical Reference
http://go.microsoft.com/fwlink/p/?LinkId=325507

Assessment Execution Engine
http://go.microsoft.com/fwlink/p/?LinkId=325508

Windows Performance Analysis
http://go.microsoft.com/fwlink/?LinkId=228914

Windows ハードウェア認定キット

プラットフォーム

クライアント – Windows 7 | Windows 8

サーバー – Windows Server 2008 R2 | Windows Server 2012

サーバー/コントローラー – Windows Server 2008 R2

説明

開発者、ISV、IHV と OEM 各社は Windows ハードウェア認定キット (Windows HCK) を利用することで、最新の Windows オペレーティング システムのためのハードウェア装置、システムおよびフィルター ドライバーの動作を保証することが可能となります。これには以下のオペレーティング システム用のハードウェアおよびフィルター ドライバーの動作保証を行うためのすべてのツールとドキュメントが含まれています。

- Windows 8
- Windows 7
- Windows Server 2012
- Windows Server 2008 R2

Windows ハードウェア認定キットは、Windows ロゴ キットに変わる Windows 認定プログラムの一部です。

利用方法とベスト プラクティス

どのようなハードウェアまたはフィルター ドライバーでもテストを行う前には、適切なテスト環境を構築する必要があります。これには、コントローラー、クライアント、拡張可能なハードウェア (たとえば、多機能デバイス) やソフトウェアも含まれます。テスト環境が構築された後、新しい HCK's Studio tool を利用してハードウェアをテストすることができます。以下のステップは、認定テスト プロセスの概要です。

1. テスト環境を準備します。
2. プロジェクトを作成します。
3. 一つ以上のマシン プールを作成します。
4. 確認する機能を選びます。
5. 機能に対してテストを選択し、実行します。
6. テスト結果を確認します。
7. パッケージを提出します。

参考資料

Windows ハードウェア開発
http://go.microsoft.com/fwlink/p/?LinkId=325509

Windows ハードウェア認定プログラム
http://go.microsoft.com/fwlink/p/?LinkId=325510

Windows ハードウェア開発関連のダウンロード
https://msdn.microsoft.com/ja-jp/windows/hardware/hh833788

Windows 用ハードウェアの開発をはじめる
http://go.microsoft.com/fwlink/p/?LinkId=325511

Windows アプリ認定キット 8.0

プラットフォーム

クライアント – Windows 8

説明

Windows アプリ 認定キット 2.2 は、Windows 8 用の Windows SDK に含まれています。認定キットは、Windows 8 の Windows ストア アプリの認定要件に準拠しているかどうかを確認するために利用します。

注: アップグレードを促進するプロンプトが表示された場合は、最新の機能が含まれた認定キットの効果を得るために、Windows アプリ 認定キットをアップグレードしてください。アプリのテスト結果の提出方法に関する詳しい情報は、後日発表されます。最新の Windows アプリ 認定キット でテストすることは、アプリが Windows ストア アプリの要件と Windows デスクトップ認定プログラムに準拠しているかどうかを保証します。

参考資料

Windows アプリ 認定キット
https://developer.microsoft.com/ja-jp/windows/develop/app-certification-kit

Windows ストア ポリシー
Windows ストア アプリの認定要件を掲載しています。
https://msdn.microsoft.com/ja-jp/library/windows/apps/dn764944.aspx

Windows RT PC でのデバッグとテスト
Windows 8 は、タブレットなどの低電力デバイスでの動作を前提に設計されています。ARM ベースプラットフォームの特性を活かすため、Windows RT も同じ原則の下で設計されています。質の高いユーザー エクスペリエンスをすべての Windows 8 プラットフォームで一貫して提供できるように、Windows RT PC で Windows ストア アプリのデバッグとテストを行ってください。
https://msdn.microsoft.com/ja-jp/library/windows/apps/xaml/bg126235.aspx

Windows アプリ認定キットを使用する
作成したアプリを Windows ストアに公開する、または、Windows 認定を受ける最善の方法は、認定のためアプリを提出する前に、ローカルでアプリの検証とテストを行うことです。
https://msdn.microsoft.com/ja-jp/library/windows/apps/xaml/hh694081.aspx