|  |
| --- |
| Podcast Utilities |
| Podcast Utilities |
| User Guide |
|  |
| **Derek Wilson** |
| **November 2022**  **Version 3.1.0** |

|  |
| --- |
| How to use and configure Podcast Utilities |

# Contents

[1. Contents 2](#_Toc83389346)

[1. What is Podcast Utilities? 5](#_Toc83389347)

[1.1. Benefits 5](#_Toc83389348)

[2. Installing Podcast Utilities from the distribution ZIP 5](#_Toc83389349)

[2.1. Prerequisites 5](#_Toc83389350)

[2.1.1. Using .NET Framework on Windows 5](#_Toc83389351)

[2.1.2. Using the cross platform .NET Core 6](#_Toc83389352)

[2.1.3. Using Mono 6](#_Toc83389353)

[2.1.4. TLS 1.0 6](#_Toc83389354)

[2.1.5. MTP 6](#_Toc83389355)

[2.2. Copying the distribution 6](#_Toc83389356)

[2.3. Distribution contents 7](#_Toc83389357)

[2.3.1. .NET Framework 7](#_Toc83389358)

[2.3.2. .NET Core 7](#_Toc83389359)

[3. Installing Podcast Utilities from Chocolatey 8](#_Toc83389360)

[3.1. Prerequisites 8](#_Toc83389361)

[3.1.1. NET Framework/.NET Core 8](#_Toc83389362)

[3.1.2. Chocolatey 8](#_Toc83389363)

[3.2. Installing 8](#_Toc83389364)

[3.2.1. .NET Framework 8](#_Toc83389365)

[3.2.2. .NET Core 9](#_Toc83389366)

[4. DownloadPodcasts 9](#_Toc83389367)

[4.1. Typical Usage 9](#_Toc83389368)

[4.2. Command Line 9](#_Toc83389369)

[5. SyncPodcasts 9](#_Toc83389370)

[5.1. Typical Usage 9](#_Toc83389371)

[5.2. Command Line 9](#_Toc83389372)

[6. PurgePodcasts 10](#_Toc83389373)

[6.1. Typical Usage 10](#_Toc83389374)

[6.2. Command Line 10](#_Toc83389375)

[7. GeneratePlaylist 10](#_Toc83389376)

[7.1. Typical Usage 10](#_Toc83389377)

[7.2. Command Line 10](#_Toc83389378)

[8. Using the utilities 11](#_Toc83389379)

[8.1. Downloading to a PC and then synchronising to a device 11](#_Toc83389380)

[8.1.1. Setting up 11](#_Toc83389381)

[8.1.2. Key elements in the control file 11](#_Toc83389382)

[8.1.3. Workflow 13](#_Toc83389383)

[8.2. Downloading to a PC and then synchronising to a device – alternative strategy 13](#_Toc83389384)

[8.2.1. Setting up 13](#_Toc83389385)

[8.2.2. Key elements in the control file 13](#_Toc83389386)

[8.2.3. Workflow 14](#_Toc83389387)

[8.3. Download directly to a flash drive or media player 14](#_Toc83389388)

[8.3.1. Setting up 14](#_Toc83389389)

[8.3.2. Key elements in the control file 14](#_Toc83389390)

[8.3.3. Workflow 16](#_Toc83389391)

[9. Control File Format 16](#_Toc83389392)

[9.1. Example Control File 16](#_Toc83389393)

[9.1.1. Global Section 17](#_Toc83389394)

[9.1.1.1. sourceRoot 17](#_Toc83389395)

[9.1.1.2. destinationRoot 18](#_Toc83389396)

[9.1.1.3. playlistFilename 18](#_Toc83389397)

[9.1.1.4. playlistFormat 18](#_Toc83389398)

[9.1.1.5. playlistPathSeparator 19](#_Toc83389399)

[9.1.1.6. freeSpaceToLeaveOnDestinationMB 19](#_Toc83389400)

[9.1.1.7. sortfield 19](#_Toc83389401)

[9.1.1.8. deleteEmptyFolder 19](#_Toc83389402)

[9.1.1.9. sortdirection 19](#_Toc83389403)

[9.1.1.10. maximumNumberOfConcurrentDownloads 19](#_Toc83389404)

[9.1.1.11. retryWaitInSeconds 20](#_Toc83389405)

[9.1.1.12. feed section 20](#_Toc83389406)

[9.1.1.13. format 20](#_Toc83389407)

[9.1.1.14. downloadStrategy 20](#_Toc83389408)

[9.1.1.15. namingStyle 20](#_Toc83389409)

[9.1.1.16. maximumDaysOld 21](#_Toc83389410)

[9.1.1.17. deleteDownloadsDaysOld 21](#_Toc83389411)

[9.1.1.18. diagnostics section 21](#_Toc83389412)

[9.1.1.19. retainTempFiles 21](#_Toc83389413)

[9.1.1.20. outputLevel 21](#_Toc83389414)

[9.1.1.21. postdownloadcommand section 22](#_Toc83389415)

[9.1.1.22. command 22](#_Toc83389416)

[9.1.1.23. arguments 22](#_Toc83389417)

[9.1.1.24. workingdirectory 22](#_Toc83389418)

[9.1.2. Podcast Section 22](#_Toc83389419)

[9.1.2.1. folder 22](#_Toc83389420)

[9.1.2.2. pattern 23](#_Toc83389421)

[9.1.2.3. number 23](#_Toc83389422)

[9.1.2.4. sortfield 23](#_Toc83389423)

[9.1.2.5. sortdirection 23](#_Toc83389424)

[9.1.2.6. deleteEmptyFolder 23](#_Toc83389425)

[9.1.2.7. postdownloadcommand section 23](#_Toc83389426)

[9.1.2.8. feed section 23](#_Toc83389427)

[9.1.2.9. url 23](#_Toc83389428)

[9.1.2.10. format 23](#_Toc83389429)

[9.1.2.11. downloadStrategy 23](#_Toc83389430)

[9.1.2.12. namingStyle 24](#_Toc83389431)

[9.1.2.13. maximumDaysOld 24](#_Toc83389432)

[9.1.2.14. deleteDownloadsDaysOld 24](#_Toc83389433)

# What is Podcast Utilities?

PodcastUtilities are a set of utilities designed to help manage digital media on removable drives.

## Benefits

There are many sophisticated media management programs available for organising digital media however the key differences that Podcast Utilities has are

* Portable  
  Podcast Utilities is a “portable” application, in that it does not need to be installed and can be run from any drive type, local network or removable flash drive
* Cross Platform  
  Podcast Utilities can be run on Windows, Linus or Mac machines by using the .NET Core cross platform version of the utilities.
* Configurable  
  Almost any synchronisation solution can be configured, for example copy 3 of this podcast, all of these podcasts, in this order, and leave a certain amount of space on the device. The number of concurrent downloads can easily be configured.
* Lightweight  
  The application does not use large amounts of memory in a system tray icon all the time its not being used, it only uses system resources when its being run.
* Support for UMS and MTP  
  PodcastUtilities supports USB Mass Storage (UMS) devices, where the device appears as a drive letter in Explorer like a flash drive, as well as MTP where the device appears as a portable device, for example many phones. MTP is not supported in the .NET Core version.
* Scriptable  
  Podcast Utilities lends itself to being run as a scheduled task or from a script. The API is “headless” and can easily be called and used from other applications
* Open Source  
  The source is open and available
* Free  
  There is no cost to using Podcast Utilities for any use.
* No need for an iTunes account  
  Podcasts promoted on iTunes can be downloaded however there is no need to have an iTunes account.

# Installing Podcast Utilities from the distribution ZIP

Podcast Utilities does not require any special installation, just copy the files from the dirstibution ZIP into any folder and run any of the utilities. The utilities can be run from local hard disks, network shares or removable media such as flash drives.

## Prerequisites

The following are required to install PodcastUtilities

### Using .NET Framework on Windows

1. A machine running Windows XP / Vista / 7 / 8 / 10
2. .NET Framework 3.5 (on Windows 10 PodcastUtilities will use .NET Framework 4)

### Using the cross platform .NET Core

1. A machine running a supported OS, Windows, Linux or MacOS  
   See link for the supported OS  
   <https://dotnet.microsoft.com>
2. .NET Core v2.1 or better

### Using Mono

1. A machine running Mono v 5.04 or better
2. Follow the instructions for using the .NET Core version of the product

### TLS 1.0

In 2018 TLS 1.0 was deprecated. Some podcast suppliers, for reasons best known to themselves deliver podcasts over SSL encrypted connections and they have also forbidden the older TLS 1.0. Older versions of Windows need to be patched to use TLS 1.1 or TLS2.0.

If you see an error like this on Windows 7

Error in: Test Match Special

System.IO.IOException: Received an unexpected EOF or 0 bytes from the transport stream.

at System.Net.TlsStream.EndWrite(IAsyncResult asyncResult)

at System.Net.PooledStream.EndWrite(IAsyncResult asyncResult)

at System.Net.ConnectStream.WriteHeadersCallback(IAsyncResult ar)

The chances are that you do not have the TLS patch applied. It should be available here

Support for TLS System Default Versions included in the .NET Framework 3.5.1 on Windows 7 SP1 and Server 2008 R2 SP1

<https://support.microsoft.com/en-us/help/3154518/support-for-tls-system-default-versions-included-in-the-net-framework>

### MTP

MTP is currently not supported by the .NET Core version of the code.

Downloading and purging podcasts will work on all platforms. If you want to sync podcasts to an MTP device, such as a mobile phone, then you will need to use the .NET framework version of the utilities. Syncing to network drives will work just fine in the .NETCore version, its only MTP pathnames that will not work.

## Copying the distribution

Unzip all the files from the distribution ZIP into any folder.

You will need to create your own control XML file for your specific media files, an example file is provided in the distribution ZIP along with this document.

## Distribution contents

The distribution contains prebuild binaries for two different platforms.

### .NET Framework

If you want to use the .NET Framework then you only need the files in the folder called **net35**, if you are on Windows then this is probably the best version for you.

You run the utilities by executing one of the EXEs directly for example **DownloadPodcasts.exe**

### .NET Core

If you want to use the .NET Core version then these files can be found in the **netcoreapp2.1** folder. Ensure that you have .NET Core installed on your target machine by typing

The programs included with the Debian GNU/Linux system are free software;

the exact distribution terms for each program are described in the

individual files in /usr/share/doc/\*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent

permitted by applicable law.

Last login: Fri Apr 12 00:08:23 2019 from 192.168.0.26

pi@cherry-pi:~ $ dotnet --info

Host (useful for support):

Version: 2.2.0

Commit: 1249f08fed

.NET Core SDKs installed:

No SDKs were found.

.NET Core runtimes installed:

Microsoft.AspNetCore.All 2.2.0 [/opt/dotnet/shared/Microsoft.AspNetCore.All]

Microsoft.AspNetCore.App 2.2.0 [/opt/dotnet/shared/Microsoft.AspNetCore.App]

Microsoft.NETCore.App 2.2.0 [/opt/dotnet/shared/Microsoft.NETCore.App]

To install additional .NET Core runtimes or SDKs:

https://aka.ms/dotnet-download

If you don’t have .NET Core installed you will see something like this

dotnet PurgePodcasts.dll .\StephPodcasts.xml

'dotnet' is not recognized as an internal or external command,

operable program or batch file.

You can install them from here

<https://dotnet.microsoft.com/download>

You run the utilities by specifying the executable as a parameter to dotnet for example

dotnet DownloadPodcasts.dll controlfile.xml

The distribution contains the utilities in a binary form execution. The utilities (exe/dll files) and their supporting dlls can be copied to any folder. Before you can use the utilities you will need to setup your control file. An example control file is included in the distribution.

If you are a developer and want to write code using PodcastUtilities then you need to add a reference to PodcastUtilities.Common.dll, this is the only file you need from the distribution though you may find it useful to refer to the PodcastUtilities.Common.chm file for documentation on the API.

PodcastUtilities.Common.dll is designed to be used “headless”, it makes no calls to win forms, web forms or console output. In the .NET Core distribution PodcastUtilities.Common.dll is a .NET Standard 2.0 DLL, in the .NET Framework distribution it is built to use .NET 3.5

The source for the console applications: DownloadPodcasts, SyncPodcasts and PurgePodcasts provides an example of using the PodcastUtilities.Common assembly.

The source for PodcastUtilities is provided under the FreeBSD license, see the LICENSE.TXT file for more information.

# Installing Podcast Utilities from Chocolatey

You can also install PodcastUtilities from Chocolatey

## Prerequisites

### NET Framework/.NET Core

This is the same prerequisite as for installing from the distribution ZIP above\

### Chocolatey

Install Chocolatey from here  
<https://docs.chocolatey.org/en-us/choco/setup>

## Installing

Install podcast utilities for .NET Framework or .NET Core like this

choco install "podcastutilities"

or

choco install "podcastutilities-core"

The files are installed here

%PROGRAMDATA%\chocolatey\lib\podcastutilities-core

The chocolatey installer also creates a link to the user documentation, this file, on the desktop and start menu. Also the commands are added to the path.

### .NET Framework

You can type this command form anywhere; the shim is added to the path

downloadpodcasts controlfile.xml

### .NET Core

The commands are shimmed like this

If you type

Downloadpodcasts-core controlfile.xml

It is shimmed to be

C:\Program Files\dotnet\dotnet %PROGRAMDATA%\chocolatey\lib\podcastutilities-core\DownloadPodcasts.dll controlfile.xml

# DownloadPodcasts

DownloadPodcasts is the command to be used to download media from all the feed urls specified int he control file. You need to have internet connectivity for this command to work.

## Typical Usage

Use this command to download podcasts either directly to a media playing device attached to your computer or to download media to your computer and then use SyncPodcasts to copy a subset to your media player.

## Command Line

The command line to run DownloadPodcasts is

DownloadPodcasts <controlfile>

Where <controlfile> is the filename of the XML file to control how DownloadPodcasts will work. For example if your control file is called MyPhone.XML then the command line would be

DownloadPodcasts MyPhone.xml

# SyncPodcasts

SyncPodcasts is the command that will copy digital media from a source location to a destination.

## Typical Usage

SyncPodcasts is typically used when you have a large number of digital media files and you want to transfer a subset of these files to a removable drive. Only files that are not present in the destination are copied, any files that have been removed in the source are also removed in the destination.

## Command Line

The command line to run SyncPodcasts is

SyncPodcasts <controlfile>

Where <controlfile> is the filename of the XML file to control how SyncPodcasts will work. For example if your control file is called MyPhone.XML then the command line would be

SyncPodcasts MyPhone.xml

# PurgePodcasts

PurgePodcasts is the command that will remove old podcasts that have been downloaded

## Typical Usage

PurgePodcasts is typically used to remove old downloads. The number of days it takes to be considered to be old is configured in the control file on a fed by feed basis, you can also disable the purging in any feed.

## Command Line

The command line to run PurgePodcasts is

PurgePodcasts <controlfile>

Where <controlfile> is the filename of the XML file to control how PurgePodcasts will work. For example if your control file is called MyPhone.XML then the command line would be

PurgePodcasts MyPhone.xml

# GeneratePlaylist

GeneratePlaylist is the command that will create a playlist file from a folder structure

## Typical Usage

GeneratePlaylist is typically used when you have a large number of digital media files that you have copied to a location and you would like to generate a playlist for all the files.

## Command Line

The command line to run GeneratePlaylist is

GeneratePlaylist <controlfile>

Where <controlfile> is the filename of the XML file to control how GeneratePlaylist will work. For example if your control file is called MyPhone.XML then the command line would be

GeneratePlaylist MyPhone.xml

GeneratePlaylist is the action performed by SyncPodcasts after all the copying has been done.

# Using the utilities

This section covers how the utilities might be used together to manage podcasts

## Downloading to a PC and then synchronising to a device

You want to download podcasts using a windows PC (either downloading to the hard disk or a memory stick) and then sync to a device.

### Setting up

To setup for this environment copy all the files from the ZIP to a folder on your hard disk or on a memory stick.

Do remember that removable media such as memory sticks can be referred to by different drive letters on different machines, however you can use the “Computer Management” console to permanently set the drive letter for a removable media device on a given computer. You can launch the Computer Management console by typing “compmgmt.msc” into a command prompt



In Windows 7 select Action -> All Tasks -> Change Drive Letters and Paths

### Key elements in the control file

The global section of the control file looks like this

<podcasts version="1.0">

<global>

<sourceRoot>.\media</sourceRoot>

<destinationRoot>W:\Podcasts</destinationRoot>

<playlistFilename>podcasts.wpl</playlistFilename>

<playlistFormat>wpl</playlistFormat>

<freeSpaceToLeaveOnDestinationMB>1000</freeSpaceToLeaveOnDestinationMB>

<sortfield>name</sortfield>

<sortdirection>asc</sortdirection>

<maximumNumberOfConcurrentDownloads>20</maximumNumberOfConcurrentDownloads>

<feed>

<maximumDaysOld>31</maximumDaysOld>

<format>rss</format>

<namingStyle>pubdate\_etitle</namingStyle>

<downloadStrategy>high\_tide</downloadStrategy>

</feed>

</global>

* All downloads will be store in a subfolder call media.
* When synchronising my device will appear as drive W:
* I use WPL format playlists
* Leave at least 1GB on the download drive and the destination device
* Download 20 at a time
* Only download episodes published in the last month
* Name the files using the published date and the episode title
* After I have listened to an episode on my device, I want to be able to delete the episode on my PC and have the delete synchronised across to the device so I use “high\_tide” as a strategy, which is also why I want to have the published date in the file name

Feeds are generally in two groups. There are feeds where I want to listen to every episode and they are formatted this this

<podcast>

<feed>

<url>http://www.hanselminutes.com/hanselminutes\_MP3Direct.xml</url>

</feed>

<folder>Hanselminutes</folder>

<pattern>\*.mp3</pattern>

<number>-1</number>

</podcast>

* I do not specify deleteDownloadsDaysOld so PurgePodcasts will never delete an old episode I need to manually delete episodes I have listened to but I do not want to miss an episode so this is fine.
* I have used “high\_tide” strategy so when I have listened to an episode I delete it on my PC and SyncPodcasts will delete the episode on my device.
* I want to copy all the epicodes to my device so I set number to -1. For some feeds I limit the number to be copied to 3 for example, then SyncPOdcasts will copy the oldest 3 episodes to my device and as I delete them on my PC will keep the device topped up until there are no more available.
* You may need to change pattern if the feed uses a different format for its files.

Then there are feeds that I subscribe to that I just want to keep the last n episodes on tap but I may not get round to listening to them. These feeds are configured like this

<podcast>

<feed>

<url>http://downloads.bbc.co.uk/podcasts/worldservice/wbc/rss.xml</url>

<deleteDownloadsDaysOld>31</deleteDownloadsDaysOld>

</feed>

<folder>World Book Club</folder>

<pattern>\*.mp3</pattern>

<number>3</number>

</podcast>

* deleteDownloadsDaysOld is set to 31 so I PurgePodcasts will delete any episodes older than that.
* high\_tide strategy is used to the old episodes will not be downloaded again
* number is set to 3 so SyncPodcasts will ensure that I have the last 3 podcasts published in the last month available on my device, as time goes on these podcasts are automatically replaced with the current month’s episodes, a bit like TiVo for your podcasts.

### Workflow

I tend to get my podcasts once or twice a week. The workflow is

1. Attach my device to the PC
2. Run PurgePodcasts to automatically remove any old podcasts
3. Manually delete any episodes I have listened to
4. Run DownloadPodcasts to get all newly published podcasts to my PC
5. Run SyncPodcasts to get the desired subset of podcasts onto my device

## Downloading to a PC and then synchronising to a device – alternative strategy

This is similar to 7.1 but with a different approach to file management.

### Setting up

The setup for this strategy is exactly the same as 7.1.1.

### Key elements in the control file

The global section of the control file looks like this:

<podcasts version="1.0">

<global>

<sourceRootD:\Podcasts</sourceRoot>

<destinationRoot>F:\Podcasts</destinationRoot>

<playlistFilename>podcasts.wpl</playlistFilename>

<playlistFormat>wpl</playlistFormat>

<freeSpaceToLeaveOnDestinationMB>1000</freeSpaceToLeaveOnDestinationMB>

<sortfield>name</sortfield>

<sortdirection>desc</sortdirection>

<maximumNumberOfConcurrentDownloads>20</maximumNumberOfConcurrentDownloads>

<feed>

<maximumDaysOld>60</maximumDaysOld>

<format>rss</format>

<namingStyle>pubdate\_etitle</namingStyle>

<downloadStrategy>high\_tide</downloadStrategy>

</feed>

</global>

The only real difference with the previous approach is that I never delete the podcasts on my PC so I use a descending sort to synchronise just the most recent podcasts. This does mean I need to make sure I sync enough episodes (and keep listening to them) otherwise the older ones will drop off my device.

I have some feeds that I don’t want to miss, and therefore I make sure I synchronise a reasonable number of them so I don’t lose any.

<podcast>

<feed>

<url>http://feeds.feedburner.com/HanselminutesCompleteMP3</url>

</feed>

<folder>Hanselminutes</folder>

<pattern>\*.mp3</pattern>

<number>10</number>

</podcast>

(Note that I never use -1 for number as I don’t delete the podcasts from my PC so would sync **all** episodes – good though Hanselminutes is, I don’t want 200+ of them on my phone!)

The rest are just in priority order (SyncPodcasts uses the order in the control file to determine priority), with numbers of 4 or 5.

### Workflow

As in the previous strategy I download podcasts a couple of times each week, but the workflow is simpler as nothing is deleted:

1. Attach my device to the PC
2. Run DownloadPodcasts to get all newly published podcasts to my PC
3. Run SyncPodcasts to get the desired subset of podcasts onto my device

## Download directly to a flash drive or media player

You want to download podcasts directly to a memory device for example so that you always have a selection of podcasts on your player but are not bothered about keeping episodes.

### Setting up

This is the same as the setup for 7.1, we copy PodcastUtilities to the root folder of the media player / flash drive.

### Key elements in the control file

The global section looks like this

<podcasts version="1.0">

<global>

<sourceRoot>.\PODCASTS</sourceRoot>

<destinationRoot>W:\Podcasts</destinationRoot>

<playlistFilename>podcasts.wpl</playlistFilename>

<playlistFormat>wpl</playlistFormat>

<freeSpaceToLeaveOnDestinationMB>100</freeSpaceToLeaveOnDestinationMB>

<freeSpaceToLeaveOnDownloadMB>100</freeSpaceToLeaveOnDownloadMB>

<sortfield>name</sortfield>

<sortdirection>asc</sortdirection>

<maximumNumberOfConcurrentDownloads>10</maximumNumberOfConcurrentDownloads>

<retryWaitInSeconds>15</retryWaitInSeconds>

<feed>

<maximumDaysOld>80</maximumDaysOld>

<deleteDownloadsDaysOld>80</deleteDownloadsDaysOld>

<format>rss</format>

<namingStyle>pubdate\_folder\_title\_url</namingStyle>

<downloadStrategy>high\_tide</downloadStrategy>

</feed>

</global>

The interesting elements are, sourceRoot is a relative pathname (remember we will have the software and the control file are in the root folder of the media player, so the podcasts will be downloaded straight to this folder. destinationRoot can be anything, it is not used.

We do use a playlist however it may be that you will just use the file system.

Some older and cheaper media players are particularly slow to write to, if this is the case you may want to limit the number of concurrent downloads to 10 or less and you may also want to adjust the retryWaitISeconds to 15 or greater.

In this strategy we are just going to keep a pool of podcasts on the device, we will remove old ones and add new ones automatically. You will need to adjust the maximumDaysOld and deleteDownloadsDaysOld depending on the number of podcasts you are following and the size of the memory in your device. The more memory you have the larger the number of days you can specify.

Then each of the podcasts that I subscribe to are done loke this

<podcast>

<feed>

<url>http://downloads.bbc.co.uk/podcasts/radio4/fooc/rss.xml</url>

</feed>

<folder>From Our Own Correspondent</folder>

<pattern>\*.mp3</pattern>

<number>5</number>

</podcast>

The number value does not matter as we do not sync these podcasts, the number we keep will depend upon the deleteDownloadsDaysOld and the frequency the podcasts are updated. If you want to prevent the automatic deletion of episodes then override this value in a podcast and set it to be a high value such as 999, however the real purpose of this strategy is to set all of the feeds to automatically delete.

### Workflow

In this strategy I tend to download less often only needing to when I have started to run out of episodes that I have not already heard.

1. Attach my device to the PC
2. Run PurgePodcasts to automatically remove any old podcasts
3. Run DownloadPodcasts to get all newly published podcasts to my device

# Control File Format

All the utilities make use of a control file to control how they operate. The format of the control file is XML, so all <elements> must have a closing element </elements>The root element of the file is <podcasts>

## Example Control File

The control file is an XML file and must be a legal XML file, in that elements must have a start and an end. The elements for example <sourceRoot> must be specified in the correct upper and lower case as shown here. This is an example of a control file

<podcasts version="1.0">

<global>

<sourceRoot>.\downloads</sourceRoot>

<destinationRoot>W:\Podcasts</destinationRoot>

<playlistFilename>podcasts.wpl</playlistFilename>

<playlistFormat>wpl</playlistFormat>

<freeSpaceToLeaveOnDestinationMB>2000</freeSpaceToLeaveOnDestinationMB>

<sortfield>name</sortfield>

<sortdirection>asc</sortdirection>

<maximumNumberOfConcurrentDownloads>20</maximumNumberOfConcurrentDownloads> <feed>

<maximumDaysOld>31</maximumDaysOld>

<namingStyle>pubdate\_etitle</namingStyle>

<downloadStrategy>high\_tide</downloadStrategy>

<format>rss</format>

</feed>

</global>

<podcast>

<feed>

<url>http://downloads.bbc.co.uk/podcasts/fivelive/tms/rss.xml</url>

<deleteDownloadsDaysOld>31</deleteDownloadsDaysOld>

<maximumNumberOfDownloadedItems>5</maximumNumberOfDownloadedItems>

</feed>

<folder>Test Match Special</folder>

<pattern>\*.mp3</pattern>

<number>-1</number>

</podcast>

<podcast>

<feed>

<url>http://www.hanselminutes.com/hanselminutes\_MP3Direct.xml</url>

</feed>

<folder>Hanselminutes</folder>

<pattern>\*.mp3</pattern>

<number>-1</number>

</podcast>

<podcast>

<folder>From Our Own Correspondent</folder>

<pattern>\*.mp3</pattern>

<number>5</number>

</podcast>

</podcasts>

If this control file was used with the SyncPodcasts utility then it would copy all the mp3 podcasts in “Test Match Special” and “Hansleminutes” folders followed by the first five mp3 files in the folder “From Our Own Correspondant” The source files are in a folder called downloads in the current folder and the destination folder is in “w:\Podcasts”. A WPL playlist will be generated and at least 2GB of space will be left on the drive.

If it was used with DownloadPodcasts then it would download all the episodes of Test Match Special and Hansleminutes published in the last 31 days. It would attempt to download 20 episodes at a time.

### Global Section

There is only one global section and it has settings that can only be set once or default values that can be overridden for each podcast

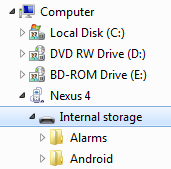
#### sourceRoot

This is the path to the source media files. It can be a relative or absolute path. The sourceRoot is ued by DownloadPodcasts as the root folder to download episodes to, SyncPodcasts uses it as the place to read media from.

Some phones, for example many newer Android phones and Windows Phone devices do not appear as a drive letter. Instead they may support MTP (Media Transfer Protocol). If that is the case you can prefix a path by “MTP:”, followed by the device name to address these devices, for example

<destinationRoot>MTP:\Nexus 4\Internal storage\Podcasts</destinationRoot>

Look in explorer, under portable devices, to find the name of the device and the storage volume.



If it is a relative path (for example “.\media”) it is relative to the current working directory, usually this is the current directory if you are running the utility from a cmd prompt.

An absolute path (for example “m:\media”) will remain unaffected by where the utility is run from. Do remember that removable media can be referred to by different drive letters on different machines, however you can use the “Computer Management” console to permanently set the drive letter for a removable media device on a given computer. You can launch the Computer Management console by typing “compmgmt.msc” into a command prompt



In Windows 7 select Action -> All Tasks -> Change Drive Letters and Paths

#### destinationRoot

SyncPodcasts uses this as the path that we shall copy to, GeneratePlaylists uses it as the root folder to find media in. This must be an absolute pathname for example p:\media. GeneratePlaylist

MTP paths can be used a as a destination.

#### playlistFilename

SyncPodcasts and GeneratePlaylist uses this as the name of the file to write the playlist that contains all the media in the destination. The playlist contains all files in the destination not just the files copied in this run.

#### playlistFormat

The format the playlist should be in. Possible values are m3u, wpl or asx, (upper or lower case). WPL format is a windows playlist for Windows Media Player and Windows Mobile based devices, M3U and ASX are formats used by many media players. Generate the playlist format that works with your player, also try adjusting the playlistPathSeparator..

#### playlistPathSeparator

Separator to use between elements in a playlist item pathname. Default is “\”. “\” works well on Windows however “/” works on Windows, Linux and Android.

#### freeSpaceToLeaveOnDestinationMB

The space in MB to leave in the device, for example 2000 is 2GB.

When using SyncPodcasts copying will stop when there is less that this space left, however if there is less that this space left when SyncPodcasts is run then files will not be deleted to free up space.

When using DownloadPodcasts downloading will stop when there is less that this space left.

#### sortfield

The field to be used to sort the media files. If we want to copy the “first” 5 files of a given podcast then this setting is used to determine which files are first.

If the value of this setting is creationtime (upper or lower case) then the files are sorted by the date and time the file was created otherwise any other value means they are sorted by name.

Many podcasts incorporate a numbering system into the filenames so that using the filename will work fine however sometimes they either have no number or it is a random id in those cases you may want to sort the files on the bases of the date the file was downloaded.

If you use the DownloadPodcasts utility to download files then you can use the namingStyle setting to control the filename to ensure that the filename can be sorted by publishing date.

This setting can be overridden on a podcast by podcast basis.

#### deleteEmptyFolder

PurgePodcasts uses this field is used to control the removal of empty folders found in the source. SyncPodcasts uses this field to control the removal of empty folders from the destination. TRUE means remove the folder FALSE means leave empty folders.

A folder is considered to be empty if it only contains downloaded files to be deleted or generated files such as thumbs.db. If you find a folder does not seem to be deleted then you may need to manually delete state.xml.

#### sortdirection

This field is also used to control the selection of media files, possible values are desc or asc , (upper or lower case) for descending or ascending sort order.

This setting can be overridden on a podcast by podcast basis.

#### maximumNumberOfConcurrentDownloads

This element is only used by DownloadPodcasts and controls how many downloads are running at the same time. Your computer or the server that holds the podcasts may limit the number of connections it will accept.

#### retryWaitInSeconds

This element can be used to resolve any file locking issues when using slow flash drives. If you are running a large number of concurrent downloads in the same feed then all the downloads will attempt to update the high tide mark in the state file. You may see a warning message saying that the state file was locked and the downloader will retry. If after a number of retries the file is still locked then an error will be thrown.

This element is the number of seconds to wait before retrying writing to the state file. You may find a higher number for example 15 or 20 seconds will help on slow flash drives, you may also find that reducing the number of concurrent downloads will help.

#### feed section

This section is only used by DownloadPodcasts and PurgePodcasts

#### format

The only supported podcast feed format is RSS

#### downloadStrategy

This is the mechanism that will be used to download episodes in a feed

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| all | All episodes in the feed will be downloaded, if a feed keeps all the episodes that have been published in it then all the episodes will be downloaded, if you delete an download then it will be replaced when DownloadPodcasts is next run |
| latest | Only the latest episode in a feed is downloaded |
| high\_tide | When first run on a new feed all the episodes are downloaded, subsequently only new episodes are downloaded. This is achieved by recording the last episode successfully downloaded in a state.xml file and only downloading episodes that have been published since this high tide date.  You can reset the high tide value by deleting the state.xml file and all episodes will then be downloaded again. |

#### namingStyle

This is the style to use to name the file downloaded from a feed. The possible values are

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| etitle | The episode title, as specified in the feed will be used. Any characters that cannot be used in a filename will be replaced with an underscore. |
| pubdate\_etitle | Published date in the YYYY MM DD HH MM (to enable the files to be sorted by published date) followed by the episode title, as specified in the feed. Any characters that cannot be used in a filename will be replaced with an underscore. |
| pubdate\_url | Published date in the YYYY MM DD HH MM (to enable the files to be sorted by published date) followed by the filename element from the end of the URL |
| pubdate\_title\_url | Published date in the YYYY MM DD HH MM (to enable the files to be sorted by published date) followed by the feed title (the folder used to store the episodes) followed by the filename element from the end of the URL |
| pubdate\_folder\_title\_url | The files are named in the same manner as pubdate\_title\_url and then the files are placed in a folder named after the month the episode was published in the form YYYY\_MM. Effectively this will group podcasts from the same month into a separate folder. |
| <anything else> | The filename element at the end of the URL for this episode |

#### maximumDaysOld

This can be used to prevent old episodes from being downloaded. The age of a podcast is calculated by comparing the current date time with the published date time in the feed.

#### deleteDownloadsDaysOld

This element is only used by PurgePodcasts and is used to remove old podcasts episodes that were published more than the specified number of days ago.

Please note that PurgePodcasts can only determine the date an episode was published if you use a namingStyle (see above) that records the publish date in the filename, otherwise the date the file was created on your local disk will be used.

This element is optional, if it is omitted or has a value of zero then purging will not be done for the specified feed.

#### maximumNumberOfDownloadedItems

This element is only used by DownloadPodcasts and is used to determine how many items to keep in the download cache. For example if there are 10 episodes available for a feed and 2 of them are already in the cache and this value is set to be 5 then 3 episodes will be downloaded. After they have been downloaded no more episodes will be downloaded until some have been deleted.

This element can be used to throttle the number of downloads, to throttle the number of items that are synced use podcast/number (see below)

This element is optional, if it is omitted then all available items for a given feed will be downloaded, subject to the downloadStrategy (see above).

#### diagnostics section

This section is optional but if present is used by the utilities to control mechanisms for helping find issues with the software.

#### retainTempFiles

This setting is used by DownloadPodcasts. Possible values are true or false. If true is specified then the RSS feed xml for each feed is saved to disk before it is processed. The file is stored in the same folder as the podcast episodes in a file called “last\_download\_feed.xml”, only the last XML downloaded is saved. If the value is anything other than true then nothing is saved. In either case the downloading of podcasts is unaffected.

#### outputLevel

Possible values are verbose or none. If it is set to verbose then the console applications will display diagnostic information. If it is set to none (the default) then no diagnostic output is displayed. This setting only affects the console applications the PodcastUtilities.Common assembly for API developers never writes to the console or produces any output of any sort.

#### postdownloadcommand section

This section is optional but if present is used to specify a default external commandto be run after the successful completion of a download. The command can be used as is or any of the parts can be overridden by podcasts

#### command

This is the command to be run. It should be an EXE or CMD etc that can be run on your computer. This is a tokenised commands where tokens are surrounded by braces {} The tokens that are supported are as follows

|  |  |
| --- | --- |
| **Token** | **Meaning** |
| Downloadfullpath | The full pathname to the downloaded file |
| Downloadroot | The root folder for downloads. This will be the same as sourceRoot |
| Downloadfolder | This is the full pathname to the folder that contains the downloaded file. |
| Exefolder | The full pathname to the folder that contains the exe that did the download. |

#### arguments

Arguments to be passed to the command. Please note that if the pathname has spaces in it you may need to surround and filenames or pathname in quotes. The tokens used in the command can be used here.

#### workingdirectory

This is the current directory for the command to be run. The tokens used in the command can be used here.

### Podcast Section

After the global section there can be any number of podcast sections. Each section describes one folder of media files to be synced or podcasts to be downloaded.

Each section must have a value for folder, pattern and number and can optionally override sortfield and sortdirection.

If you want to use DownloadPodcasts to download the episodes then you will need a feed/url element, you can optionally override any other element in the feed section of the global section.

#### folder

SyncPodcasts uses this as the folder relative to the source that will be used to copy from and relative to the destination to copy to.

For example if the sourceRoot is “C:\Podcasts” and the destinationRoot is “X:\Media” and folder is “fred” then we will copy from “C:\Podcasts\fred” to “X:\Media\fred”

GeneratePlaylists uses this as the folder to find media to add to the playlist.

It is also used by DownloadPodcasts as the folder relative to the sourceRoot to download files to and also as the title of the podcast feed if the title is to be included in the episode file name using the namingStyle element.

#### pattern

SyncPodcasts uses this as the filename pattern to look for when copying files, for example \*.mp3 will match all the mp3 files.

#### number

SyncPodcasts uses this as the number of files to copy. -1 means that all file in the source folder will be copued.

This element can be used to throttle the number of items that are synced, to throttle the number of items downloaded use feed/maximumNumberOfDownloadedItems (see above)

#### sortfield

This is the same as the setting in the global section, if it is missing then the one from the global section is used

#### sortdirection

This is the same as the setting in the global section, if it is missing then the one from the global section is used

#### deleteEmptyFolder

This is the same as the setting in the global section, if it is missing then the one from the global section is used.

#### postdownloadcommand section

This section is only used by DownloadPodcasts

This is the same as the setting in the global section, if it is missing then no command is run for this podcast. If the section is present but empty (see the example podcast file) then the command specified in the global section is run. Any or all of the parts of the global section can be overridden for each podcast by specifying the element here.

#### feed section

This section is only used by DownloadPodcasts and PurgePodcasts

#### url

This is the url to the podcast feed, often this will be displayed as “RSS Feed” on a web site. If this is not present then DownloadPodcasts will ignore the podcast element.

#### format

This is the same as the setting in the global section, if it is missing then the one from the global section is used

#### downloadStrategy

This is the same as the setting in the global section, if it is missing then the one from the global section is used

#### namingStyle

This is the same as the setting in the global section, if it is missing then the one from the global section is used

#### maximumDaysOld

This is the same as the setting in the global section, if it is missing then the one from the global section is used

#### deleteDownloadsDaysOld

This is the same as the setting in the global section, if it is missing then the one from the global section is used

#### maximumNumberOfDownloadedItems

This is the same as the setting in the global section, if it is missing then the one from the global section is used