Windows Vista 時代のデバイス・ドライバ開発

第7回　WDF（UMDF）の概要と解説

日高 亜友，川出 智幸，相良 徹

今回はWDFが新たに提供するUMDFに基づいたユーザ・モード・ドライバのアーキテクチャを紹介し，開発手法からインストールまでをサンプル・ドライバを使用して解説する．サンプル・ドライバを改造して別のターゲットに置き換えてテストする方法についても検証する．（筆者）

1. UMDF（User Mode Driver Framework）とは

前回（2007年12月号，pp.166-176）はWDF（Windows Driver Foundation）を構成するKMDF（Kernel Mode Driver Framework）を解説したので，今回はもう一つの構成要素であるUMDF（User Mode Driver Framework）を取り上げます．

WDFには，新しいドライバ・モデルをカーネル・モードとユーザ・モードで実現するための2種類のデバイス・ドライバ・フレームワーク（プログラマ・インターフェースであるライブラリやクラス）と，それをサポートする開発ツールが用意されています．これらはWindows Vistaの登場に合わせて開発されたWDK（Windows Driver Kit）に含まれます．

Windowsにおいてユーザ・モードで汎用的なデバイス・ドライバを開発する仕組みは，今回のWDKで初めて導入されました．UMDF導入の経緯については前回も触れました．これは，WDFという新しいデバイス・ドライバ開発用のフレームワークを開発者向けに提供するため，優先して検討された重要なことです．特に以下の点を目標にして開発されています．

- ドライバのエラーがシステム・ダウンを起こさない安定性の提供
- 次項のような既存システムとの互換性の維持
  - -API（Application Program Interface）
  - -プラグ＆プレイ，電源管理
  - -ロードとアンロード，インストール・パッケージの仕組み
  - -ドライバの階層化，同期機構，メッセージ・パッシング駆動など，既存DDIとの共存
- ユーザ・スペースで動作することによるセキュリティの確保
- KMDFとのドライバ・モデルの共通化〔現在は同一のDDI（Device Driver Interface）やデータ構造を提供することはできていないが，ドライバ・モデルは共通化している〕

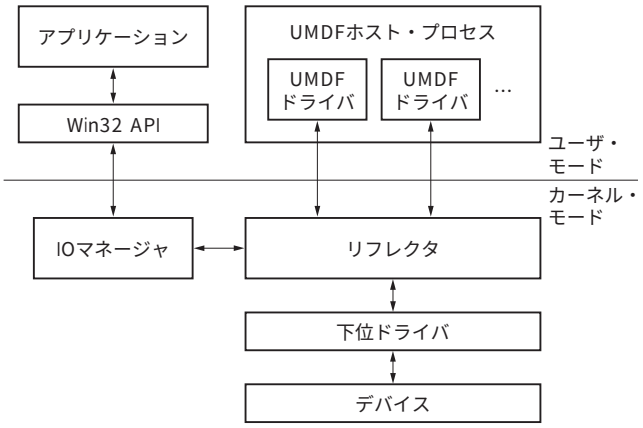

図1　UMDFドライバの構成概念

● UMDFドライバの構造

UMDFドライバとそれを取り巻くシステム・コンポーネントの関係を**図1**に示します．従来のドライバは比較的単純な階層構造を持っていました．それと比べて，UMDFは本来，カーネル・モードで動作するべきデバイス・ドライバの仕事をユーザ・モードに移動するために，「リフレクタ」と呼ぶ機構を導入しています．リフレクタはUMDFの開発に合わせてMicrosoft社が提供しているものですが，この仕組みによって最上位のカーネル・モード・ドライバをユーザ・モードで代替させることができます．現在はリフレクタよりも上位にカーネル・モードのドライバやフィルタ・ドライバを配置することはできませんが，ユーザ・

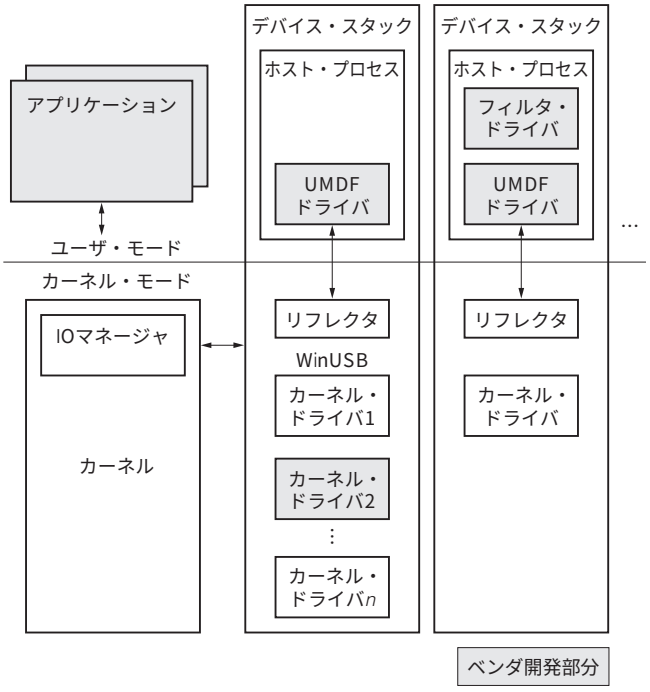

図2　UMDF ドライバの構成図

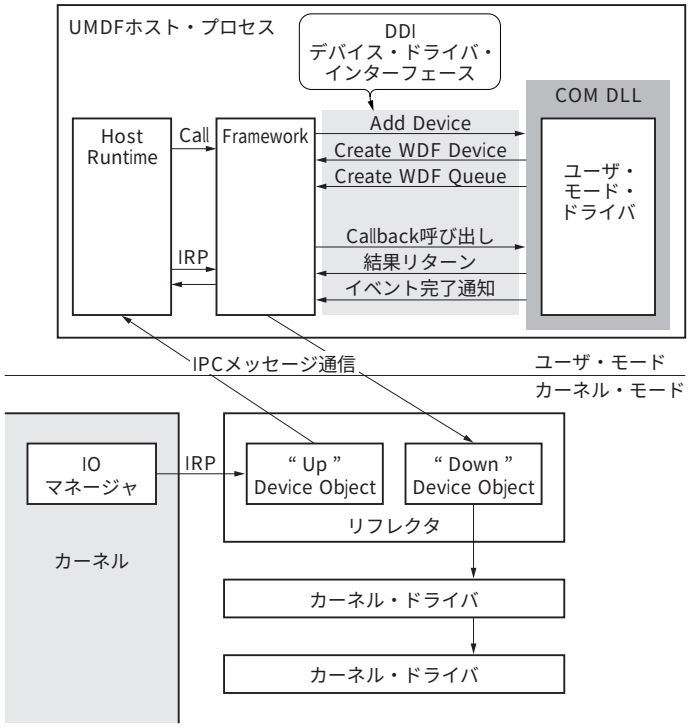

図3　ホスト・プロセスと呼び出し関係

モードのドライバは階層構造を持つことができ，またユーザ・モードのフィルタ・ドライバを配置することも考慮されています(**図2**).

カーネル内の IO マネージャがカーネル・モード・ドライバの管理を行うというアーキテクチャは，Windows NT 時代から変わっていません．UMDF によるユーザ・モード・ドライバの実装では，最上位のカーネル・ドライバ部分においてカーネル・ドライバと同じ振る舞いを IO マネージャと下位層のドライバに対して行いながら，ホスト・プロセス内のユーザ・モード・ドライバと相互通信を行うのがリフレクタの役割です．リフレクタの下位には，後述する WinUSB のように Microsoft 社が提供している Windows にあらかじめ組み込まれているインハウス・ドライバを置くこともでき，ベンダが開発するカーネル・モード・ドライバを配置することも可能です．

カーネルの IO マネージャが，デバイスに対して IO リクエストを発行した場合，複数のドライバがリクエストを処理します．各ドライバはデバイスへのリクエストを処理するために，デバイス・オブジェクトで関連付けられます．デバイス・オブジェクトは，デバイス・ドライバが IO マネージャとやりとりするためのデータ構造やディスパッチ・ルーチンのポインタを持つものです．しかし，関連する複数のデバイス・オブジェクトは，IO リクエストの処理において各デバイス用のスタックとは別の「デバイス・スタック」として構成されます．「デバイス・スタック」からは，デバイス・オブジェクトのスタックのほかに構成するドライバを参照することができ，リクエストにおいて処理対象のデバイスを識別するためのハンドルと関連付けられています．

図3は，ユーザ・モード・ドライバを含むホスト・プロセスと呼び出し関係を中心に示したものです．デバイス・オブジェクトはドライバが駆動するデバイスごとに作成され，COM アーキテクチャの DLL として実装され，個別のユーザ・プロセスとして動作します．

WDM のカーネル・モードのドライバでは，IO マネージャからの要求をディスパッチ・ルーチンのコールバックと IRP によるメッセージ・パッシングで受け取って処理していました．けれどもユーザ・モード・ドライバの構成では，WDF に対する呼び出しと，COM (Component Object Model) インターフェースを介して登録したコールバック・ルーチンの呼び出しという形式で実装しています．

● COM の採用

COM とは Microsoft 社が開発・提唱しているコンポー