Digital Fountain / Windows Media Player Integration

1. Introduction

It is essential to the interests of Digital Fountain to support streaming of digital media to the Windows Media Player. Of the various codecs and media formats supported by Windows Media Player, Windows Media Format (i.e. .asf, .wmv, .wma) is the most important. Windows Media Player versions 6 and 7 are both important due to the platform and browser-plugin restrictions of Windows Media Player 7. In this document, Windows Media Player refers to both version 6 and version 7 unless explicitly stated.

The standard solution for delivering media data to the Windows Media Player over a proprietary transport protocol is to write a DirectShow Source Filter implementing that protocol. Unfortunately, the standard source filter approach does not work for Windows Media Format files. The only way to use standard, published Microsoft APIs and SDKs to implement a source filter for Windows Media Format files is to use the Windows Media Format SDK to implement a combination Source, Splitter and Decompressor filter that will work only for WMF files. This requires a WMFSDK license and redistributable; unfortunately this use of the SDK is in violation of the terms of this license agreement.

The rest of this document outlines the various technical approaches to delivering WMF media data to a Windows Media Player via a proprietary transport protocol.

2. Integration Solutions

Microsoft offers a group of SDKs collectively known as the Windows Media SDK as part of its Windows Media Ecosystem. Of these, the Windows Media Format SDK (WMFSDK) is central to any 3rd-party integration with Windows Media Format-based applications. In addition, the Microsoft DirectShow SDK, part of the Microsoft DirectX set of SDKs, is generally required for tight Windows Media Player integration. The Windows Media SDKs are offered individually under restrictive licensing agreements; the DirectShow SDK is offered on a less restrictive license.

(The current version of the Windows Media Format SDK is 7.0; the current DirectShow SDK is version 8.0, although most of the functionality can be accessed via the previous 6.0 version.)

Digital Fountain has identified several different levels of integration with Window Media Player using the WMFSDK. It is the understanding of Digital Fountain that all of the following proposed uses of the WMFSDK are in violation of the WMFSDK license agreement. At this time, do to the unavailability of the license agreement in question, we have not identified the specific ways that these approaches violate the agreement.

2.1 Standard HTTP proxy solution

This is our current approach to delivering streaming media to all of our supported media players, including Windows Media Player. It does not require the execution of any Microsoft licensing agreements. This solution works as follows. A Digital Fountain URL (rmsp://) is encountered either in a web page or via the Windows shell. This URL is registered to launch our HTTP proxy application. This application manages a

connection with a Digital Fountain server, and presents a standard HTTP socket interface bound to a TCP port on the local machine. The proxy then launches Windows Media Player, passing the URL of its local HTTP socket on the command line. The Windows Media Player behaves as if it were receiving data from a standard HTTP server.

2.1.1 Advantages

The chief advantages of this approach are its simplicity of implementation, the lack of licensing issues, and its applicability to all of our supported player applications (i.e. Windows Media Player, RealPlayer, and QuickTimePlayer)

2.1.2 Disadvantages

The main disadvantages of this approach include the lack of support for seeking, and the general lack of tight integration with the Windows Media Player. Thus we cannot optimize our delivery based on player events such as Pause and Stop. RealPlayer and QuickTime offer some support for seeking using this method.

2.2 Building a custom DirectShow source filter based on WMFSDK

Using the WMFSDK, it is possible to write a DirectShow Source Filter that supports WMF. One would use the IStream interface to the IWMFReader object to introduce media data to the WMFSDK, and would have the WMFSDK output decoded audio and video to the output pins of the Source Filter. This source filter would essentially duplicate the functionality of the Microsoft Windows Media Source Filter, which includes all of the supported streams and functionality of WMF files. This would be a considerable engineering and support effort.

2.2.1 Advantages

Tight, supported integration with both WMFSDK and DirectShow. Support for seeking, catching player events such as pause, etc

2.2.2 Disadvantages

Requires significant development resources for full support of WMF and Windows Media Player functionality. Some interfaces used by the Player itself are undocumented. Requires redistribution of WMFSDK for WMP6 support (unacceptably large download for most of our partners and customers). Would be a challenging integration of both WMF and non-WMF support in the same Source Filter. In addition, due to the very large feature set of the Microsoft Windows Media Source Filter, would require extensive testing and support knowledge for features unrelated to the delivery of data (i.e. URL flipping).

2.3 MMS Proxy

An HTTP proxy does not permit seeking for WMF media. An MMS proxy does. There are two approaches to implementing an MMS proxy: using the WMFSDK, or directly implementing MMS. I treat the two approaches separately below.

Although both of these solutions permit a Digital Fountain/Windows Media Player integrated client to support seeking, Digital Fountain's transport technology does not support true random-access seeking to positions past that which has already been received at the client. In order to disable seeking past this point, it has been necessary to implement a "dummy" Source Filter which implements the DirectShow IMediaSeeking interface. This is explained below.

2.3.1 "Dummy" source filter

The "dummy" source filter is the first filter instantiated by the filter graph for a given URL. In this case, it would be instantiated when a Digital Fountain-registered URL is encountered (i.e. rmsp://...). It sets its output pin count to zero, and subsequently requests the filter graph manager to render another URL,

referring to the local MMS proxy (this proxy is described below). In practice, the filter graph manager permits the original "dummy" source filter to remain a full participant in the filter graph, despite the fact that it is not involved in the actual business of delivering media data to the player. It can then implement the IMediaSeeking interface to "gray out" the portion of the Windows Media Player seek-bar to which seeking should be disallowed.

This approach works in Windows Media Player 6 for all media types without any special Microsoft software license. For Windows Media Player 7, an instance of the Source Filter "CLSID_WMAsfReader" must be manually added to support playout of a WMF file, which requires the WMFSDK license.

2.3.2 Using WMFSDK to create a local MMS proxy

For this solution, we would implement an IStream object to feed media data into an IWMFReader object, transfer uncompressed samples directly to an IWMFWriter object, and register an IWMFWriterNetworkSink with that reader. The IWMFWriterNetworkSInk object has the capability of streaming in the MMS protocol. Thus, no special knowledge of WMF or MMS is required, although a WMFSDK license and WMFSDK redistribution are required.

2.3.2.1 Advantages

Seeking would be enabled. Player events might be available (depending on what is exposed by the WMFSDK; needs more investigation).

2.3.2.2 Disadvantages

WMFSDK must be redistributed. WMFSDK license is required.

2.3.3 Implementing a local MMS proxy directly

In this approach, Digital Fountain would directly implement a local MMS server based on the MMS protocol spec. However, no knowledge of the Windows Media Audio and Video codecs would be necessary, since Digital Fountain will only be transporting the data, not rendering it.

2.3.3.1 Advantages

The WMFSDK would not need to be redistributed. Seeking would be enabled. Pause and other player events would be accessible.

2.3.3.2 Disadvantages

The only disadvantage here is that in order to have a Source Filter instantiated for our protocol, we would have to use the "dummy source filter" method described above. If this continues to function as expected in future versions of the Windows Media Player, this is not a restriction. However, the "dummy source filter" feature is not officially supported by Microsoft.

2.4 Pluggable Protocol

Windows Media Player 7 (based on the WMFSDK) will attempt to use registered Pluggable Protocols as a data source. This will allow Digital Fountain to introduce media data without the execution of any special Microsoft licensing agreements. Unfortunately, Windows Media Player 6 does not support Pluggable Protocols for WMF files, which makes this solution unacceptable, in and of itself.

One other issue with the Pluggable Protocol approach was the performance limitation of using the disk to deliver data to the player. It seems that for non-WMF streams, the player requires data to be dumped to disk ahead of time, from which it will be read by the player. This hurts Digital Fountain in particular due to the disk usage required for storing our received data before forward-error-correction decoding has taken place. We have not determined if WMF files suffer from this restriction.

3. Ideal Solution and Conclusions

Of the solutions listed in this document, the "direct local MMS proxy" (2.3.3 above) is the best that has been identified by Digital Fountain. As listed in its disadvantages however, it is not ideal. The best possible solution for both companies (short of native Windows Media Platform support for Digital Fountain protocols) would be to use the standard Source Filter interfaces to introduce data into the Windows Media Player, such that no "undocumented features" are in use. Digital Fountain is currently unaware of any technical solution that meets this requirement and which does not require a prohibitively large client download (regardless of any licensing issues).

3.1 Native Windows Media Platform support for Digital Fountain protocols

The most complete integration between Digital Fountain's transport protocols and the Windows Media Player would be for Microsoft to support the protocols natively. This could be done either at the WMFSDK level (for support of WMF content only) or at a more general level (for example, URLMON Pluggable Protocols). In this way, Digital Fountain is at no risk of breaking interoperability for any WMFSDK-based applications. Windows Media Platform applications can make use of the benefits of Digital Fountain's technology, and still maintain their full flexibility and functionality.\

3.2 Other uses for Windows Media Format SDK

It should also be noted that Digital Fountain will require the use of the WMFSDK or MMS protocol specification for future products, in particular for integration with the Windows Media Encoder.

4. Integration Solution Matrix

	HTTP proxy	Pluggable Protocol	Custom Source Filter	WMFSDK MMS proxy	Direct MMS proxy
Windows Media Player 6.4	No seeking, no player differentiation	No support.	Requires WMFSDK license Substantial Development and support effort Requires WMFSDK redistribution	Requires WMFSDK license Requires WMFSDK redistribution	Requires MMS specification and license
Windows Media Player 7	No seeking, no player differentiation	Full support for WMF. Performance drawback on disk.	Requires WMFSDK license Substantial Development and support effort	Requires WMFSDK license	Requires MMS specification and license