Introduction to DMOTest

DMOTest is a test utility for Microsoft® DirectX® Media Objects (DMOs). DMOTest helps you to verify that a DMO meets the DMO specification. If you create a DMO, DMOTest should be part of your testing process.

DMOTest supports two kinds of test:

- **Streaming tests:** These tests are designed to verify that the DMO can process data correctly. In these tests, the DMOTest application gives the DMO sample data to process. The application checks for correct return codes, consistent flags, and so forth. (Of course, it cannot verify whether the output is correct, because that depends completely on the DMO.)
- **API tests:** These tests perform various method calls on the DMO. The test application checks for correct return codes, valid parameters, and so forth.

For the streaming tests, DMOTest uses test files written in a custom file format. The test files contain sample data taken from a media source file. For information about the format, see DMO Test Application File Format.

To use DMOTest, perform the following steps:

1. Create test files.

   See Creating Test Files.

2. For each DMO, select which test files to use.

   See Selecting the DMOs.

3. Select the tests.

   See Selecting Test Suites.

4. Run the tests.

   See Running the Tests.

Creating Test Files

Before you run a streaming test on a DMO, you must generate a file with test data for that DMO. The file must conform to a custom format which is recognized by DMOTest. For information about the format, see DMO Test Application File Format.

The test data consists of samples taken from a media source file. The source file must contain data that the DMO can process. For example, if you are testing an audio effect filter, use an audio source file.

To create a test file, you will need:

- A media file that contains the format you want to test.
- The DMO Data Dump filter (included with the Microsoft® DirectX® 8 SDK).
- The GraphEdit utility for Microsoft® DirectShow®

Note If you're not sure what media formats the DMO can accept, see Viewing a DMO's Input Types.

To generate the test data, do the following:

1. Launch GraphEdit.
2. Render the media file that you want to use.
    - From the **File** menu, choose **Render Media File**.
    - Select the media file.
    - Click OK.
    - GraphEdit creates a filter graph and displays it.
3. Locate the filter whose output pin produces the media type you want to test.
    - Right-click a connection point (the arrow that connects an output pin to an input pin).
    - On the pop-up menu, choose **Properties**.
    - GraphEdit displays a property page that lists the media type for the connection.
    - Repeat these steps until you locate the correct filter.
4. Delete all the downstream filters. (Follow the arrows from the output pin.)

    - For each filter, select the filter and type DELETE.
5. Add the DMO Data Dump filter to the graph.
    - From the Graph menu, choose **Insert Filters**.
    - Expand the **DirectShow Filters** node.
    - Select **DMO Data Dump** and click **Insert Filters**.
    - Type a name for the data file, or select an existing data file to overwrite.
    - Click **Open**.
    - Click **Close**.
6. Connect the output pin from step 3 to the DMO Data Dump filter.
    - Drag the mouse from the output pin to the DMO Data Dump filter's input pin.
    - Or, you can right-click the output pin and choose **Render**.
7. Run the graph.
    - Click the play button.
    - Or, choose **Play** from the **Graph** menu.
    - Wait for the graph to stop. (The play option will become enabled again.)
8. Exit GraphEdit.

For more information about GraphEdit, see the topic "Using the GraphEdit Utility" in the DirectShow documentation. Or, run GraphEdit and choose **Contents** from the **Help** menu.

Selecting the DMOs

To select which DMOs you want to test, do the following:

1. In the **Tests** menu, choose **Select DMOs...**
2. Select the check box next to each DMO that you want to test.
3. If you are running streaming tests, select the test files that will provide the test data.
    - Right-click the DMO.
    - In the pop-up menu, choose **Select Test File**
    - To add test files, click **Add File**.
    - To remove files, click **Remove File** or **Remove All**.

For information about generating test files, see Creating Test Files.

Selecting Test Suites

The DMOTest application provides two test suites, a streaming test suite and an **IMediaObject** interface test suite. You can select an entire suite, or select individual tests from within a suite.

For information about the test suites, see Test Suites. For information about how to run the tests, see Running the Tests.

Select an Individual Test

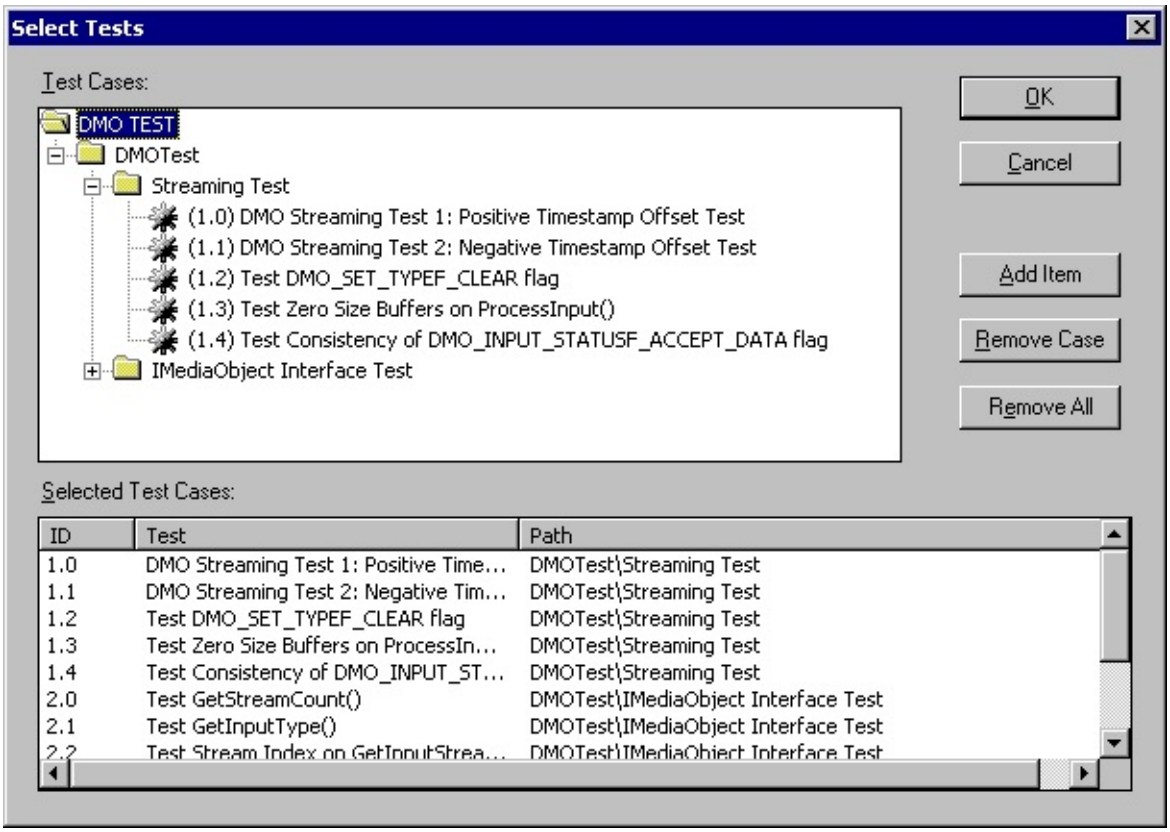
1. From the **Tests** menu, choose **Select Tests...**
2. Expand the **DMOTest** node.
3. Expand the **Streaming Test** node or the **IMediaObject Interface Test** node.
4. Select a test.
5. Click **Add Item**.

Select an Entire Suite

1. From the **Tests** menu, choose **Select Tests...**
2. Expand the **DMOTest** node.
3. Select either the **Streaming Test** node or the **IMediaObject Interface Test** node.
4. Click **Add Item**.

Select All Tests

1. From the **Tests** menu, choose **Select Tests...**
2. Select the **DMOTest** node.
3. Click **Add Item**.



Running the Tests

To run tests on one or more DMOs, do the following:

1. Select the DMOs you want to test.

   See Selecting the DMOs.

2. Select test suites.

   See Selecting Test Suites.

3. From the **Tests** menu, choose **Run Tests**.

The output from the tests appears in the application window. The application also writes the output to a log file. For information about setting the log file, see Setting the Log File.

The test output includes the following information:

- The API calls made for each test.
- Whether each DMO passed or failed a given test.
- The cause of any failures. If the DMO did not handle a parameter correctly, the parameter value is listed. If the DMO gave an invalid return value, the HRESULT is listed.
- A summary of the results.

Viewing a DMO's Input Types

To view the preferred media types on a DMO's input streams, do the following:

1. From the **Tests** menu, choose **Select DMOs...**
2. Right-click the DMO you want to test.
3. On the pop-up menu, choose **Get Properties**.
4. DMOTest displays a list of the DMO's preferred formats for each input stream.

Saving Test Configurations

By default, the DMOTest application saves your most recent configuration in a file called Dmotest.ini. This file is located in the same directory as the application. The next time you run DMOTest, it loads the configuration from this file.

You can also save a configuration into a separate project file. Do the following:

1. From the **File** menu, choose **Save Settings As**.
2. Type the name of the project file. The file name extension is .pro.
3. Click **Save**.

To load a saved project file, do the following:

1. From the **File** menu, choose **Load Settings**.
2. Select a project file.
3. Click **Open**.

Setting the Log File

To specify the log file where DMOTest saves test results, do the following:

1. From the **Options** menu, choose **Set Logging**.
2. In the **Destination** list, click **Log File**.
3. Click **Setup**.
4. In the **File Name** box, type the file name.
5. Or, click **Browse** and select a file.

DMO Test Application File Format

This section describes the file format what is used by the DMOTest application. For information about how to create a test file, see Creating a Test File.

This section contains the following topics.

File Format Structure

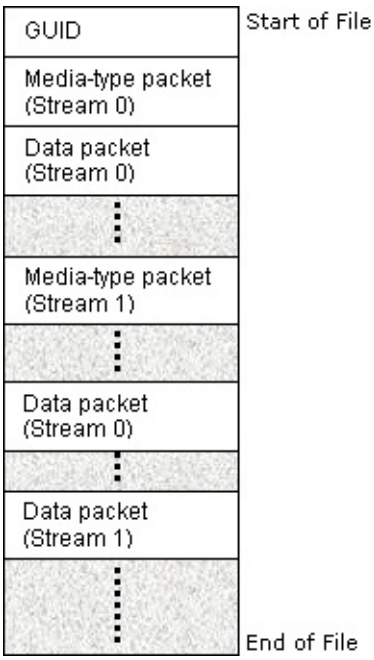
The file format consists of a globally unique identifier (GUID) followed by a series of *packets*. Each packet contains data for the DMO to process. Some packets define a media type, other packets contain media samples. The file must have the following format.

- The first 16 bytes of data contain the following GUID:

  {3A337620-9497-11D3-B30B-444553540000}

- After the GUID, the file contains media-type packets and data packets. A media-type packet defines the media type for one input stream on the DMO. A data packet contains media samples for an input stream to process.
- For each input stream, at least one media-type packet must appear before any data packets appear. Otherwise, the test application cannot set the input stream's media type, and the DMO will reject the data.
- A stream can change media types by including another media-type packet. The new format applies to the subsequent data packets for that stream, until the next media-type packet.
- Packets for different input streams can be interleaved. The test application does not assume that streams are ordered in any particular way.
- A media-type packet does not need to be followed by any data packets.
- The file is not required to have multiple streams, and a stream is not required to have multiple media-type packets.

The following illustration schematizes the file structure.


GUID
Start of File
Media-type packet (Stream 0)
Data packet (Stream 0)
Media-type packet (Stream 1)
Data packet (Stream 0)
Data packet (Stream 1)
End of File

Packet Header Description

Each packet starts with a packet header.

Packet headers enable the test application to determine a packet's type, length, and stream index.

Byte Offset	Field	Size (Bytes)
0–3	Packet Length	4
4–7	Packet Type	4
8–11	Stream Index	4

Packet Length (Unsigned Integer)
: The number of bytes stored in the packet. For media-type packets, the length includes the optional format structure, which is appended to the end of the packet. For data packets, the length includes the length of the media sample.

Packet Type (Unsigned Integer)
: This field contains an ANSI string, which does *not* include a terminating Null character. The string identifies the packet type.
  - Media-type packet: TYPE
  - Data packet: DATA

Stream Index (Unsigned Integer)
: The index of an input stream on the DMO. For a media-type packet, the test application sets the stream's media type to the type specified by the packet. For a data packet, the application calls the DMO to process the packet's media sample. The stream index is zero-based.

Media-Type Packet Description

Media-type packets instruct the test application to set the media type on one of the DMO's input streams. For each input stream, the first packet must be a media-type packet. Otherwise, the DMO cannot decode the information contained in the data packets.

Byte Offset	Field	Size (Bytes)
0–11	Packet Header	12
12–15	Reserved	4
16–87	Media Type Structure	72
88–End of packet	Optional Format Structure	Variable

Packet Header
: For a description of this field, see Packet Header Description.

Reserved (Unsigned Integer)
: Must be zero.

Media Type Structure (DMO_MEDIA_TYPE)
: This field defines the media type. For a description of the DMO_MEDIA_TYPE structure, refer to the Microsoft® DirectX® documentation. Structure members are stored in the order they are declared. The **punk** and **pbFormat** members must equal NULL (zero). If the **cbFormat** member is not zero, the format structure must be appended to the packet.

Optional Format Structure
: If you include this field, specify the size of the field in the **cbFormat** member of the DMO_MEDIA_TYPE structure. If the **cbFormat** member is zero, the packet must not include this field. For more information, see the documentation for the DMO_MEDIA_TYPE structure.

Data Packet Description

Data packets contain media samples for the DMO to process. Each data packet also contains the sample's time stamp, duration, and associated flags.

Byte Offset	Field	Size (Bytes)
0–11	Packet Header	12
12–15	Sample Flags	4
16–23	Sample Start Time	8
24–31	Sample Duration	8
32–End of packet	Sample Data	Variable

Packet Header
: For a description of this field, see Packet Header Description.

Sample Flags
: This field can have the same values as the *dwFlags* parameter of the **IMediaObject::ProcessInput** method. For more information, see the documentation for the **IMediaObject** interface.

Sample Start Time
: This field can have the same values as the *rtTimestamp* parameter of the **IMediaObject::ProcessInput** method. For more information, see the documentation for the **IMediaObject** interface.

Sample Flags
: This field can have the same values as the *rtTimelength* parameter of the **IMediaObject::ProcessInput** method. For more information, see the documentation for the **IMediaObject** interface.

Sample Data
: The data in the sample, which is processed by DMO. The length of the sample data is equal to the length of the packet (given in the packet header) minus 32. Zero-length samples are invalid.

Test Suites

DMOTest supports two test suites:

- Streaming Tests
- API Tests

Streaming Tests

These tests are designed to test whether the DMO processes data correctly. They require a file with test data. For more information, see DMO Test Application File Format and Creating Test Files.

1.0 DMO Streaming Test 1: Positive Timestamp Offset Test

This test uses the DMO to process all of the data in the test file. It writes the output to a temporary file. Then, it makes a second pass using the same data. In the second pass, the test application adds a positive offset to each time stamp on the input data. It writes the output to a second temporary file, and compares the two files. They should contain the same data.

1.1 DMO Streaming Test 2: Negative Timestamp Offset Test

This test is identical to the test described in **1.0**, except that it adds a negative offset on the second pass, rather than a positive offset.

1.2 Test DMO_SET_TYPEF_CLEAR flag

This test verifies that the DMO correctly handles the DMO_SET_TYPEF_TEST_ONLY flag. If the media types are cleared on all of the DMO's streams, the DMO should accept new media types. The test sets media types for all the streams. Then it clears all the media types and tries to set the original types again.

1.3 Test zero-size buffers on ProcessInput

This method verifies that the DMO correctly handles zero-size buffers. First it calls the **IMediaObject::ProcessInput** method with a zero-size buffer. The DMO should return an error code. Then it calls the **IMediaObject::ProcessOutput** method. Again, the DMO should return an error code. Also, it should not return the DMO_OUTPUT_DATA_BUFFERF_INCOMPLETE flag.

1.4 Test consistency of DMO_INPUT_STATUSF_ACCEPT_DATA flag

In the **IMediaObject::GetInputStatus** method, the DMO should return the DMO_INPUT_STATUSF_ACCEPT_DATA flag if it can accept more input. The reverse is also true: If the DMO does not return this flag, it should not accept more input.

This test attempts to process input in both situations. It verifies that the return code from the **IMediaObject::ProcessInput** method is consistent with the status reported by **GetInputStatus**.

API Tests

These tests are designed to verify that the DMO follows the **IMediaObject** specification, with regard to error codes, parameter checking, and so forth.

2.0 Test GetStreamCount

This test verifies that the **IMediaObject::GetStreamCount** method returns E_POINTER when given a NULL parameter.

2.1 Test GetInputType

This test verifies that the **IMediaObject::GetInputType** method correctly handles invalid parameters for the stream index and the media-type index.

2.2 Test Stream Index on GetInputStreamInfo

This test verifies that the **IMediaObject::GetInputStreamInfo** method correctly handles invalid stream indices. It also validates the flags returned by this method.

2.3 Test Stream Index on GetOutputStreamInfo

This test verifies that the **IMediaObject::GetOutputStreamInfo** method correctly handles invalid stream indices. It also validates the flags returned by this method.

2.4 Test Invalid Parameter on GetInputStreamInfo

This test verifies that the **IMediaObject::GetInputStreamInfo** method correctly handles NULL parameters. It also validates the flags returned by this method.

2.5 Test Invalid Parameter on GetOutputStreamInfo

This test verifies that the **IMediaObject::GetOutputStreamInfo** method correctly handles NULL parameters. It also validates the flags returned by this method.