System Transport Controls Developer Guide

February 28, 2012

Abstract

This paper provides information about using the media transport controls to develop music applications for Windows 8 Beta. It provides guidelines for developers to access and configure the media transport controls. It assumes that the reader is familiar with Windows events and event handling, HTML5 and JavaScript programming.

This information applies to the following operating systems:  
 Windows 8 Consumer Preview

References and resources discussed here are listed at the end of this paper.

The current version of this paper is maintained on the web at:   
 [System Transport Controls Developer Guide](http://msdn.microsoft.com/library/windows/hardware/hh833781)

**Disclaimer**: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet website references, may change without notice. Some information relates to pre-released product which may be substantially modified before it’s commercially released. Microsoft makes no warranties, express or implied, with respect to the information provided here. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2012 Microsoft. All rights reserved.

Microsoft logo

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| February 28, 2012 | First publication | | | |

Contents

[Introduction to Windows 8 transport controls 3](#_Toc315420701)

[User experiences 4](#_Toc315420702)

[The volume control experience 4](#_Toc315420703)

[The default volume control 4](#_Toc315420704)

[The media transport control experience 4](#_Toc315420705)

[The default media transport control 4](#_Toc315420706)

[The background media transport control 4](#_Toc315420707)

[Working with the Windows 8 volume and media transport controls 4](#_Toc315420708)

[The volume control 4](#_Toc315420709)

[Media transport controls 5](#_Toc315420710)

[Play/Pause media controls 5](#_Toc315420711)

[Previous and Next media controls 6](#_Toc315420712)

[Reference for media transport controls 8](#_Toc315420713)

[Working with Metadata 9](#_Toc315420714)

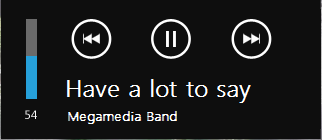
[Resources 10](#_Toc315420715)

# Introduction to Windows 8 transport controls

The current media buttons for Windows 8 rely on Windows media commands (WM\_APPCOMMANDS) to transmit the button press events to the audio application. Applications can either listen for the WM\_APPCOMMAND in their window, where the event happened, or they can register for a shell hook in the Windows shell, and intercept the media button–related events.

In the case of Metro style apps, there is no concept of a window, and the desktop shell hook mechanism is not projected for these apps to access. So without the new *media transport control* features in Windows 8, Metro style apps would have needed substantial additional programming to use the media buttons.

The system-wide media transport controls enable music application developers to plug into a system-based mechanism for monitoring and manipulating media buttons. The media transport controls can, for example, allow a user to control a music application that is in the background, and also to get current information about the track that is playing.



**Figure 1: Media transport controls with current track info**

For the media buttons to work correctly the application must be able to detect that a button press has happened. Parts of the media transport controls have been wired up for the Windows 8 Media Engine so that if an app is based on the HTML5 media element, then whenever the app has focus the app will also have access to the following media commands:

* Play
* Pause
* Play/Pause toggle
* Stop
* Next track
* Previous track

In addition to the media transport controls discussed in the preceding paragraphs, there is a volume control that will also be presented in this paper.

The Windows 8 media transport control and volume control features make it easy for Metro style app developers to add audio capabilities to their apps, to enhance the user experience. The following section presents a few examples of such user experiences.

# User experiences

The Windows media commands are some of the most widely-used button shortcuts on keyboards, whether we’re looking at the desktop PC or laptop keyboards.

The following scenarios show some of the ways a user’s media experience can be enhanced by apps that have been developed using Windows 8 media transport and volume controls.

## The volume control experience

### The default volume control

Jane is watching a YouTube clip in the Metro style environment. The clip is a little too loud so she presses the volume down button on her keyboard. The volume level becomes quieter.

## The media transport control experience

### The default media transport control

John is watching a movie in the Metro style environment. His popcorn has just finished popping but he doesn’t want to miss any parts of the movie. So John presses the pause button on his keyboard and the movie pauses.

### The background media transport control

John is listening to music while browsing the web in the Metro style environment. A track starts to play that John doesn’t like, so he presses the next track button on his keyboard. John’s browsing experience is not interrupted and his media player skips to the next track in the background.

# Working with the Windows 8 volume and media transport controls

When you develop a Metro style app and you use the HTML5 media tag to add audio capabilities, then your application can respond to media controls if it registers to receive the events.

## The volume control

The default way in which Windows 8 displays the volume control is by using a stand-alone volume control flyout.

The volume control flyout is activated by pressing the hardware volume up or volume down keys on a device.



**Figure 2: Volume control flyout**

In the case of keyboard, laptop, and audio device volume controls, the volume level indicated in the flyout is the default endpoint level. But in the case of headsets and Bluetooth audio–aware devices, the indicated volume level is for the specific device endpoint.

Windows 8 provides built-in handling for the volume control, so apps do not need additional programming to use this control.

For more information about audio endpoints, see [Audio Endpoint Devices](http://msdn.microsoft.com/en-us/library/windows/desktop/dd370793(v=vs.85).aspx) and [Audio Endpoint Builder Algorithm](http://msdn.microsoft.com/en-us/library/windows/hardware/ff536200(v=vs.85).aspx).

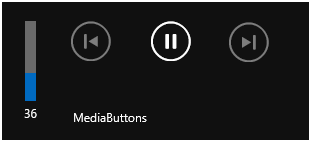
## Media transport controls

If an application is registered to play music in the background, the minimum support that must be enabled is the handling of the play and pause buttons. By having the application respond to these events, users will be presented with a user interface (UI) that will allow them to play/pause the music stream in the background without bringing the application to the foreground. If the application supports multiple tracks, the next and previous buttons should be registered and unregistered as appropriate.

**Note:** Media controls are required for music that plays in the background. So your app has to, at least, provide the minimum support for the play and pause buttons.

### Play/Pause media controls

When an application provides support for the play/pause toggle, or the play and pause buttons discretely, that support activates the play and pause buttons in the media transport UI. In addition, the application should notify the operating system of the playing state so that the appropriate button can be displayed.



**Figure 3: Media transport controls with previous/next buttons disabled**

If the app doesn’t support multiple tracks, or supports multiple tracks but the previous/next buttons are not registered and unregistered as appropriate, then the previous/next buttons are disabled as shown in Figure 3.

Here is a JavaScript code snippet that shows how to register for events from the buttons, and how to respond to those events with event handlers.

// Assign the button object to MediaControls

MediaControls = Windows.Media.MediaControl;

// Add event listeners for the buttons

MediaControls.addEventListener(“playpressed”, play, false);

MediaControls.addEventListener(“pausepressed”, pause, false);

MediaControls.addEventListener(“playpausetogglepressed”, playpausetoggle, false);

// Add event listeners for the audio element

document.GetElementById(“audiotag”).addEventListener(“playing”, playing, false);

document.GetElementById(“audiotag”).addEventListener(“paused”, paused, false);

document.GetElementById(“audiotag”).addEventListener(“ended”, ended, false);

// Define functions that will be the event handlers

function play() {

document.getElementById(“audiotag”).play();

}

function pause() {

document.getElementById(“audiotag”).pause();

}

function playpausetoggle() {

if(MediaControls.isPlaying === true) {

document.getElementById(“audiotag”).pause();

} else {

document.getElementById(“audiotag”).play();

}

}

function playing() {  
 MediaControls.isPlaying = true;

}

function paused() {

MediaControls.isPlaying = false;

}

function ended() {

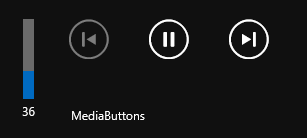
MediaControls.isPlaying = false;

}

List End

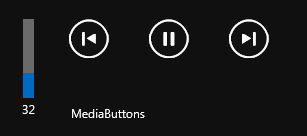
### Previous and Next media controls

The previous and next track buttons should only be enabled when the buttons are functional in the application. For example, at the beginning of a playlist the previous track button should be unregistered and at the end of a playlist the next track button should be unregistered to disable these buttons. But, if applicable, then for all tracks after the first and before the last track, the previous and next track buttons should be enabled.



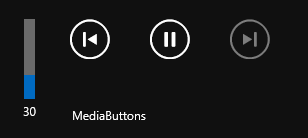
**Figure 4: Media transport controls at beginning of playlist**

At the beginning of the playlist, there is no previous track, so notice in the preceding image that the previous track button is disabled.



**Figure 5: Media transport controls in middle of playlist**

Notice that after the first track, but before the last track – in the middle of the playlist – the previous and next track buttons are enabled.



**Figure 6: Media transport controls at end of playlist**

At the end of the playlist, there is no next track, so notice in the preceding image that the next track button is disabled.

Here is a JavaScript code snippet that shows how to enable the previous and next track buttons by adding event listeners to the MediaControl object.

// Assign the button object to MediaControls

MediaControls = Windows.Media.MediaControl;

…

// Enable the previous track button

MediaControls.addEventListener(“previoustrackpressed”, previoustrack, false);

// Enable the next track button

MediaControls.addEventListener(“nexttrackpressed”, nexttrack, false);

List End

Here is a JavaScript code snippet that shows how to disable the previous and next track buttons by removing their event listeners from the MediaControl object.

// Assign the button object to MediaControls

MediaControls = Windows.Media.MediaControl;

…

// Disable the previous track button

MediaControls.removeEventListener(“previoustrackpressed”, previoustrack);

// Disable the next track button

MediaControls.removeEventListener(“nexttrackpressed”, nexttrack);

List End

## Reference for media transport controls

As shown in the preceding sections and code snippets, you add or remove event listeners to the MediaControl object to enable or disable the media control buttons as appropriate.

The MediaControl class is a member of the Windows.Media namespace and it provides access to 12 events that you can monitor and respond to in your Metro style app.

Here is a list of the events for the Windows.Media.MediaControl class:

* ChannelDownPressed
* ChannelUpPressed
* FastForwardPressed
* NextTrackPressed
* PausePressed
* PlayPauseTogglePressed
* PlayPressed
* PreviousTrackPressed
* RecordPressed
* RewindPressed
* SoundLevelChanged
* StopPressed

For more information about the MediaControl class, see [MediaControl class](http://msdn.microsoft.com/library/windows/apps/windows.media.devices.mediacontrol.aspx).

# Working with Metadata

An application can use the various properties of the MediaControl class to provide information about the audio tracks in a playlist. And it is considered good programming practice to set the metadata whenever a source attribute (src) is set in an audio tag.

Setting the trackName and artistName on the MediaControl object sets the corresponding text in the transport control UI.

The following JavaScript code snippet shows how to set the artistName and trackName properties for an audio track.

// Setting artistName and trackName

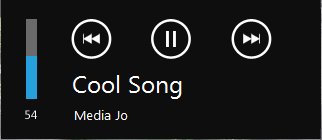
fileObject.properties.getMusicPropertiesAsync().then(function (musicProperties) {

MediaControls.artistName = musicProperties.artist;

MediaControls.trackName = musicProperties.title;

});

List End



**Figure 7: Displaying artist and track names**

Clearing the trackName and artistName on the MediaControl object clears the corresponding text in the transport control UI. The following JavaScript code snippet shows how to clear the metadata.

// Clearing metadata

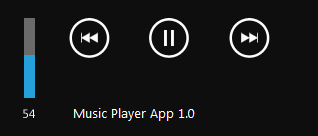
MediaControls.artistName = “”;

MediaControls.trackName = “”;

List End

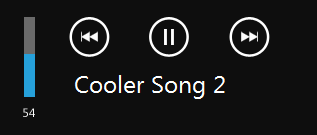
If you clear the trackName and artistName properties and leave them blank, the UI simply displays the name of the app.

Also, if you don’t set either name (artist or track), or you only set the artist name, then the name of the app is displayed. The following image shows how the app’s name is displayed.



**Figure 8: Displaying app name**

You can also decide to set the trackName property and leave out artistName. When you do that, the track information is displayed and the artist’s name is left blank.



**Figure 9: Displaying track name but no artist name**

For more information about the MediaControl properties that expose metadata information, see [MediaControl class](http://msdnpreview.redmond.corp.microsoft.com/en-us/library/windows/apps/windows.media.mediacontrol.aspx).

# Resources

Audio Endpoint Devices

<http://msdn.microsoft.com/library/windows/desktop/dd370793(v=vs.85).aspx>

Audio Endpoint Builder Algorithm

<http://msdn.microsoft.com/library/windows/hardware/ff536200(v=vs.85).aspx>

MediaControl class

<http://msdn.microsoft.com/library/windows/apps/windows.media.devices.mediacontrol.aspx>

Adding audio

<http://msdn.microsoft.com/library/windows/apps/hh452716.aspx>