# UMMS introduction

## A Unified Multi Media Service for MeeGoTV and other Linux

*Presenter:*

Geoffroy Van Cutsem, Technical Marketing Engineer

*But credits go to the UMMS team:*

Dominique Le Foll, Senior Architect for TV and IVI

SmartTV middleware team: Rui Long, Zhiwen Wu

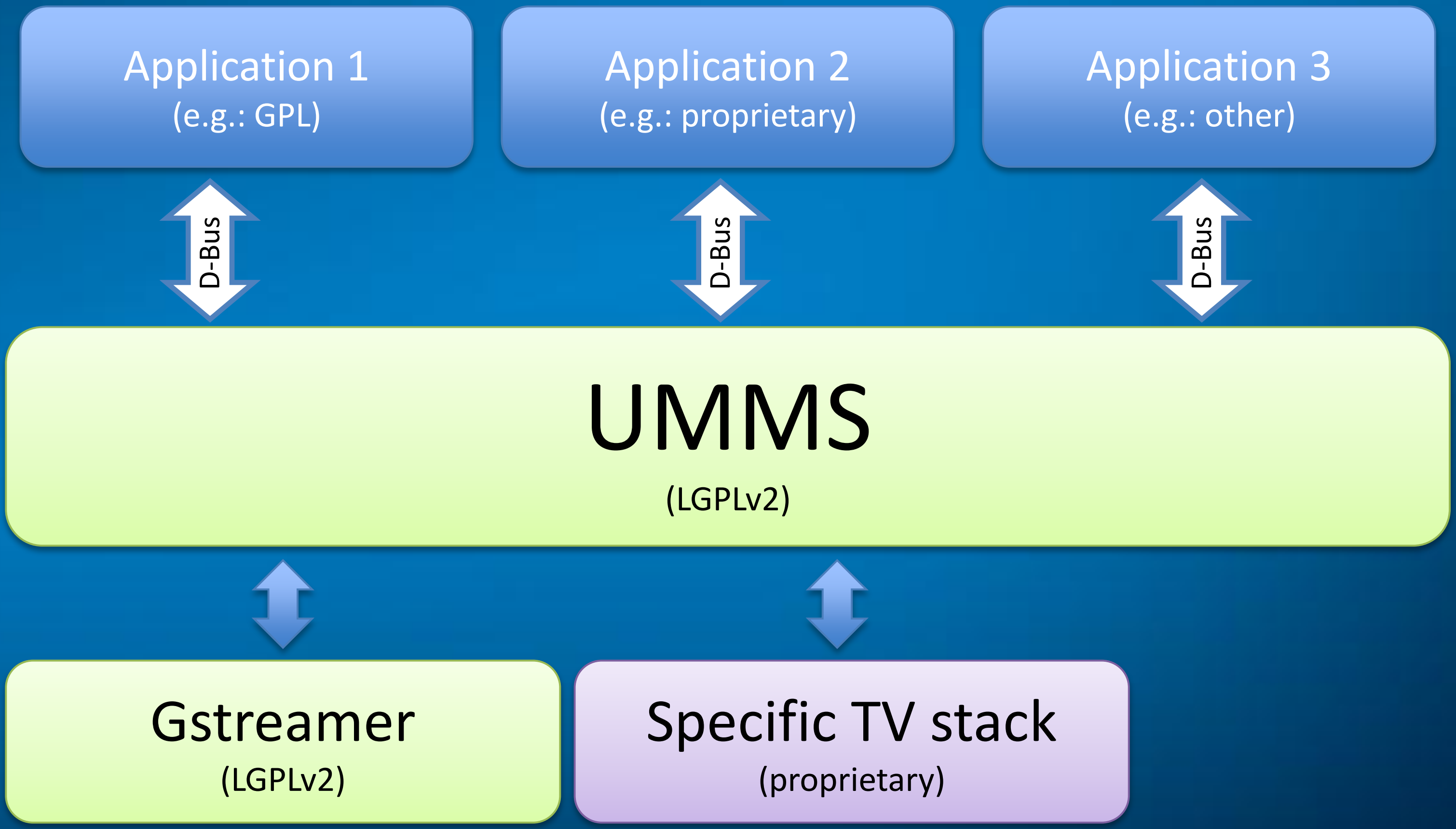
## Agenda

- What is UMMS?
- Typical use cases
- API introduction
- A First Implementation
- Conclusion

## What is UMMS?

- Unified Multi Media Service
- UMMS offers a service to enable a large community of developers to benefit from the best possible Audio and Video capabilities provided by various Linux implementations without having to worry about the underlying details.
- Comprehensive, yet easy-to-use API
- A set of D-Bus APIs for multimedia application developer
- A framework for back-end engine developers

# Architecture of UMMS



## What is the rationale for UMMS?

*In defining UMMS, the key objectives are to provide:*

- Hardware platform independence
  - Can be used indifferently in all segments: TV, Netbook, IVI, Tablet, ...
- Programming language independence
  - Various D-Bus bindings available (Python, C++, Java, glib....)
- Support for TV-specific features
  - DVB, PVR, EPG, Time-shifting, CA, DRM
- License isolation
- Support new HW features
  - E.g. video as an OpenGL texture

## What is the rationale for UMMS?

- Flexibility to have various backends
  Gstreamer, ffmpeg, platform-specific player
- HW resource management
  UMMS acts as a daemon abstracting the limited media processing resources (e.g. HW decoder)

## Agenda

- What is UMMS?
- Typical use cases
- API  introduction
- A First Implementation
- Conclusion

## Typical Use Case 1: simple media player

The application creates an attended request to UMMS.

- Set the URI
- Start playing

- During playback, query the play status (e.g. elapsed time, time till end…) and update the UI.

- If the content is reported as ‘seekable’, the application can also use a cursor to navigate through the video.

## Typical User Case 2: PVR

Personal Video Recorder

- When the recording needs to start, the application triggers an unattended request to the UMMS giving it the time it needs to execute.
- The application gives a Live TV source (URI) and a local file target to start the recording.

## Typical Use Case 3: browser integration

Integrate with Browser for HTML5 video tag or javaScript Video object

- Browser creates an attended request to the UMMS.
- Set URI (e.g. http://xxx.ogg)
- Set target of UMMS, either as a physical or a UI element. When the user scrolls up and down the page, the browser simply provides the updated position to the UMMS to allow the video to repositioned correctly.

## Agenda

- What is UMMS?
- Typical use cases
- API introduction
- A First Implementation
- Conclusion

## Attended vs. Unattended

- There are types of MediaPlayer object that can be requested:
  - Attended and Unattended

1. *Attended*: the application remains active during the execution. UMMS will monitor that the application is still alive.
   - A small client library that wraps this interaction is provided for convenience.
2. *Unattended:* the application does not need to remain active during the AV execution

# API Introduction

- The API definition is still work in progress
- For the most up-to-date definition, check the spec/ folder in the source code

UMMSObjectManager Methods:

| Function Name | Parameter Name | Parameter Type | Direction |
|---|---|---|---|
| RequestMediaPlayer | object_path | string | output |
| RequestMediaPlayerUnattended | time_to_execute | double | input |
| | object_path | string | output |
| RemoveMediaPlayer | object_path | string | input |

# API Introduction

UMMSMediaPlayer Methods:

| Function Name | Parameter Name | Parameter Type | Direction |
|---|---|---|---|
| SetUri | uri | string | input |
| SetTarget | type | int | input |
|  | param | a{sv} | input |
| Play |  |  |  |
| Pause |  |  |  |
| Stop |  |  |  |
| SetPosition | position | int64 | input |
| GetPosition | position | int64 | output |
| SetPlaybackRate | rate | double | input |
| GetPlaybackRate | rate | double | output |
| SetVolume | volume | int32 | input |
| GetVolume | volume | int32 | output |
| SetWindowId | window_id | double | input |

# API Introduction

UMMSMediaPlayer Methods:

| Function Name | Parameter Name | Parameter Type | Direction |
|---|---|---|---|
| SetVideoSize | y | uint32 | input |
| | w | uint32 | input |
| | h | uint32 | input |
| GetVideoSize | w | uint32 | output |
| | h | uint32 | output |
| GetBufferedTime | buffered_time | int64 | output |
| GetBufferedBytes | buffered_bytes | int64 | ouput |
| GetMediaSizeTime | duration | int64 | output |
| GetMediaSizeBytes | length | int64 | output |
| HasVideo | has_video | boolean | output |
| HasAudio | has_audio | boolean | output |
| IsStreaming | is_streaming | boolean | output |
| IsSeekable | seekable | boolean | output |
| SupportFullscreen | fullscreenable | boolean | output |
| GetPlayerState | state | int32 | output |
| SetProxy | Param | a{sv} | Input |

## Agenda

- What is UMMS?
- Typical use cases
- API  introduction
- A First Implementation
- Conclusion

## A First Implementation

- Initially targeting MeeGoTV
  - Running on Intel CE4100
  - Basic netbook support is included
- The work that's being done:
  - Define the APIs
  - Framework design
  - Implement a backend (using Gstreamer)
  - Sample application

## Initial Implementation

- D-Bus service
  - system bus

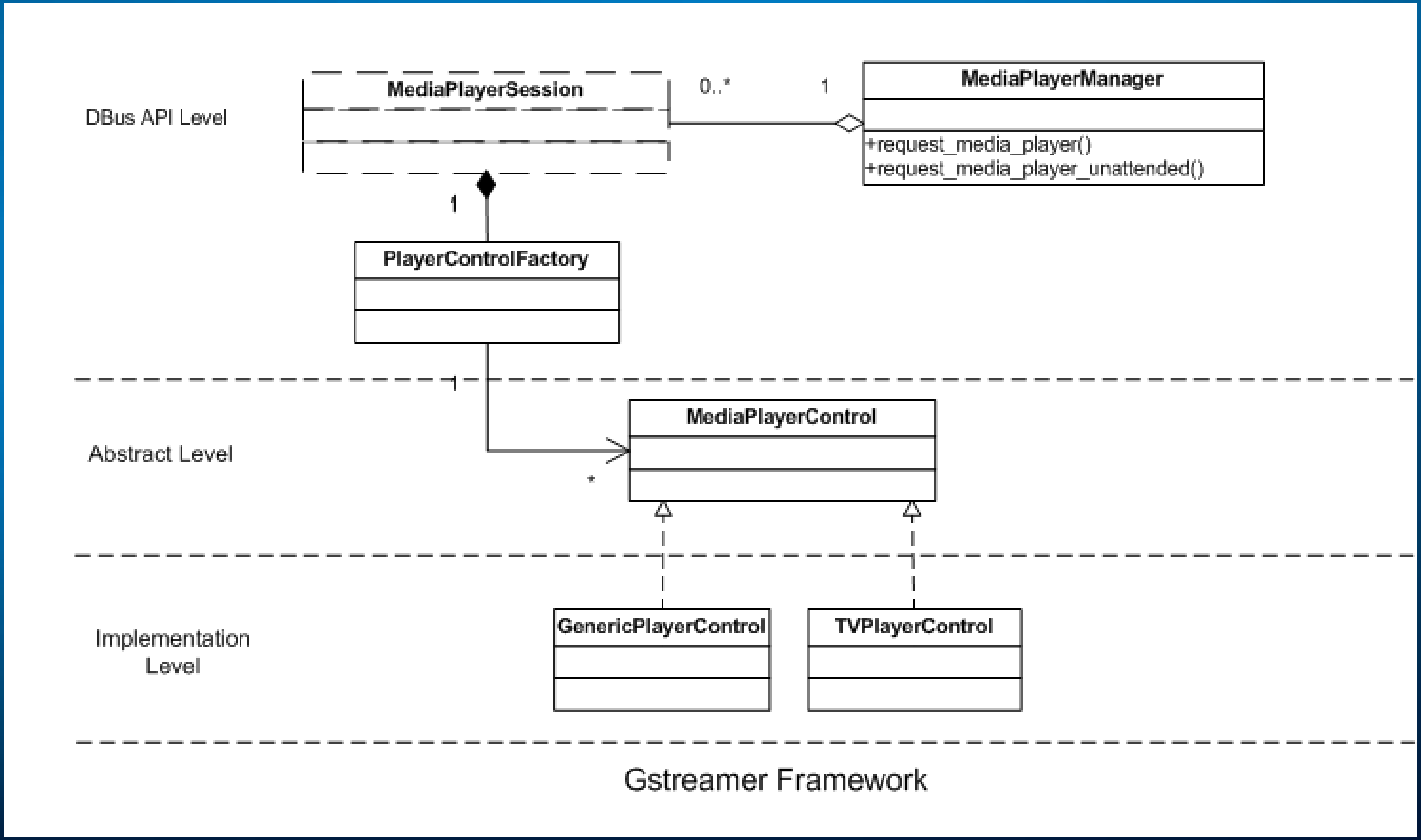
    Provides service for all users simultaneously.

  - related files
    - /etc/dbus-1/system.d/com.UMMS.conf
    - $(datadir)/dbus-1/system-services/com.UMMS.service
    - $(libexecdir)/umms-server
  - service name

    com.UMMS

## Initial Implementation

- Class diagram



## A First Implementation

- The implementation is not complete but the baseline is there
- Further improvements planned:
  - Dynamic pipeline creation and loading
  - Declaration of URI handling capability
  - Capability rank
  - Generic Resource Management Framework

## Agenda

- What is UMMS?
- Typical use cases
- A First Implementation
- API  introduction
- Conclusion

## Conclusions

- UMMS is a service that proposes a unified MultiMedia API across devices and hardware
- A draft specification for UMMS is available
  - http://wiki.meego.com/File:Meego_Unified_MultiMedia_Service_V0.4.odt
  - There were some initial discussions on meego-dev and meego-tv mailing list with great feedback
- A First Implementation is also available
  - Open-source code: available on MeeGo OBS

## Some Resources

- Our initial Code for UMMS is in the MeeGo public OBS
  - Search for the 'umms' package
- Wiki: http://wiki.meego.com/Umms
  - General architecture: http://wiki.meego.com/UMMS_Architecture
  - User Manual: http://wiki.meego.com/UMMS_User_Manual
- Latest draft specification:
  - http://wiki.meego.com/File:Meego_Unified_MultiMedia_Service_V0.4.odt
- There was an initial round of feedback on *meego-dev* and *meego-tv* mailing lists

# Thank You!