MPRIS2 Documentation

Release 0.9.3

hugosenari

December 04, 2016

Contents

This is a copy of mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/

That also works as python lib.

CHAPTER 1

Version 2.2

CHAPTER 2

About

The Media Player Remote Interfacing Specification is a standard D-Bus interface which aims to provide a common programmatic API for controlling media players.

It provides a mechanism for compliant media players discovery, basic playback and media player state control as well as a tracklist interface which is used to add context to the current item.

CHAPTER 3

Changes

Changes (Permalink)

From 2.1 to 2.2:

- Added the optional Fullscreen and CanSetFullscreen properties to the org.mpris.MediaPlayer2 interface.
- The path /org/mpris/MediaPlayer2/TrackList/NoTrack now represents “no track” where required in the org.mpris.MediaPlayer2.TrackList interface (since empty paths are not allowed by D-Bus).
- The suggested unique instance identifier no longer violates the D-Bus specification by begining with a digit.

From 2.0 to 2.1:

- Added the optional org.mpris.MediaPlayer2.Playlists interface.

CHAPTER 4

Bus Name Policy

Each media player *must* request a unique bus name which begins with *org.mpris.MediaPlayer2*. For example:

- org.mpris.MediaPlayer2.audacious
- org.mpris.MediaPlayer2.vlc
- org.mpris.MediaPlayer2.bmp
- org.mpris.MediaPlayer2.xmms2

This allows clients to list available media players (either already running or which can be started via D-Bus activation)

In the case where the media player allows multiple instances running simultaneously, each additional instance should request a unique bus name, adding a dot and a unique identifier to its usual bus name, such as one based on a UNIX process id. For example, this could be:

- org.mpris.MediaPlayer2.vlc.7389

Note: According to the D-Bus specification, the unique identifier "must only contain the ASCII characters '[A-Z][a-z][0-9]_-'" and "must not begin with a digit".

CHAPTER 5

Entry point

The media player *must* expose the */org/mpris/MediaPlayer2* object path, which *must* implement the following interfaces:

- org.mpris.MediaPlayer2
- org.mpris.MediaPlayer2.Player

The */org/mpris/MediaPlayer2* object may implement the org.mpris.MediaPlayer2.TrackList interface.

The */org/mpris/MediaPlayer2* object may implement the org.mpris.MediaPlayer2.Playlists interface.

CHAPTER 6

The PropertiesChanged signal

The MPRIS uses the org.freedesktop.DBus.Properties.PropertiesChanged signal to notify clients of changes in the media player state. If a client implementation uses D-Bus bindings which do not support this signal, then it should connect to it manually. If a media player implementation uses D-Bus bindings which do not support this signal, then it should send it manually

CHAPTER 7

Corrections

2010-09-26: Added EmitsChangedSignal annotation to Volume property on the Player interface.

2011-01-26: Added PlaylistChanged signal to the Playlists interface.

CHAPTER 8

Interfaces

- org.mpris.MediaPlayer2
- org.mpris.MediaPlayer2.Player
- org.mpris.MediaPlayer2.Playlists
- org.mpris.MediaPlayer2.TrackList

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

CHAPTER 10

Require:

To use this lib you need: Python dbus

CHAPTER 11

Contents:

11.1 MPRIS2 Interfaces

class mpris2.**Interfaces**

This class contains the constants defined at index of MPRIS2 definition:

Interfaces:

- **MEDIA_PLAYER** 'org.mpris.MediaPlayer2'
- **TRACK_LIST** 'org.mpris.MediaPlayer2.TrackList'
- **PLAYER** 'org.mpris.MediaPlayer2.Player'
- **PLAYLISTS** 'org.mpris.MediaPlayer2.Playlists'
- **PROPERTIES** 'org.freedesktop.DBus.Properties'

Signals:

- **SIGNAL** 'PropertiesChanged'

Objects:

- **OBJECT_PATH** '/org/mpris/MediaPlayer2'

class mpris2.**MediaPlayer2**(**args*, ***kw*)

Interface for MediaPlayer2 (org.mpris.MediaPlayer2)

CanQuit

Returns

Read only Inject attrs from decorator at new object then return obje

When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

If false, calling Quit will have no effect, and may raise a NotSupported error. If true, calling Quit will cause the media application to attempt to quit (although it may still be prevented from quitting by the user, for example).

CanRaise

Returns

Read only If false, calling Raise will have no effect, and may raise a NotSupported error. If true, calling Raise will cause the media application to attempt to bring its user interface to the front, although it may be prevented from doing so (by the window manager, for example).

When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

CanSetFullscreen

Returns

Read only If false, attempting to set Fullscreen will have no effect, and may raise an error. If true, attempting to set Fullscreen will not raise an error, and (if it is different from the current value) will cause the media player to attempt to enter or exit fullscreen mode.

This property is optional. Clients should handle its absence gracefully.

When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Added in 2.2.

..note:: Note that the media player may be unable to fulfil the request. In this case, the value will not change. If the media player knows in advance that it will not be able to fulfil the request, however, this property should be false.

DesktopEntry

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The basename of an installed .desktop file which complies with the Desktop entry specification, with the '.desktop' extension stripped.

Example: The desktop entry file is '/usr/share/applications/vlc.desktop', and this property contains 'vlc'

This property is optional. Clients should handle its absence gracefully

Fullscreen

Returns

Read Write Whether the media player is occupying the fullscreen.

This property is optional. Clients should handle its absence gracefully.

When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

This is typically used for videos. A value of true indicates that the media player is taking up the full screen.

Media center software may well have this value fixed to true

If CanSetFullscreen is true, clients may set this property to true to tell the media player to enter fullscreen mode, or to false to return to windowed mode.

If CanSetFullscreen is false, then attempting to set this property should have no effect, and may raise an error. However, even if it is true, the media player may still be unable to fulfil the request, in which case attempting to set this property will have no effect (but should not raise an error).

Added in 2.2.

HasTrackList

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Indicates whether the /org/mpris/MediaPlayer2 object implements the org.mpris.MediaPlayer2.TrackList interface.

Identity
Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

If false, calling Raise will have no effect, and may raise a NotSupported error. If true, calling Raise will cause the media application to attempt to bring its user interface to the front, although it may be prevented from doing so (by the window manager, for example).

PropertiesChanged
Parameters:

- **args - list** unnamed parameters passed by dbus signal
- **kw - dict** named parameters passed by dbus signal

Every time that some property change, signal will be called

Quit
Causes the media player to stop running.

The media player may refuse to allow clients to shut it down. In this case, the CanQuit property is false and this method does nothing.

..note:: Media players which can be D-Bus activated, or for which there is no sensibly easy way to terminate a running instance (via the main interface or a notification area icon for example) should allow clients to use this method. Otherwise, it should not be needed.

If the media player does not have a UI, this should be implemented

Raise
Brings the media player's user interface to the front using any appropriate mechanism available.

The media player may be unable to control how its user interface is displayed, or it may not have a graphical user interface at all. In this case, the Identity property is false and this method does nothing.

SupportedMimeTypes
Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The mime-types supported by the media player.

Mime-types should be in the standard format (eg: audio/mpeg or application/ogg).

Note: This is important for clients to know when using the editing capabilities of the Playlist interface, for example.

SupportedUriSchemes
Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The URI schemes supported by the media player.

This can be viewed as protocols supported by the player in almost all cases. Almost every media player will include support for the 'file' scheme. Other common schemes are 'http' and 'rtsp'.

Note that URI schemes should be lower-case.

Note: This is important for clients to know when using the editing capabilities of the Playlist interface, for example.

class mpris2.**Player**(**args*, ***kw*)

This interface implements the methods for querying and providing basic control over what is currently playing.

CanControl

Returns

Read only The org.freedesktop.DBus.Properties.PropertiesChanged signal is not emitted when this property changes.

Whether the media player may be controlled over this interface.

This property is not expected to change, as it describes an intrinsic capability of the implementation.

If this is false, clients should assume that all properties on this interface are read-only (and will raise errors if writing to them is attempted); all methods are not implemented and all other properties starting with 'Can' are also false.

CanGoNext

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether the client can call the Next method on this interface and expect the current track to change.

If CanControl is false, this property should also be false.

CanGoPrevious

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether the client can call the Previous method on this interface and expect the current track to change.

If CanControl is false, this property should also be false.

CanPause

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether playback can be paused using Pause or PlayPause.

Note that this is an intrinsic property of the current track: its value should not depend on whether the track is currently paused or playing. In fact, if playback is currently paused (and CanControl is true), this should be true.

If CanControl is false, this property should also be false.

CanPlay

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether playback can be started using Play or PlayPause.

Note that this is related to whether there is a 'current track': the value should not depend on whether the track is currently paused or playing. In fact, if a track is currently playing CanControl is true), this should be true.

If CanControl is false, this property should also be false.

CanSeek

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether the client can control the playback position using Seek and SetPosition. This may be different for different tracks.

If CanControl is false, this property should also be false.

LoopStatus

Returns

Read/Write When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The current loop / repeat status

May be:

- 'None' if the playback will stop when there are no more tracks to play
- 'Track' if the current track will start again from the begining once it has finished playing
- 'Playlist' if the playback loops through a list of tracks

This property is optional, and clients should deal with NotSupported errors gracefully.

If CanControl is false, attempting to set this property should have no effect and raise an error.

MaximumRate

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The maximum value which the Rate property can take. Clients should not attempt to set the Rate property above this value.

This value should always be 1.0 or greater.

Metadata

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The metadata of the current element.

If there is a current track, this must have a 'mpris:trackid' entry at the very least, which contains a string that uniquely identifies this track.

See the type documentation for more details.

MinimumRate

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The minimum value which the Rate property can take. Clients should not attempt to set the Rate property below this value.

Note that even if this value is 0.0 or negative, clients should not attempt to set the Rate property to 0.0.

This value should always be 1.0 or less.

Next

Skips to the next track in the tracklist.

If there is no next track (and endless playback and track repeat are both off), stop playback.

If playback is paused or stopped, it remains that way.

If CanGoNext is false, attempting to call this method should have no effect.

OpenUri

Parameters:

- **Uri - s (Uri)** Uri of the track to load. Its uri scheme should be an element of the org.mpris.MediaPlayer2.SupportedUriSchemes property and the mime-type should match one of the elements of the org.mpris.MediaPlayer2.SupportedMimeTypes.

Opens the Uri given as an argument

If the playback is stopped, starts playing

If the uri scheme or the mime-type of the uri to open is not supported, this method does nothing and may raise an error. In particular, if the list of available uri schemes is empty, this method may not be implemented.

Clients should not assume that the Uri has been opened as soon as this method returns. They should wait until the mpris:trackid field in the Metadata property changes.

If the media player implements the TrackList interface, then the opened track should be made part of the tracklist, the org.mpris.MediaPlayer2.TrackList.TrackAdded or org.mpris.MediaPlayer2.TrackList.TrackListReplaced signal should be fired, as well as the org.freedesktop.DBus.Properties.PropertiesChanged signal on the tracklist interface.

Pause

Pauses playback.

If playback is already paused, this has no effect.

Calling Play after this should cause playback to start again from the same position.

If CanPause is false, attempting to call this method should have no effect.

Play

Starts or resumes playback.

If already playing, this has no effect.

If there is no track to play, this has no effect.

If CanPlay is false, attempting to call this method should have no effect.

PlayPause

Pauses playback.

If playback is already paused, resumes playback.

If playback is stopped, starts playback.

If CanPause is false, attempting to call this method should have no effect and raise an error.

PlaybackStatus

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The current playback status.

May be 'Playing', 'Paused' or 'Stopped'.

Position

Returns

Read only The org.freedesktop.DBus.Properties.PropertiesChanged signal is not emitted when this property changes.

The current track position in microseconds, between 0 and the 'mpris:length' metadata entry (see Metadata).

Note: If the media player allows it, the current playback position can be changed either the SetPosition method or the Seek method on this interface. If this is not the case, the CanSeek property is false, and setting this property has no effect and can raise an error.

If the playback progresses in a way that is inconstistant with the Rate property, the Seeked signal is emited.

Previous

Skips to the previous track in the tracklist.

If there is no previous track (and endless playback and track repeat are both off), stop playback.

If playback is paused or stopped, it remains that way.

If CanGoPrevious is false, attempting to call this method should have no effect.

PropertiesChanged

Parameters

- **args - list** unnamed parameters passed by dbus signal
- **kw - dict** named parameters passed by dbus signal

Every time that some property change, signal will be called

Rate

Returns

Read/Write When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The current playback rate.

The value must fall in the range described by MinimumRate and MaximumRate, and must not be 0.0. If playback is paused, the PlaybackStatus property should be used to indicate this. A value of 0.0 should not be set by the client. If it is, the media player should act as though Pause was called.

If the media player has no ability to play at speeds other than the normal playback rate, this must still be implemented, and must return 1.0. The MinimumRate and MaximumRate properties must also be set to 1.0.

Not all values may be accepted by the media player. It is left to media player implementations to decide how to deal with values they cannot use; they may either ignore them or pick a 'best fit' value. Clients are recommended to only use sensible fractions or multiples of 1 (eg: 0.5, 0.25, 1.5, 2.0, etc).

`Seek`

Parameters:

- **Offset - x (Time_In_Us)** The number of microseconds to seek forward.

Seeks forward in the current track by the specified number of microseconds.

A negative value seeks back. If this would mean seeking back further than the start of the track, the position is set to 0.

If the value passed in would mean seeking beyond the end of the track, acts like a call to Next.

If the CanSeek property is false, this has no effect.

`Seeked`

Parameters:

- **Position - x (Time_In_Us)** The new position, in microseconds.

Indicates that the track position has changed in a way that is inconsistant with the current playing state.

When this signal is not received, clients should assume that:

- When playing, the position progresses according to the rate property.
- When paused, it remains constant.

This signal does not need to be emitted when playback starts or when the track changes, unless the track is starting at an unexpected position. An expected position would be the last known one when going from Paused to Playing, and 0 when going from Stopped to Playing.

`SetPosition`

Parameters

- **TrackId - o (Track_Id)** The currently playing track's identifier.

  If this does not match the id of the currently-playing track, the call is ignored as 'stale'.

- **Position - x (Time_In_Us)** Track position in microseconds.

  This must be between 0 and <track_length>.

Sets the current track position in microseconds.

If the Position argument is less than 0, do nothing.

If the Position argument is greater than the track length, do nothing.

If the CanSeek property is false, this has no effect.

`Shuffle`

Returns

Read/Write When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

A value of false indicates that playback is progressing linearly through a playlist, while true means playback is progressing through a playlist in some other order.

This property is optional, and clients should deal with NotSupported errors gracefully.

If CanControl is false, attempting to set this property should have no effect and raise an error.

`Stop`

Stops playback.

If playback is already stopped, this has no effect.

Calling Play after this should cause playback to start again from the beginning of the track.

If CanControl is false, attempting to call this method should have no effect and raise an error.

Volume

Returns

Read/Write When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The volume level.

When setting, if a negative value is passed, the volume should be set to 0.0.

If CanControl is false, attempting to set this property should have no effect and raise an error.

class mpris2.**Playlists**(**args*, ***kw*)

Provides access to the media player's playlists.

Since D-Bus does not provide an easy way to check for what interfaces are exported on an object, clients should attempt to get one of the properties on this interface to see if it is implemented.

ActivatePlaylist

Parameters:

- **PlaylistId - o** The id of the playlist to activate.

Starts playing the given playlist.

Note that this must be implemented. If the media player does not allow clients to change the playlist, it should not implement this interface at all.

It is up to the media player whether this completely replaces the current tracklist, or whether it is merely inserted into the tracklist and the first track starts. For example, if the media player is operating in a 'jukebox' mode, it may just append the playlist to the list of upcoming tracks, and skip to the first track in the playlist.

ActivePlaylist

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The currently-active playlist.

If there is no currently-active playlist, the structure's Valid field will be false, and the Playlist details are undefined.

Note that this may not have a value even after ActivatePlaylist is called with a valid playlist id as ActivatePlaylist implementations have the option of simply inserting the contents of the playlist into the current tracklist.

GetPlaylists

Parameters:

- **Index - u** The index of the first playlist to be fetched (according to the ordering).
- **MaxCount - u** The maximum number of playlists to fetch.
- **Order - s (Playlist_Ordering)** The ordering that should be used.
- **ReverseOrder - b** Whether the order should be reversed.

Returns

- **Playlists - a(oss) (Playlist_List)** A list of (at most MaxCount) playlists.

Gets a set of playlists.

Orderings

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The avaislable orderings. At least one must be offered.

PlaylistChanged

Parameters

- **Playlist - (oss) (Playlist)** The playlist whose details have changed.

Indicates that the name or icon for a playlist has changed.

Note that, for this signal to operate correctly, the id of the playlist must not change when the name changes.

PlaylistCount

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The number of playlists available.

class mpris2.**TrackList**(**args*, ***kw*)

Interface for TrackList (org.mpris.MediaPlayer2.TrackList)

Provides access to a short list of tracks which were recently played or will be played shortly. This is intended to provide context to the currently-playing track, rather than giving complete access to the media player's playlist.

Example use cases are the list of tracks from the same album as the currently playing song or the Rhythmbox play queue.

Each track in the tracklist has a unique identifier. The intention is that this uniquely identifies the track within the scope of the tracklist. In particular, if a media item (a particular music file, say) occurs twice in the track list, each occurrence should have a different identifier. If a track is removed from the middle of the playlist, it should not affect the track ids of any other tracks in the tracklist.

As a result, the traditional track identifiers of URLs and position in the playlist cannot be used. Any scheme which satisfies the uniqueness requirements is valid, as clients should not make any assumptions about the value of the track id beyond the fact that it is a unique identifier.

Note that the (memory and processing) burden of implementing the TrackList interface and maintaining unique track ids for the playlist can be mitigated by only exposing a subset of the playlist when it is very long (the 20 or so tracks around the currently playing track, for example). This is a recommended practice as the tracklist interface is not designed to enable browsing through a large list of tracks, but rather to provide clients with context about the currently playing track.

AddTrack

Parameters:

- **Uri - s (Uri)** The uri of the item to add. Its uri scheme should be an element of the org.mpris.MediaPlayer2.SupportedUriSchemes property and the mime-type should match one of the elements of the org.mpris.MediaPlayer2.SupportedMimeTypes
- **AfterTrack - o (Track_Id)** The identifier of the track after which the new item should be inserted. The path /org/mpris/MediaPlayer2/TrackList/NoTrack indicates that the track should be inserted at the start of the track list.
- **SetAsCurrent - b** Whether the newly inserted track should be considered as the current track. Setting this to trye has the same effect as calling GoTo afterwards.

Adds a URI in the TrackList.

If the CanEditTracks property is false, this has no effect.

Note: Clients should not assume that the track has been added at the time when this method returns. They should wait for a TrackAdded (or TrackListReplaced) signal.

`CanEditTracks`

Returns:

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

If false, calling AddTrack or RemoveTrack will have no effect, and may raise a NotSupported error.

`GetTracksMetadata`

Parameters:

- **TrackIds - ao (Track_Id_List)** The list of track ids for which metadata is requested.

Returns

- **Metadata - aa{sv} (Metadata_Map_List)** Metadata of the set of tracks given as input.

See the type documentation for more details.

Gets all the metadata available for a set of tracks.

Each set of metadata must have a 'mpris:trackid' entry at the very least, which contains a string that uniquely identifies this track within the scope of the tracklist.

`GoTo`

Parameters:

- **TrackId - o (Track_Id)** Identifier of the track to skip to.

  /org/mpris/MediaPlayer2/TrackList/NoTrack is not a valid value for this argument.

Skip to the specified TrackId.

If the track is not part of this tracklist, this has no effect.

If this object is not /org/mpris/MediaPlayer2, the current TrackList's tracks should be replaced with the contents of this TrackList, and the TrackListReplaced signal should be fired from /org/mpris/MediaPlayer2.

`RemoveTrack`

Parameters:

- **TrackId - o (TrackId)** Identifier of the track to be removed. /org/mpris/MediaPlayer2/TrackList/NoTrack is not a valid value for this argument.

Removes an item from the TrackList.

If the track is not part of this tracklist, this has no effect.

If the CanEditTracks property is false, this has no effect.

Note: Clients should not assume that the track has been removed at the time when this method returns. They should wait for a TrackRemoved (or TrackListReplaced) signal.

TrackAdded

Parameters:

- **Metadata - a{sv} (Metadata_Map)** The metadata of the newly added item.

  This must include a mpris:trackid entry.

  See the type documentation for more details.

- **AfterTrack - o (Track_Id)** The identifier of the track after which the new track was inserted. The path /org/mpris/MediaPlayer2/TrackList/NoTrack indicates that the track was inserted at the start of the track list.

Indicates that a track has been added to the track list.

TrackListReplaced

Parameters:

- **Tracks - ao (Track_Id_List)** The new content of the tracklist.
- **CurrentTrack - o (Track_Id)** The identifier of the track to be considered as current.

  /org/mpris/MediaPlayer2/TrackList/NoTrack indicates that there is no current track.

  This should correspond to the mpris:trackid field of the Metadata property of the org.mpris.MediaPlayer2.Player interface.

Indicates that the entire tracklist has been replaced.

It is left up to the implementation to decide when a change to the track list is invasive enough that this signal should be emitted instead of a series of TrackAdded and TrackRemoved signals.

TrackMetadataChanged

Parameters:

- **TrackId - o (Track_Id)** The id of the track which metadata has changed.

  If the track id has changed, this will be the old value.

  /org/mpris/MediaPlayer2/TrackList/NoTrack is not a valid value for this argument.

- **Metadata - a{sv} (Metadata_Map)** The new track metadata.

  This must include a mpris:trackid entry.

  See the type documentation for more details.

Indicates that the metadata of a track in the tracklist has changed.

This may indicate that a track has been replaced, in which case the mpris:trackid metadata entry is different from the TrackId argument.

TrackRemoved

Parameters:

- **TrackId - o (Track_Id)** The identifier of the track being removed.

  /org/mpris/MediaPlayer2/TrackList/NoTrack is not a valid value for this argument.

Indicates that a track has been removed from the track list.

Tracks

Returns:

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted, but the new value is not sent.

An array which contains the identifier of each track in the tracklist, in order.

The org.freedesktop.DBus.Properties.PropertiesChanged signal is emited every time this property changes, but the signal message does not contain the new value. Client implementations should rather rely on the TrackAdded, TrackRemoved and TrackListReplaced signals to keep their representation of the tracklist up to date.

11.2 MPRIS2 Types

class mpris2.types.**Loop_Status**(*status*)
A repeat / loop status

- **None (None)** The playback will stop when there are no more tracks to play
- **Track (Track)** The current track will start again from the begining once it has finished playing
- **Playlist (Playlist)** The playback loops through a list of tracks

class mpris2.types.**Metadata_Map**(*metadata*)
A mapping from metadata attribute names to values.

The mpris:trackid attribute must always be present. This contains a string that uniquely identifies the track within the scope of the playlist.

If the length of the track is known, it should be provided in the metadata property with the 'mpris:length' key. The length must be given in microseconds, and be represented as a signed 64-bit integer.

If there is an image associated with the track, a URL for it may be provided using the 'mpris:artUrl' key. For other metadata, fields defined by the Xesam ontology should be used, prefixed by 'xesam:'. See http://wiki.xmms2.xmms.se/wiki/MPRIS_Metadata for a list of common fields.

Lists of strings should be passed using the array-of-string ('as') D-Bus type. Dates should be passed as strings using the ISO 8601 extended format (eg: 2007-04-29T14:35:51). If the timezone is known, RFC 3339's internet profile should be used (eg: 2007-04-29T14:35:51+02:00).

- **Attribute - s** The name of the attribute; see http://wiki.xmms2.xmms.se/wiki/MPRIS_Metadata for guidelines on names to use.
- **Value - v** The value of the attribute, in the most appropriate format.

class mpris2.types.**Playback_Rate**(*rate=1.0*)
A playback rate

This is a multiplier, so a value of 0.5 indicates that playback is happening at half speed, while 1.5 means that 1.5 seconds of 'track time' is consumed every second.

class mpris2.types.**Playback_Status**(*status*)
A playback state.

- **Playing (Playing)** A track is currently playing.
- **Paused (Paused)** A track is currently paused.
- **Stopped (Stopped)** There is no track currently playing.

class mpris2.types.**Playlist**(*playlist*)
A data structure describing a playlist.

- **Id - o (Playlist_Id)** A unique identifier for the playlist.

  This should remain the same if the playlist is renamed.

- **Name - s** The name of the playlist, typically given by the user.

- **Icon - s (Uri)** The URI of an (optional) icon.

class mpris2.types.**Maybe_Playlist**(*maybe_playlist=None*)

- **Valid - b** Whether this structure refers to a valid playlist.
- **Playlist - (oss) (Playlist)** The playlist, providing Valid is true, otherwise undefined.

When constructing this type, it should be noted that the playlist ID must be a valid object path, or D-Bus implementations may reject it. This is true even when Valid is false. It is suggested that '/' is used as the playlist ID in this case.

class mpris2.types.**Playlist_Id**(*playlist_id*)
Unique playlist identifier.

class mpris2.types.**Playlist_Ordering**(*ordering*)
Specifies the ordering of returned playlists.

- **Alphabetical (Alphabetical)** Alphabetical ordering by name, ascending.
- **CreationDate (Created)** Ordering by creation date, oldest first.
- **ModifiedDate (Modified)** Ordering by last modified date, oldest first.
- **LastPlayDate (Played)** Ordering by date of last playback, oldest first.
- **UserDefined (User)** A user-defined ordering.

class mpris2.types.**Time_In_Us**(*time=0*)
Time in microseconds.

class mpris2.types.**Uri**(*uri*)
A unique resource identifier.

class mpris2.types.**Volume**(*volume=1.0*)
Audio volume level

- 0.0 means mute.
- 1.0 is a sensible maximum volume level (ex: 0dB).

Note that the volume may be higher than 1.0, although generally clients should not attempt to set it above 1.0.

11.3 mpris2

11.3.1 mpris2 package

Subpackages

mpris2.decorator package

Submodules

mpris2.decorator.attribute module This is not part of specification

Helper class to make it work as python lib

class mpris2.decorator.attribute.**DbusAttr**(*meth=None*, *produces=<function <lambda>>*)
Bases: *mpris2.decorator.base.Decorator*

https://docs.python.org/2/howto/descriptor.html#properties

mpris2.decorator.base module This is not part of specification

Helper class to make it work as python lib

class mpris2.decorator.base.**Decorator**
Bases: object

mpris2.decorator.interface module This is not part of specification

Helper class to make it work as python lib

class mpris2.decorator.interface.**DbusInterface**(*iface=None*, *path=None*, *uri=None*, *dbus_object=None*, *session=None*)
Bases: *mpris2.decorator.base.Decorator*

dbusWrapedInterface(*info_property*, **args*, ***kw*)
Called when some decoreted class was called Inject attrs from decorator at new object then return object

@param *args: list of args to call constructor @param **kw: dict of keywords, can redefine class default parameters @return: instance of decoreted class, with new attributes @see: mpris2.mediaplayer2 to see some examples

mpris2.decorator.method module This is not part of specification

Helper class to make it work as python lib

class mpris2.decorator.method.**DbusMethod**(*meth=None*, *iface=None*, *produces=<function <lambda>>*, *args_to_dbus=<function args_to_dbus>*, *kw_to_dbus=<function kw_to_dbus>*, *std_args=()*, *std_kwds={}*)
Bases: *mpris2.decorator.base.Decorator*

convert_args_to_dbus_args(**args*)

convert_kw_to_dbus_kw(***kw*)

classmethod merge_args(*args*, *std_args*)

classmethod merge_kwds(*kwds*, *std_kwds*)

mpris2.decorator.method.**args_to_dbus**(**args*)

mpris2.decorator.method.**kw_to_dbus**(***kw*)

mpris2.decorator.signal module
class mpris2.decorator.signal.**DbusSignal**(*meth=None*, *iface=None*)
Bases: *mpris2.decorator.base.Decorator*

https://docs.python.org/2/howto/descriptor.html#properties

Module contents This is not part of specification

Helper class to make it work as python lib

mpris2.types package

Submodules

mpris2.types.loop_status module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/Player_Interface.html#Enum:Loop_Status

class mpris2.types.loop_status.**Loop_Status**(*status*)
Bases: str

A repeat / loop status

- **None (None)** The playback will stop when there are no more tracks to play
- **Track (Track)** The current track will start again from the begining once it has finished playing
- **Playlist (Playlist)** The playback loops through a list of tracks

NONE = 'None'

PLAYLIST = 'Playlist'

TRACK = 'Track'

VALUES = ('None', 'Track', 'Playlist')

mpris2.types.metadata_map module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/Track_List_Interface.html#Mapping:Metadata_Map

class mpris2.types.metadata_map.**Metadata_Map**(*metadata*)
Bases: dict

A mapping from metadata attribute names to values.

The mpris:trackid attribute must always be present. This contains a string that uniquely identifies the track within the scope of the playlist.

If the length of the track is known, it should be provided in the metadata property with the 'mpris:length' key. The length must be given in microseconds, and be represented as a signed 64-bit integer.

If there is an image associated with the track, a URL for it may be provided using the 'mpris:artUrl' key. For other metadata, fields defined by the Xesam ontology should be used, prefixed by 'xesam:'. See http://wiki.xmms2.xmms.se/wiki/MPRIS_Metadata for a list of common fields.

Lists of strings should be passed using the array-of-string ('as') D-Bus type. Dates should be passed as strings using the ISO 8601 extended format (eg: 2007-04-29T14:35:51). If the timezone is known, RFC 3339's internet profile should be used (eg: 2007-04-29T14:35:51+02:00).

- **Attribute - s** The name of the attribute; see http://wiki.xmms2.xmms.se/wiki/MPRIS_Metadata for guidelines on names to use.
- **Value - v** The value of the attribute, in the most appropriate format.

ALBUM = 'xesam:album'

ALBUM_ARTIST = 'xesam:albumArtist'

ARTIST = 'xesam:artist'

ART_URI = 'mpris:artUrl'

AS_TEXT = 'xesam:asText'

AUDIO_BPM = 'xesam:audioBPM'

AUTO_RATING = 'xesam:autoRating'

COMMENT = 'xesam:comment'

COMPOSER = 'xesam:composer'

CONTENT_CREATED = 'xesam:contentCreated'

DISC_NUMBER = 'xesam:discNumber'

FIRST_USED = 'xesam:firstUsed'

GENRE = 'xesam:genre'

LAST_USED = 'xesam:lastUsed'

LENGTH = 'mpris:length'

LYRICIST = 'xesam:lyricist'

TITLE = 'xesam:title'

TRACKID = 'mpris:trackid'

TRACK_NUMBER = 'xesam:trackNumber'

URL = 'xesam:url'

USER_RATING = 'xesam:userRating'

USE_COUNT = 'xesam:useCount'

mpris2.types.playback_rate module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/Player_Interface.html#Simple-Type:Playback_Rate

class mpris2.types.playback_rate.**Playback_Rate**(*rate=1.0*)
Bases: float

A playback rate

This is a multiplier, so a value of 0.5 indicates that playback is happening at half speed, while 1.5 means that 1.5 seconds of 'track time' is consumed every second.

mpris2.types.playback_status module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/Player_Interface.html#Enum:Playback_Status

class mpris2.types.playback_status.**Playback_Status**(*status*)
Bases: str

A playback state.

- **Playing (Playing)** A track is currently playing.
- **Paused (Paused)** A track is currently paused.
- **Stopped (Stopped)** There is no track currently playing.

PAUSED = 'Paused'

PLAYING = 'Playing'

STOPPED = 'Stopped'

VALUES = ('Playing', 'Paused', 'Stopped')

mpris2.types.playlist module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/2.2/Playlists_Interface.html#Struct:Playlist

class mpris2.types.playlist.**Maybe_Playlist**(*maybe_playlist=None*)
Bases: dbus.Struct

- **Valid - b** Whether this structure refers to a valid playlist.
- **Playlist - (oss) (Playlist)** The playlist, providing Valid is true, otherwise undefined.

When constructing this type, it should be noted that the playlist ID must be a valid object path, or D-Bus implementations may reject it. This is true even when Valid is false. It is suggested that '/' is used as the playlist ID in this case.

Playlist

Valid

class mpris2.types.playlist.**Playlist**(*playlist*)
Bases: dbus.Struct

A data structure describing a playlist.

- **Id - o (Playlist_Id)** A unique identifier for the playlist.

  This should remain the same if the playlist is renamed.
- **Name - s** The name of the playlist, typically given by the user.
- **Icon - s (Uri)** The URI of an (optional) icon.

Icon

Id

Name

mpris2.types.playlist_id module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/2.2/Playlists_Interface.html#Simple-Type:Playlist_Id

class mpris2.types.playlist_id.**Playlist_Id**(*playlist_id*)
Bases: str

Unique playlist identifier.

mpris2.types.playlist_ordering module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/2.2/Playlists_Interface.html#Enum:Playlist_Ordering

class mpris2.types.playlist_ordering.**Playlist_Ordering**(*ordering*)
Bases: str

Specifies the ordering of returned playlists.

- **Alphabetical (Alphabetical)** Alphabetical ordering by name, ascending.
- **CreationDate (Created)** Ordering by creation date, oldest first.
- **ModifiedDate (Modified)** Ordering by last modified date, oldest first.
- **LastPlayDate (Played)** Ordering by date of last playback, oldest first.
- **UserDefined (User)** A user-defined ordering.

ALPHABETICAL = 'Alphabetical'

CREATION_DATE = 'CreationDate'

LAST_PLAY_DATE = 'LastPlayDate'

MODIFIED_DATE = 'ModifiedDate'

USER_DEFINE = 'UserDefined'

VALUES = ('Alphabetical', 'CreationDate', 'ModifiedDate', 'LastPlayDate', 'UserDefined')

mpris2.types.time_in_us module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/Player_Interface.html#Simple-Type:Time_In_Us

class mpris2.types.time_in_us.**Time_In_Us**(*time=0*)
Bases: int

Time in microseconds.

mpris2.types.uri module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/2.2/Playlists_Interface.html#Simple-Type:Uri

class mpris2.types.uri.**Uri**(*uri*)
Bases: str

A unique resource identifier.

mpris2.types.volume module From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/Player_Interface.html#Simple-Type:Volume

class mpris2.types.volume.**Volume**(*volume=1.0*)
Bases: float

Audio volume level

- 0.0 means mute.
- 1.0 is a sensible maximum volume level (ex: 0dB).

Note that the volume may be higher than 1.0, although generally clients should not attempt to set it above 1.0.

MAX = 1.0

MIN = 0.0

RANGE = set([0.0, 0.5, 0.2, 0.4, 1.0, 0.8, 0.6, 0.3, 0.1, 0.9, 0.7])

n = 10

Module contents

Submodules

mpris2.interfaces module

From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/2.2/#Interfaces

class mpris2.interfaces.**Interfaces**
Bases: object

This class contains the constants defined at index of MPRIS2 definition:

Interfaces:

- **MEDIA_PLAYER** 'org.mpris.MediaPlayer2'
- **TRACK_LIST** 'org.mpris.MediaPlayer2.TrackList'
- **PLAYER** 'org.mpris.MediaPlayer2.Player'
- **PLAYLISTS** 'org.mpris.MediaPlayer2.Playlists'
- **PROPERTIES** 'org.freedesktop.DBus.Properties'

Signals:

- **SIGNAL** 'PropertiesChanged'

Objects:

- **OBJECT_PATH** '/org/mpris/MediaPlayer2'

MEDIA_PLAYER = 'org.mpris.MediaPlayer2'

OBJECT_PATH = '/org/mpris/MediaPlayer2'

PLAYER = 'org.mpris.MediaPlayer2.Player'

PLAYLISTS = 'org.mpris.MediaPlayer2.Playlists'

PROPERTIES = 'org.freedesktop.DBus.Properties'

SIGNAL = 'PropertiesChanged'

TRACK_LIST = 'org.mpris.MediaPlayer2.TrackList'

mpris2.mediaplayer2 module

From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/Media_Player.html

class mpris2.mediaplayer2.**MediaPlayer2**(**args*, ***kw*)
Bases: *mpris2.interfaces.Interfaces*

Interface for MediaPlayer2 (org.mpris.MediaPlayer2)

CanQuit
Returns

Read only Inject attrs from decorator at new object then return obje

When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

If false, calling Quit will have no effect, and may raise a NotSupported error. If true, calling Quit will cause the media application to attempt to quit (although it may still be prevented from quitting by the user, for example).

CanRaise
Returns

Read only If false, calling Raise will have no effect, and may raise a NotSupported error. If true, calling Raise will cause the media application to attempt to bring its user interface to the front, although it may be prevented from doing so (by the window manager, for example).

When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

CanSetFullscreen
Returns

Read only If false, attempting to set Fullscreen will have no effect, and may raise an error. If true, attempting to set Fullscreen will not raise an error, and (if it is different from the current value) will cause the media player to attempt to enter or exit fullscreen mode.

This property is optional. Clients should handle its absence gracefully.

When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Added in 2.2.

..note:: Note that the media player may be unable to fulfil the request. In this case, the value will not change. If the media player knows in advance that it will not be able to fulfil the request, however, this property should be false.

DesktopEntry
Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The basename of an installed .desktop file which complies with the Desktop entry specification, with the '.desktop' extension stripped.

Example: The desktop entry file is '/usr/share/applications/vlc.desktop', and this property contains 'vlc'

This property is optional. Clients should handle its absence gracefully

Fullscreen
Returns

Read Write Whether the media player is occupying the fullscreen.

This property is optional. Clients should handle its absence gracefully.

When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

This is typically used for videos. A value of true indicates that the media player is taking up the full screen.

Media center software may well have this value fixed to true

If CanSetFullscreen is true, clients may set this property to true to tell the media player to enter fullscreen mode, or to false to return to windowed mode.

If CanSetFullscreen is false, then attempting to set this property should have no effect, and may raise an error. However, even if it is true, the media player may still be unable to fulfil the request, in which case attempting to set this property will have no effect (but should not raise an error).

Added in 2.2.

HasTrackList
Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Indicates whether the /org/mpris/MediaPlayer2 object implements the org.mpris.MediaPlayer2.TrackList interface.

`Identity`
Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

If false, calling Raise will have no effect, and may raise a NotSupported error. If true, calling Raise will cause the media application to attempt to bring its user interface to the front, although it may be prevented from doing so (by the window manager, for example).

`PROPERTIES_CAN_QUIT` = 'CanQuit'

`PROPERTIES_CAN_RAISE` = 'Identity'

`PROPERTIES_DESKTOP_ENTRY` = 'DesktopEntry'

`PROPERTIES_HAS_TRACK_LIST` = 'HasTrackList'

`PROPERTIES_IDENTITY` = 'Identity'

`PROPERTIES_SUPPORTED_MINE_TYPES` = 'SupportedMimeTypes'

`PROPERTIES_SUPPORTED_URI_SCHEMES` = 'SupportedUriSchemes'

`PropertiesChanged`
Parameters:

- **args - list** unnamed parameters passed by dbus signal
- **kw - dict** named parameters passed by dbus signal

Every time that some property change, signal will be called

`Quit`
Causes the media player to stop running.

The media player may refuse to allow clients to shut it down. In this case, the CanQuit property is false and this method does nothing.

..note:: Media players which can be D-Bus activated, or for which there is no sensibly easy way to terminate a running instance (via the main interface or a notification area icon for example) should allow clients to use this method. Otherwise, it should not be needed.

If the media player does not have a UI, this should be implemented

`Raise`
Brings the media player's user interface to the front using any appropriate mechanism available.

The media player may be unable to control how its user interface is displayed, or it may not have a graphical user interface at all. In this case, the Identity property is false and this method does nothing.

`SIGNALS_PROPERTIES_CHANGED` = 'PropertiesChanged'

`SupportedMimeTypes`
Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The mime-types supported by the media player.

Mime-types should be in the standard format (eg: audio/mpeg or application/ogg).

Note: This is important for clients to know when using the editing capabilities of the Playlist interface, for example.

SupportedUriSchemes

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The URI schemes supported by the media player.

This can be viewed as protocols supported by the player in almost all cases. Almost every media player will include support for the 'file' scheme. Other common schemes are 'http' and 'rtsp'.

Note that URI schemes should be lower-case.

Note: This is important for clients to know when using the editing capabilities of the Playlist interface, for example.

---

mpris2.player module

From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/Player_Interface.html

class mpris2.player.**Player**(**args*, ***kw*)

Bases: *mpris2.interfaces.Interfaces*

This interface implements the methods for querying and providing basic control over what is currently playing.

CanControl

Returns

Read only The org.freedesktop.DBus.Properties.PropertiesChanged signal is not emitted when this property changes.

Whether the media player may be controlled over this interface.

This property is not expected to change, as it describes an intrinsic capability of the implementation.

If this is false, clients should assume that all properties on this interface are read-only (and will raise errors if writing to them is attempted); all methods are not implemented and all other properties starting with 'Can' are also false.

CanGoNext

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether the client can call the Next method on this interface and expect the current track to change.

If CanControl is false, this property should also be false.

CanGoPrevious

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether the client can call the Previous method on this interface and expect the current track to change.

If CanControl is false, this property should also be false.

CanPause

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether playback can be paused using Pause or PlayPause.

Note that this is an intrinsic property of the current track: its value should not depend on whether the track is currently paused or playing. In fact, if playback is currently paused (and CanControl is true), this should be true.

If CanControl is false, this property should also be false.

CanPlay

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether playback can be started using Play or PlayPause.

Note that this is related to whether there is a 'current track': the value should not depend on whether the track is currently paused or playing. In fact, if a track is currently playing CanControl is true), this should be true.

If CanControl is false, this property should also be false.

CanSeek

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

Whether the client can control the playback position using Seek and SetPosition. This may be different for different tracks.

If CanControl is false, this property should also be false.

LoopStatus

Returns

Read/Write When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The current loop / repeat status

May be:

- 'None' if the playback will stop when there are no more tracks to play
- 'Track' if the current track will start again from the begining once it has finished playing
- 'Playlist' if the playback loops through a list of tracks

This property is optional, and clients should deal with NotSupported errors gracefully.

If CanControl is false, attempting to set this property should have no effect and raise an error.

MaximumRate

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The maximum value which the Rate property can take. Clients should not attempt to set the Rate property above this value.

This value should always be 1.0 or greater.

Metadata

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The metadata of the current element.

If there is a current track, this must have a 'mpris:trackid' entry at the very least, which contains a string that uniquely identifies this track.

See the type documentation for more details.

MinimumRate

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The minimum value which the Rate property can take. Clients should not attempt to set the Rate property below this value.

Note that even if this value is 0.0 or negative, clients should not attempt to set the Rate property to 0.0.

This value should always be 1.0 or less.

Next

Skips to the next track in the tracklist.

If there is no next track (and endless playback and track repeat are both off), stop playback.

If playback is paused or stopped, it remains that way.

If CanGoNext is false, attempting to call this method should have no effect.

OpenUri

Parameters:

- **Uri - s (Uri)** Uri of the track to load. Its uri scheme should be an element of the org.mpris.MediaPlayer2.SupportedUriSchemes property and the mime-type should match one of the elements of the org.mpris.MediaPlayer2.SupportedMimeTypes.

Opens the Uri given as an argument

If the playback is stopped, starts playing

If the uri scheme or the mime-type of the uri to open is not supported, this method does nothing and may raise an error. In particular, if the list of available uri schemes is empty, this method may not be implemented.

Clients should not assume that the Uri has been opened as soon as this method returns. They should wait until the mpris:trackid field in the Metadata property changes.

If the media player implements the TrackList interface, then the opened track should be made part of the tracklist, the org.mpris.MediaPlayer2.TrackList.TrackAdded or org.mpris.MediaPlayer2.TrackList.TrackListReplaced signal should be fired, as well as the org.freedesktop.DBus.Properties.PropertiesChanged signal on the tracklist interface.

Pause

Pauses playback.

If playback is already paused, this has no effect.

Calling Play after this should cause playback to start again from the same position.

If CanPause is false, attempting to call this method should have no effect.

Play

Starts or resumes playback.

If already playing, this has no effect.

If there is no track to play, this has no effect.

If CanPlay is false, attempting to call this method should have no effect.

PlayPause

Pauses playback.

If playback is already paused, resumes playback.

If playback is stopped, starts playback.

If CanPause is false, attempting to call this method should have no effect and raise an error.

PlaybackStatus

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The current playback status.

May be 'Playing', 'Paused' or 'Stopped'.

Position

Returns

Read only The org.freedesktop.DBus.Properties.PropertiesChanged signal is not emitted when this property changes.

The current track position in microseconds, between 0 and the 'mpris:length' metadata entry (see Metadata).

Note: If the media player allows it, the current playback position can be changed either the SetPosition method or the Seek method on this interface. If this is not the case, the CanSeek property is false, and setting this property has no effect and can raise an error.

If the playback progresses in a way that is inconstistant with the Rate property, the Seeked signal is emited.

Previous

Skips to the previous track in the tracklist.

If there is no previous track (and endless playback and track repeat are both off), stop playback.

If playback is paused or stopped, it remains that way.

If CanGoPrevious is false, attempting to call this method should have no effect.

PropertiesChanged

Parameters

- **args - list** unnamed parameters passed by dbus signal
- **kw - dict** named parameters passed by dbus signal

Every time that some property change, signal will be called

Rate

Returns

Read/Write When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The current playback rate.

The value must fall in the range described by MinimumRate and MaximumRate, and must not be 0.0. If playback is paused, the PlaybackStatus property should be used to indicate this. A value of 0.0 should not be set by the client. If it is, the media player should act as though Pause was called.

If the media player has no ability to play at speeds other than the normal playback rate, this must still be implemented, and must return 1.0. The MinimumRate and MaximumRate properties must also be set to 1.0.

Not all values may be accepted by the media player. It is left to media player implementations to decide how to deal with values they cannot use; they may either ignore them or pick a 'best fit' value. Clients are recommended to only use sensible fractions or multiples of 1 (eg: 0.5, 0.25, 1.5, 2.0, etc).

Seek

Parameters:

- **Offset - x (Time_In_Us)** The number of microseconds to seek forward.

Seeks forward in the current track by the specified number of microseconds.

A negative value seeks back. If this would mean seeking back further than the start of the track, the position is set to 0.

If the value passed in would mean seeking beyond the end of the track, acts like a call to Next.

If the CanSeek property is false, this has no effect.

Seeked

Parameters:

- **Position - x (Time_In_Us)** The new position, in microseconds.

Indicates that the track position has changed in a way that is inconsistant with the current playing state.

When this signal is not received, clients should assume that:

- When playing, the position progresses according to the rate property.
- When paused, it remains constant.

This signal does not need to be emitted when playback starts or when the track changes, unless the track is starting at an unexpected position. An expected position would be the last known one when going from Paused to Playing, and 0 when going from Stopped to Playing.

SetPosition

Parameters

- **TrackId - o (Track_Id)** The currently playing track's identifier.

  If this does not match the id of the currently-playing track, the call is ignored as 'stale'.

- **Position - x (Time_In_Us)** Track position in microseconds.

  This must be between 0 and <track_length>.

Sets the current track position in microseconds.

If the Position argument is less than 0, do nothing.

If the Position argument is greater than the track length, do nothing.

If the CanSeek property is false, this has no effect.

Shuffle

Returns

Read/Write When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

A value of false indicates that playback is progressing linearly through a playlist, while true means playback is progressing through a playlist in some other order.

This property is optional, and clients should deal with NotSupported errors gracefully.

If CanControl is false, attempting to set this property should have no effect and raise an error.

Stop

Stops playback.

If playback is already stopped, this has no effect.

Calling Play after this should cause playback to start again from the beginning of the track.

If CanControl is false, attempting to call this method should have no effect and raise an error.

Volume

Returns

Read/Write When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The volume level.

When setting, if a negative value is passed, the volume should be set to 0.0.

If CanControl is false, attempting to set this property should have no effect and raise an error.

mpris2.playlists module

From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/Playlists_Interface.html

class mpris2.playlists.**Playlists**(**args*, ***kw*)

Bases: *mpris2.interfaces.Interfaces*

Provides access to the media player's playlists.

Since D-Bus does not provide an easy way to check for what interfaces are exported on an object, clients should attempt to get one of the properties on this interface to see if it is implemented.

ActivatePlaylist

Parameters:

- **PlaylistId - o** The id of the playlist to activate.

Starts playing the given playlist.

Note that this must be implemented. If the media player does not allow clients to change the playlist, it should not implement this interface at all.

It is up to the media player whether this completely replaces the current tracklist, or whether it is merely inserted into the tracklist and the first track starts. For example, if the media player is operating in a 'jukebox' mode, it may just append the playlist to the list of upcoming tracks, and skip to the first track in the playlist.

`ActivePlaylist`

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The currently-active playlist.

If there is no currently-active playlist, the structure's Valid field will be false, and the Playlist details are undefined.

Note that this may not have a value even after ActivatePlaylist is called with a valid playlist id as ActivatePlaylist implementations have the option of simply inserting the contents of the playlist into the current tracklist.

`GetPlaylists`

Parameters:

- **Index - u** The index of the first playlist to be fetched (according to the ordering).
- **MaxCount - u** The maximum number of playlists to fetch.
- **Order - s (Playlist_Ordering)** The ordering that should be used.
- **ReverseOrder - b** Whether the order should be reversed.

Returns

- **Playlists - a(oss) (Playlist_List)** A list of (at most MaxCount) playlists.

Gets a set of playlists.

`Orderings`

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The avaislable orderings. At least one must be offered.

`PlaylistChanged`

Parameters

- **Playlist - (oss) (Playlist)** The playlist whose details have changed.

Indicates that the name or icon for a playlist has changed.

Note that, for this signal to operate correctly, the id of the playlist must not change when the name changes.

`PlaylistCount`

Returns

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

The number of playlists available.

mpris2.some_players module

class mpris2.some_players.**Some_Players**
Bases: object

Not defined in documentation

Maybe this player (and other) implement mpris2

Some players

- **AUDACIOUS** 'audacious'
- **BANSHEE** 'banshee'
- **BEATBOX** 'beatbox'
- **BMP** 'bmp'
- **CLEMENTINE** 'clementine'
- **DRAGONPLAYER** 'dragonplayer'
- **EXAILE** 'exaile'
- **GMUSICBROWSER** 'gmusicbrowser'
- **GMPC** 'gmpc'
- **GUAYADEQUE** 'guayadeque'
- **MOPIDY** 'mopidy'
- **MPDRIS** 'mpDris'
- **QUODLIBET** 'quodlibet'
- **RAVEND** 'ravend'
- **RHYTHMBOX** 'rhythmbox'
- **SPOTIFY** 'spotify'
- **VLC** 'vlc'
- **XBMC** 'xbmc'
- **XMMS2** 'xmms2'
- **XNOISE** 'xnoise'

AUDACIOUS = 'audacious'

BANSHEE = 'banshee'

BEATBOX = 'beatbox'

BMP = 'bmp'

CLEMENTINE = 'clementine'

DRAGONPLAYER = 'dragonplayer'

EXAILE = 'exaile'

GMPC = 'gmpc'

GMUSICBROWSER = 'gmusicbrowser'

GUAYADEQUE = 'guayadeque'

MOPIDY = 'mopidy'

MPDRIS = 'mpDris'

QUODLIBET = 'quodlibet'

RAVEND = 'ravend'

RHYTHMBOX = 'rhythmbox'

SPOTIFY = 'spotify'

VLC = 'vlc'

XBMC = 'xbmc'

XMMS2 = 'xmms2'

XNOISE = 'xnoise'

static **get_dict**()

mpris2.tracklist module

From mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/2.2/Track_List_Interface.html

class mpris2.tracklist.**TrackList**(**args*, ***kw*)

Bases: *mpris2.interfaces.Interfaces*

Interface for TrackList (org.mpris.MediaPlayer2.TrackList)

Provides access to a short list of tracks which were recently played or will be played shortly. This is intended to provide context to the currently-playing track, rather than giving complete access to the media player's playlist.

Example use cases are the list of tracks from the same album as the currently playing song or the Rhythmbox play queue.

Each track in the tracklist has a unique identifier. The intention is that this uniquely identifies the track within the scope of the tracklist. In particular, if a media item (a particular music file, say) occurs twice in the track list, each occurrence should have a different identifier. If a track is removed from the middle of the playlist, it should not affect the track ids of any other tracks in the tracklist.

As a result, the traditional track identifiers of URLs and position in the playlist cannot be used. Any scheme which satisfies the uniqueness requirements is valid, as clients should not make any assumptions about the value of the track id beyond the fact that it is a unique identifier.

Note that the (memory and processing) burden of implementing the TrackList interface and maintaining unique track ids for the playlist can be mitigated by only exposing a subset of the playlist when it is very long (the 20 or so tracks around the currently playing track, for example). This is a recommended practice as the tracklist interface is not designed to enable browsing through a large list of tracks, but rather to provide clients with context about the currently playing track.

AddTrack

Parameters:

- **Uri - s (Uri)** The uri of the item to add. Its uri scheme should be an element of the org.mpris.MediaPlayer2.SupportedUriSchemes property and the mime-type should match one of the elements of the org.mpris.MediaPlayer2.SupportedMimeTypes
- **AfterTrack - o (Track_Id)** The identifier of the track after which the new item should be inserted. The path /org/mpris/MediaPlayer2/TrackList/NoTrack indicates that the track should be inserted at the start of the track list.
- **SetAsCurrent - b** Whether the newly inserted track should be considered as the current track. Setting this to trye has the same effect as calling GoTo afterwards.

Adds a URI in the TrackList.

If the CanEditTracks property is false, this has no effect.

Note: Clients should not assume that the track has been added at the time when this method returns. They should wait for a TrackAdded (or TrackListReplaced) signal.

CanEditTracks

Returns:

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted with the new value.

If false, calling AddTrack or RemoveTrack will have no effect, and may raise a NotSupported error.

GetTracksMetadata

Parameters:

- **TrackIds - ao (Track_Id_List)** The list of track ids for which metadata is requested.

Returns

- **Metadata - aa{sv} (Metadata_Map_List)** Metadata of the set of tracks given as input.

See the type documentation for more details.

Gets all the metadata available for a set of tracks.

Each set of metadata must have a 'mpris:trackid' entry at the very least, which contains a string that uniquely identifies this track within the scope of the tracklist.

GoTo

Parameters:

- **TrackId - o (Track_Id)** Identifier of the track to skip to.

  /org/mpris/MediaPlayer2/TrackList/NoTrack is not a valid value for this argument.

Skip to the specified TrackId.

If the track is not part of this tracklist, this has no effect.

If this object is not /org/mpris/MediaPlayer2, the current TrackList's tracks should be replaced with the contents of this TrackList, and the TrackListReplaced signal should be fired from /org/mpris/MediaPlayer2.

PROPERTIES_CAN_EDIT_TRACKS = 'CanEditTracks'

PROPERTIES_TACKS = 'Tracks'

RemoveTrack

Parameters:

- **TrackId - o (TrackId)** Identifier of the track to be removed. /org/mpris/MediaPlayer2/TrackList/NoTrack is not a valid value for this argument.

Removes an item from the TrackList.

If the track is not part of this tracklist, this has no effect.

If the CanEditTracks property is false, this has no effect.

Note: Clients should not assume that the track has been removed at the time when this method returns. They should wait for a TrackRemoved (or TrackListReplaced) signal.

SIGNALS_PROPERTIES_CHANGED = 'PropertiesChanged'

SIGNALS_TRACK_ADDED = 'TrackAdded'

SIGNALS_TRACK_LIST_REPLACED = 'TrackListReplaced'

SIGNALS_TRACK_METADATA_CHANGED = 'TrackMetadataChanged'

SIGNALS_TRACK_REMOVED = 'TrackRemoved'

TrackAdded

Parameters:

- **Metadata - a{sv} (Metadata_Map)** The metadata of the newly added item.

  This must include a mpris:trackid entry.

  See the type documentation for more details.

- **AfterTrack - o (Track_Id)** The identifier of the track after which the new track was inserted. The path /org/mpris/MediaPlayer2/TrackList/NoTrack indicates that the track was inserted at the start of the track list.

Indicates that a track has been added to the track list.

TrackListReplaced

Parameters:

- **Tracks - ao (Track_Id_List)** The new content of the tracklist.
- **CurrentTrack - o (Track_Id)** The identifier of the track to be considered as current.

  /org/mpris/MediaPlayer2/TrackList/NoTrack indicates that there is no current track.

  This should correspond to the mpris:trackid field of the Metadata property of the org.mpris.MediaPlayer2.Player interface.

Indicates that the entire tracklist has been replaced.

It is left up to the implementation to decide when a change to the track list is invasive enough that this signal should be emitted instead of a series of TrackAdded and TrackRemoved signals.

TrackMetadataChanged

Parameters:

- **TrackId - o (Track_Id)** The id of the track which metadata has changed.

  If the track id has changed, this will be the old value.

  /org/mpris/MediaPlayer2/TrackList/NoTrack is not a valid value for this argument.

- **Metadata - a{sv} (Metadata_Map)** The new track metadata.

  This must include a mpris:trackid entry.

  See the type documentation for more details.

Indicates that the metadata of a track in the tracklist has changed.

This may indicate that a track has been replaced, in which case the mpris:trackid metadata entry is different from the TrackId argument.

TrackRemoved

Parameters:

- **TrackId - o (Track_Id)** The identifier of the track being removed.

  /org/mpris/MediaPlayer2/TrackList/NoTrack is not a valid value for this argument.

Indicates that a track has been removed from the track list.

Tracks

Returns:

Read only When this property changes, the org.freedesktop.DBus.Properties.PropertiesChanged signal is emitted, but the new value is not sent.

An array which contains the identifier of each track in the tracklist, in order.

The org.freedesktop.DBus.Properties.PropertiesChanged signal is emited every time this property changes, but the signal message does not contain the new value. Client implementations should rather rely on the TrackAdded, TrackRemoved and TrackListReplaced signals to keep their representation of the tracklist up to date.

mpris2.utils module

utils functions not defined in espec

mpris2.utils.**get_player_id_from_uri**(*uri*)
Returns player mpris2 id from uri @param uri: string mpris2 player dbus uri @return: string mrpis2 id

mpris2.utils.**get_players_id**(*pattern=None*)
Return string of player mpris2 id @param pattern=None: string RegEx that filter response @return: array string of players bus name

mpris2.utils.**get_players_uri**(*pattern=''*)
Return string of player bus name @param pattern=None: string RegEx that filter response @return: array string of players bus name

mpris2.utils.**implements**(*uri*, *interface*, *path='/org/mpris/MediaPlayer2'*, *bus=None*)

mpris2.utils.**list_interfaces**(*uri*, *path=None*, *bus=None*)

Module contents

This is a copy of mprisV2.2 documentation

http://specifications.freedesktop.org/mpris-spec/latest/

That also works as python lib.

Version 2.2

About

The Media Player Remote Interfacing Specification is a standard D-Bus interface which aims to provide a common programmatic API for controlling media players.

It provides a mechanism for compliant media players discovery, basic playback and media player state control as well as a tracklist interface which is used to add context to the current item.

Changes

Changes (Permalink)

From 2.1 to 2.2:

- Added the optional Fullscreen and CanSetFullscreen properties to the org.mpris.MediaPlayer2 interface.
- The path /org/mpris/MediaPlayer2/TrackList/NoTrack now represents “no track” where required in the org.mpris.MediaPlayer2.TrackList interface (since empty paths are not allowed by D-Bus).
- The suggested unique instance identifier no longer violates the D-Bus specification by begining with a digit.

From 2.0 to 2.1:

- Added the optional org.mpris.MediaPlayer2.Playlists interface.

Bus Name Policy

Each media player *must* request a unique bus name which begins with *org.mpris.MediaPlayer2*. For example:

- org.mpris.MediaPlayer2.audacious
- org.mpris.MediaPlayer2.vlc
- org.mpris.MediaPlayer2.bmp
- org.mpris.MediaPlayer2.xmms2

This allows clients to list available media players (either already running or which can be started via D-Bus activation)

In the case where the media player allows multiple instances running simultaneously, each additional instance should request a unique bus name, adding a dot and a unique identifier to its usual bus name, such as one based on a UNIX process id. For example, this could be:

- org.mpris.MediaPlayer2.vlc.7389

Note: According to the D-Bus specification, the unique identifier "must only contain the ASCII characters '[A-Z][a-z][0-9]_-'" and "must not begin with a digit".

Entry point

The media player *must* expose the */org/mpris/MediaPlayer2* object path, which *must* implement the following interfaces:

- org.mpris.MediaPlayer2
- org.mpris.MediaPlayer2.Player

The */org/mpris/MediaPlayer2* object may implement the org.mpris.MediaPlayer2.TrackList interface.

The */org/mpris/MediaPlayer2* object may implement the org.mpris.MediaPlayer2.Playlists interface.

The PropertiesChanged signal

The MPRIS uses the org.freedesktop.DBus.Properties.PropertiesChanged signal to notify clients of changes in the media player state. If a client implementation uses D-Bus bindings which do not support this signal, then it should connect to it manually. If a media player implementation uses D-Bus bindings which do not support this signal, then it should send it manually

Corrections

2010-09-26: Added EmitsChangedSignal annotation to Volume property on the Player interface.

2011-01-26: Added PlaylistChanged signal to the Playlists interface.

Interfaces

- org.mpris.MediaPlayer2
- org.mpris.MediaPlayer2.Player
- org.mpris.MediaPlayer2.Playlists
- org.mpris.MediaPlayer2.TrackList

Here is some examples, that shows how to work with this lib.

11.4 Configure dbus and mainloop

```
>>> # configure mainloop (not required if you wont expect signals)
>>> from dbus.mainloop.glib import DBusGMainLoop
>>> DBusGMainLoop(set_as_default=True)
```

11.5 Discover your player mpris uri

```
>>> # you can use get_players_uri to get current running players uri
>>> from mpris2 import get_players_uri
>>> # next raise StopIteration if not found
>>> uri = next(get_players_uri())
```

11.6 Connect to player

```
>>> # create you player
>>> from mpris2 import Player
>>> player = Player(dbus_interface_info={'dbus_uri': uri})
```

11.7 Call methods

```
>>> player.Next() # play next media
```

11.8 Get attributes

```
>>> print(player.Metadata) #current media data
```

11.9 Wait signal

```
>>> def another_handler(self, *args, **kw):
>>>     print(args, kw)
>>>
>>> player.PropertiesChanged = another_handler
>>> # python3
>>> import gi.repository.GLib
>>> mloop = gi.repository.GLib.MainLoop()
>>> mloop.run()
```

11.10 Other examples:

```
>>> # old versions mainloop
>>> import gobject
>>> mloop = gobject.MainLoop()
```

```
>>> # list all running players
>>> from mpris2 import get_players_uri
>>> print([uri for uri in get_players_uri()])
>>> # get_players_uri can be called with filter parameter
>>> get_players_uri('.+rhythmbox')
>>> # you can set it yourself
>>> uri = 'org.mpris.MediaPlayer2.gmusicbrowser'
```

```
>>> # or use one predefined
>>> from mpris2 import SomePlayers, Interfaces
>>> uri = '.'.join([Interfaces.MEDIA_PLAYER, SomePlayers.GMUSICBROWSER])
```

```
>>> # test other interfaces
>>> from mpris2 import MediaPlayer2
>>> mp2 = MediaPlayer2(dbus_interface_info={'dbus_uri': uri})
>>> # not all players implement this:
>>> from mpris2 import Playlists, TrackList
>>> pls = Playlists(dbus_interface_info={'dbus_uri': uri})
>>> tl = TrackList(dbus_interface_info={'dbus_uri': uri})
```

m

A

B

C

D

N

O

P

Q

R

S

T

U

V

X