Grant Agreement No.: 687645
Research and Innovation action
Call Topic: H2020 ICT-19-2015

ORPHEUS
Object-Based Audio Experience

Object-based broadcasting – for European leadership in next generation audio experiences

D3.6: Implementation and documentation of object-based editing and mixing

Version: V1.0

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	30/11/2017
Submission date	30/11/2017
Lead editor	Tilman Herberger (MAGIX)
Authors	MAGIX: Tilman Herberger, Dr. Volker Mühle, Marius Vopel BBC: Matt Firth IRCAM: Olivier Warusfel
Reviewers	Andreas Silzle (FHG), Chris Baume (BBC), Werner Bleisteiner (BR)
Work package, Task	WP3, T3.2
Keywords	audio production, audio mixing, object-based broadcast

Abstract

This document describes object-based production tools, which were created as part of WP3. It offers details on the challenges and results of implementing features that are required for editing and mixing in an object-based production environment.

[End of abstract]

Document revision history

Version	Date	Description of change	List of contributor(s)
v0.1	18/10/2017	Table of contents, document structure	Tilman Herberger (MAGIX)
V0.2	20/10/2017	Topics filled from developers, Executive summary included	Dr. Mühle, M. Vopel
V0.3	27/10/2017	Filled first chapters	T. Herberger
V0.4	14/11/2017	Drawings and screens added	T. Herberger
V0.5	17/11/2017	All MAGIX screens inserted	T. Herberger
V0.6	18/11/2017	BBC content inserted	T. Herberger / Matt Firth
V0.7	20/11/2017	Additions and corrections	M. Vopel
V0.8	26/11/2017	IRCAM contribution inserted	O. Warusfel, T. Herberger
V0.9	27/11/2017	Further additions and corrections	M. Vopel
V0.94	28/11/2017	Addressed internal review feedback	T. Herberger, M. Vopel
V1.0	30/11/2017	Final summary edited	T. Herberger, M. Vopel

Disclaimer

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive Summary

This document describes the challenges which had to be solved to realize an object based workflow for editing and mixing audio – in a pre-production situation as well as in a live mixing environment.

The DAW Sequoia had to support new object-based features and workflows as proposed in the ORPHEUS project.

The most important enhancement is the support for BW64 files with ADM metadata. Basic import, export and editing functionality have been implemented. Various improvements to the internal surround and mixing engine of Sequoia were made in order to allow 3D panning, a new recording workflow and an integrated rendering solution.

In order to create live object-based audio productions in the IP Studio software, it was necessary to design and develop a new audio production interface since no commercial product exists to provide this functionality at the time of writing. This tool would allow the operator to design and control his or her object-based soundscape, and would generate the necessary representative metadata to recreate the experience on client devices.

The ADMix tool suite has been already introduced in deliverable D3.5. Originally developed for experimenting with incorporating reverb into an object-based production workflow, the ADMix tool suite can be used in a more general context to create, load and play back audio files containing ADM metadata. In its current state, only a subset of the ADM specifications is supported, but many common use cases are already covered.

While there is still room for future improvements, the currently implemented solutions can already be used for the required workflows in object-based production.

Table of Contents

List of Figures

Abbreviations

ADM	Audio Definition Model, ITU-R BS.2076-1
AR	Augmented Reality
DAW	Digital Audio Workstation, e.g. MAGIX Sequoia
ES2016	ECMA Script Language Specification
HOA	Higher Order Ambisonics
MPEG-H	MPEG-H 3D Audio Standard, ISO/IEC 23008-3 (MPEG-H Part 3)
N3D-ACN	One of the common Normalisation and Ordering conventions used for HOA streams. N3D stands for full 3D normalisation and ACN for Ambisonic Channel Number.
SN3D-ACN	One of the common Normalisation and Ordering conventions used for HOA streams. SN3D stands for Schmidt semi-normalisation and ACN for Ambisonic Channel Number.
MPEG-4	MPEG-4 Standard, ISO/IEC 14496 – Coding of audio-visual objects
UMCP	Universal Media Composition Protocol
VBAP	Vector-based amplitude panning
VIP	Virtual Project, standard project format for MAGIX Sequoia DAW
VR	Virtual Reality

1 Introduction

Starting to produce content in an object-based broadcast environment comes with a number of new considerations for audio engineers. It is essential to provide them with the appropriate tools, in order to make this transition as comfortable as possible.

This document describes object-based production tools, which were created as part of WP3. It offers details on the challenges and results of implementing features that are required for editing and mixing in an object-based production environment.

Chapter 2 describes the development process of MAGIX' DAW Sequoia to new object-based features and workflows. Following that, BBC R&D's newly created live mixing tool is presented in Chapter 3. Finally, Chapter 4 provides updated and more detailed information on IRCAM's ADMix tools, previously covered in D3.5.

2 Extensions to the DAW Sequoia for object-based editing and mixing

Sequoia is an object-based audio workstation from ORPHEUS partner MAGIX. Its roots go back into the early 1990s when their first audio editor Samplitude was created.

From the very beginning Samplitude allowed the use of clip-based real-time effects – so called "object effects", which was a big advantage compared to widely used track-based effects, especially for mastering purposes. Even in multi-track productions these clip-based effects helped to reduce the number of tracks and increased the overall clarity of projects.

Sequoia was derived from Samplitude around 2000 as an extended version, specifically targeting workflows in a broadcast environment. It contains a variety of enhancements such as database support, 4-point cut, a professional crossfade editor and "MuSyC", a special cutting principle commonly used in classical music productions.

Based on this set of features, Sequoia is the ideal tool to be adapted to the new needs of object-based broadcasting in the context of the ORPHEUS project. The following sections show the challenges of this process as well as the implemented solutions to provide new object based workflows for editing and mixing. More detailed step-by-step explanations of the new workflows are provided in Appendix A.

2.1 Mapping the ADM to the project structure of Sequoia

Sequoia projects are so-called VIPs (virtual projects), which may contain any number of tracks and busses. Each one of these tracks can hold any number of audio clips – virtual references to audio files. Real-time effects can be applied to tracks, busses and the master section in the mixer. Tracks can be grouped using so-called folder tracks, providing users with a way of organizing large projects for a better overview. So far this is a common setup applicable to most DAWs.

Sequoia's speciality however is the ability to process real time effects for each clip. This requires a very flexible audio processing engine, since each clip on every track can hold its own chain of internal effects as well as external VST plugins.

Being able to use effects on a per-clip-basis can be very helpful especially with lengthy projects, where different parts of the audio need different effects. Without this feature, a high number of tracks would need to be used, making it harder to manage an already large project. This workflow is often used for mastering of whole CDs, DVDs or in classical music production with hundreds of clips per musical piece.

A simplified model of Sequoia's project structure with its four levels of hierarchy is depicted below in Figure 1.

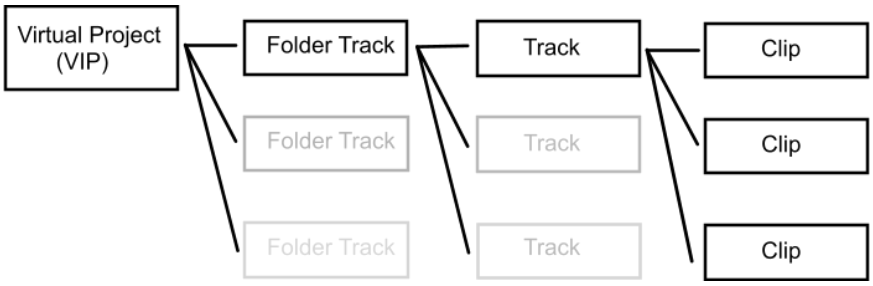


Figure 1: Internal structure of a Sequoia project

By comparison the ADM format allows for much more flexible ways of describing and organizing audio and further associated metadata. There is no fixed hierarchy system, since objects can be nested arbitrarily (see Figure 2).

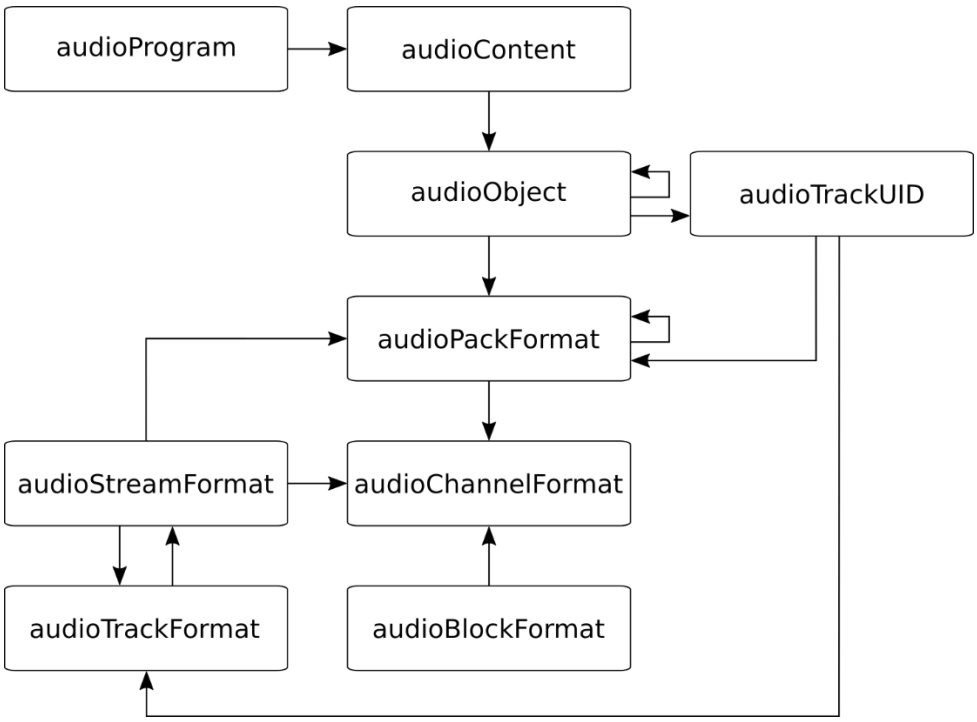


Figure 2: ADM structure

Therefore it was a non-trivial task to decide on the best way of translating this model to Sequoias internal project structure. At first it seemed like the obvious solution would be to map ADM objects to Sequoia clips, but after discussing the most important use cases for ADM projects, it was decided to have Sequoia tracks fulfil this role (see Figure 3).

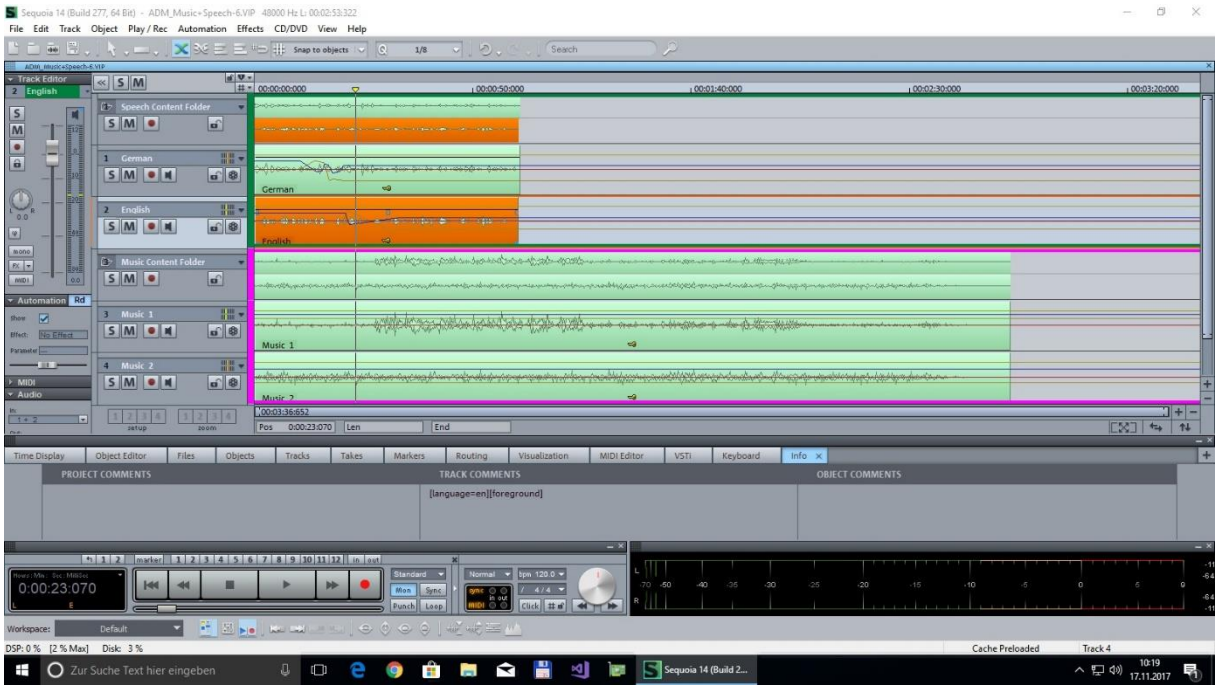


Figure 3: VIP with open ADM project

The following factors brought us to this decision:

User Experience for audio engineers: We wanted to make the ADM integration as transparent and comfortable as possible for audio engineers. They typically use tracks to organize large productions

and assign panning information and other metadata to these tracks.

Integration of interactivity features: The ADM allows for the definition of a variety of interactivity features, giving the end client the means of presenting a more individualized experience. Within the ORPHEUS project the focus has been on language switching and foreground/background balancing. In both cases the audio material is typically represented in parallel tracks. Representing ADM objects as clips would likely make the editing of these features rather tedious.

Current technical limitations: A normal track bus in Sequoia as well as any clips placed on it cannot hold more than two audio channels and one set of automation parameters. An ADM object however may contain any number of audio channels with individual pan and gain data. Representing ADM objects as clips would require very extensive and time-consuming modifications to Sequoia's internal structures.

The sum of these factors led to the decision of representing ADM objects using tracks and ADM content elements using folder tracks as a way of grouping objects together.

2.2 Automation of object parameters

One issue, which had to be overcome, was the import of parameter automation data from ADM files. The ADM format does not support continuous curves for applicable parameters such as gain or pan.

Instead it defines blocks with a set duration, start time relative to the parent object, a set of values for these parameters as well as information specifying if and how any linear interpolation between the preceding and current blocks should be performed (see Figure 4).

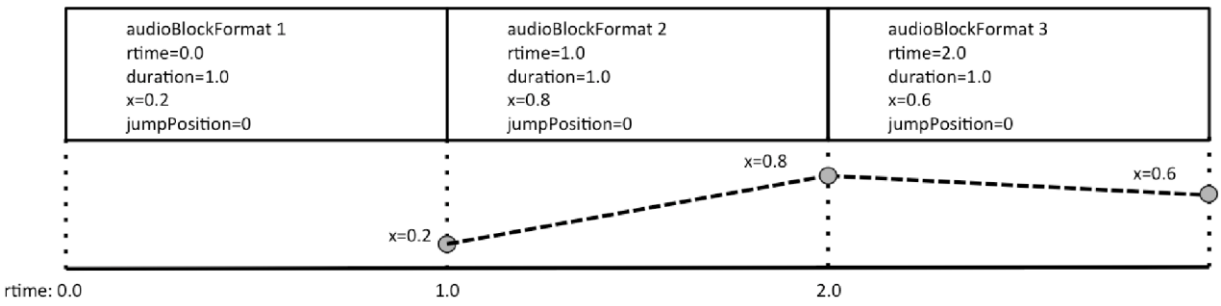


Figure 4: Visualization of audio blocks in ADM [8]

For the initial ADM import implementation these audio blocks were directly converted to audio clips in the DAW project, each with their respective parameter data assigned. For simple demonstrations this worked fine, but importing more complex productions would result in generating possibly hundreds or thousands of small independent audio clips per track. Starting to make edits from here would be very frustrating for an audio engineer. For an ADM export the engineer would also need to make sure that any clips in the project that use automation are split up sufficiently, because only one value per clip would be written to the resulting ADM block.

In the current implementation all sequential audio blocks are merged into one clip in the DAW and only one set of parameter curves per track is generated from all block parameter data of an object channel. With this solution the editing can take place in the usual convenient way. During export these audio clips are then automatically split into blocks corresponding to any existing automation curves (see Figure 5).

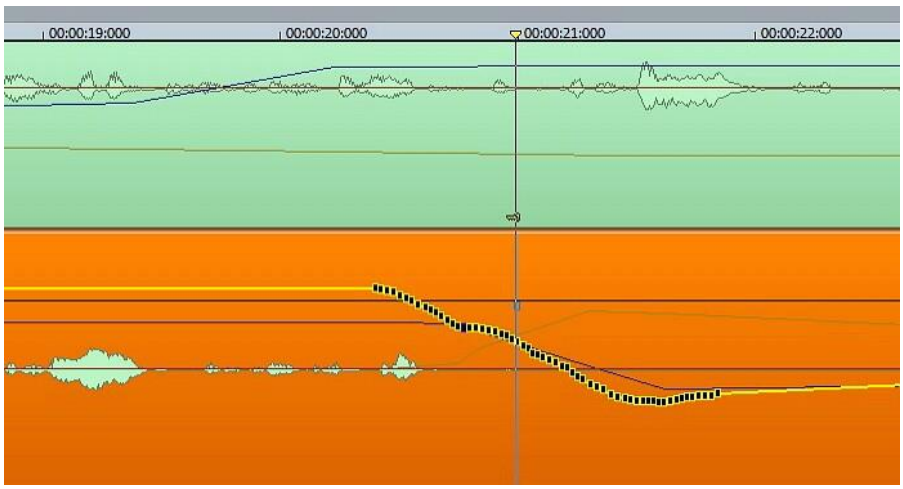


Figure 5: Sequoia parameter curves

2.3 Introducing 3D panning

Another important enhancement of Sequoia to support editing and mixing in an ADM context was the introduction of 3D surround panning. The existing x-y-surround panning was extended by a new z-coordinate, since ADM audio objects describe their surround position in three-dimensional space, even if their position is static or moves in only two dimensions.

Further, new speaker presets were added to support common 3D loudspeaker setups, such as 4+5+0 or 4+7+0[2]. The user interface for panning had to be enhanced as well. The z-coordinate is now represented with a slider and the current height of the source and speakers are visualized using coloured circles (see Figure 6).

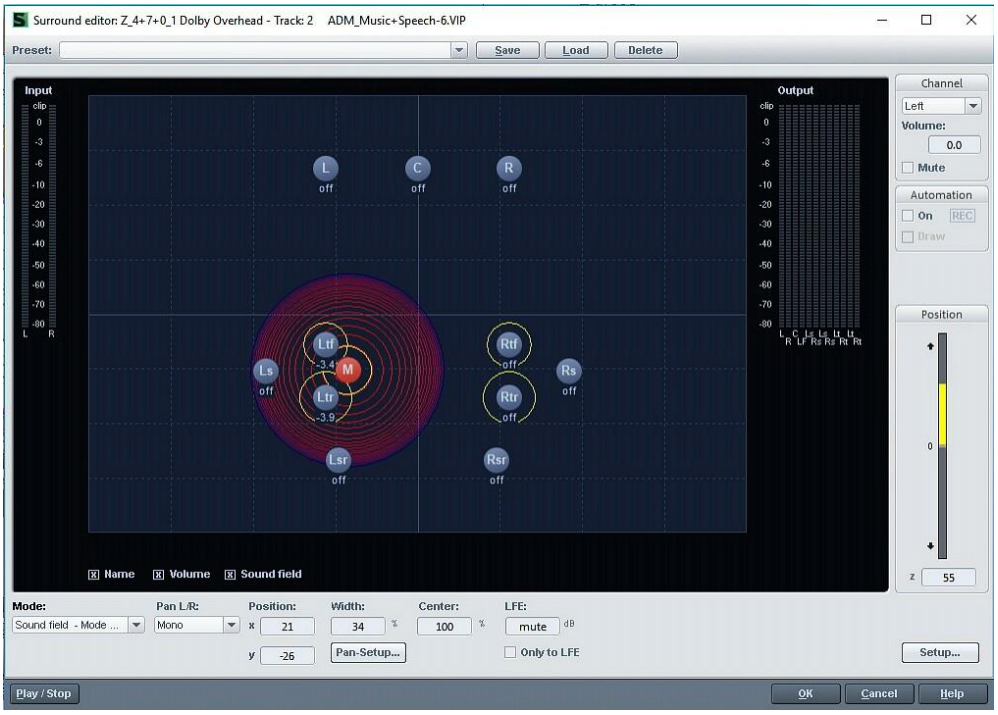


Figure 6: 3D surround editor

[2] ITU notation for height+midlle+buttom layer

To create better 3D-panning results using Sequoias built-in surround module, a new panning algorithm based on the VBAP approach (vector-based amplitude panning) was implemented. VBAP ensures that the audible energy of an audio signal is kept constant regardless of its position i.e. an object can be moved freely without creating unexpected volume changes.

As another option, third party panning plugins such as the "WAVES 360 Surround Tools" (see Figure 7) may be used to control the 3D panning of an audio signal. Sequoia's support for the corresponding aspects of the VST interface were improved to make this a possibility.

Figure 7: Waves surround tools plug in

2.4 Recording to ADM

One of the aspects of the production workflow within the envisaged architecture of ORPHEUS is the recording and processing of 3D audio signals. For that purpose ORPHEUS partner b<>com developed a set of VST plugins. To offer a simple way of feeding such recordings to the rest of the chain, a special recording workflow was developed.

The aim of creating this workflow was to provide a simple approach for:

1. Using input from an Eigenmike (mh acoustics) consisting of 32 channels,
2. Processing this input using b<>com's plugins to create an HOA representation of the recording and in turn to create a speaker rendering from the HOA signals and
3. Writing the results to an ADM file.

This posed several challenges to Sequoia.

Firstly, the width of the surround master bus had to be expanded from the previous 12 to 32 channels. With this enhancement and some improvements of the handling of multichannel effects plugins, the 32 input signals could be routed to this bus and the processing plugins could be inserted on it. At this point playing back the speaker rendering was possible. This signal flow is illustrated below in Figure 8.

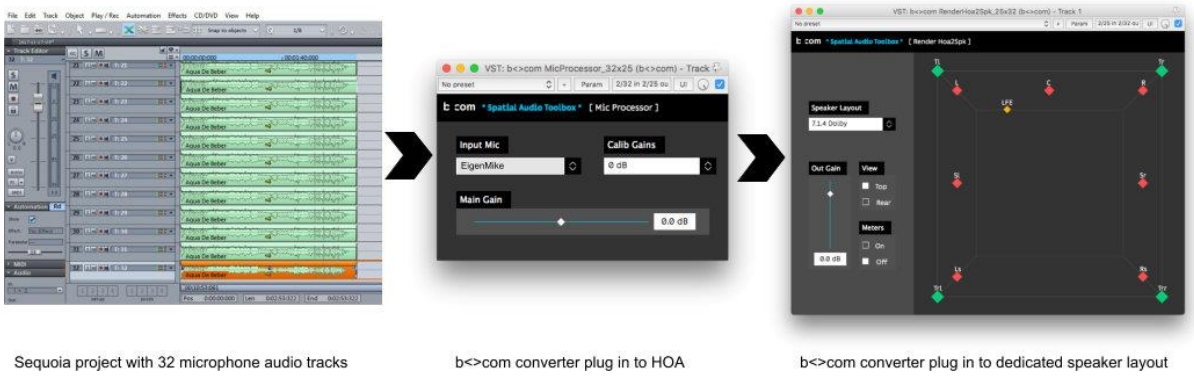


Figure 8: Recording workflow using b<>com plugins

For the final step a special behaviour of the ADM export functionality was implemented. Creating a folder track with a specific label next to the input tracks changes the export in the following way: The audio portion of the export is generated using the output of the surround master bus instead of performing the usual track bounce. However the accompanying metadata is based on the folder track. By adding tracks with pan positions that reflect the chosen target speaker layout, the user can now render and export the recording to an ADM file at once. The extra steps of creating the folder track and its contents can also be done once beforehand and then be saved as a project template of the given speaker layout for future use.

Section 2.10.3 provides a step-by-step guide for this newly created workflow.

2.5 Integration of a rendering solution

A fundamental aspect of object-based audio is that the final rendering is produced on the end-users device instead of the audio engineers mixing desk to allow for a listening experience, which is tailored to the end-user's individual circumstances, preferences and interactive choices. To produce suitable content, the audio engineer needs to be able to simulate the end-user experience as part of the pre-production process in order to guarantee the best possible audio quality in all cases. Integrating a rendering solution directly into the DAW would allow to monitor the different outcomes of mixing decisions in a way that is both seamless and familiar to the audio engineer.

For Sequoia, an integrated rendering solution was realized in form of a set of internal effects plugins. During playback, one plugin – the 'RenderCollector' – is inserted on all input tracks and aggregates audio buffers and the corresponding metadata (e.g. gain automation). The other plugin – the 'Renderer' – sits on the surround master bus and receives this data at once (see screenshot below). This is where the actual processing takes place. The results can be listened to from the surround master output (see Figure 9).

This behaviour bypasses parts of Sequoia's internal processing chain such as volume and pan processing. Other effects on tracks or clips can still be used freely. Creating and editing pan automation is still possible using the panning dialog since the resulting automation curves are read when aggregating data.

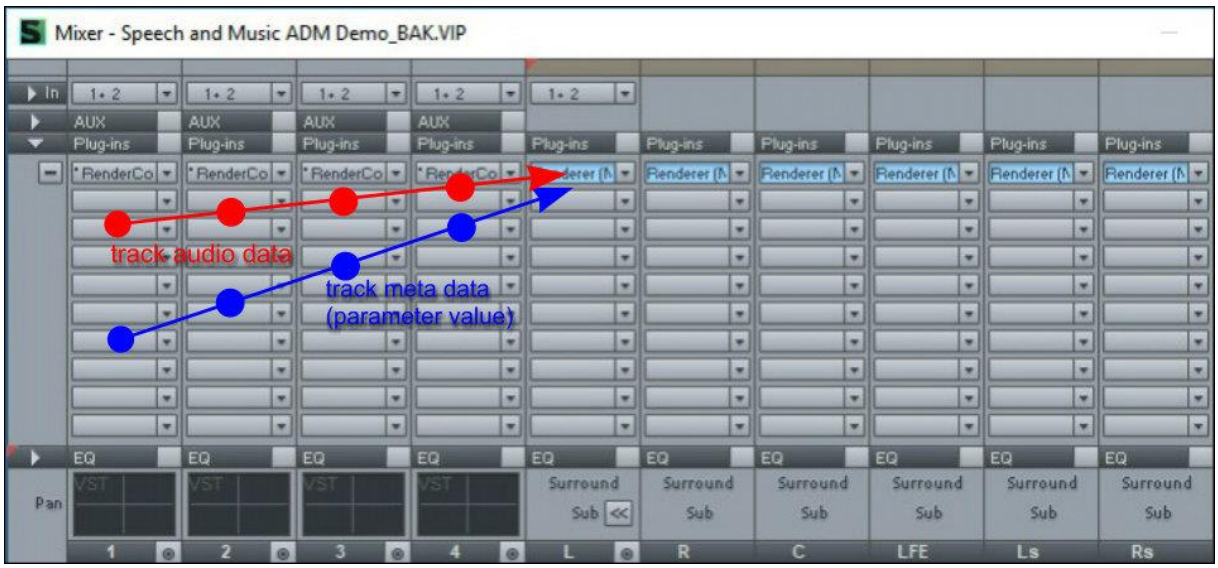


Figure 9: Signal flow of the renderer plugin

Initially it was planned to integrate an anticipated ITU baseline renderer, but no results were available in the task's time frame. In the end, rendering based on MPEG-H was implemented with support from ORPHEUS partner Fraunhofer IIS (see Figure 10). This approach has the advantage of the same rendering algorithms being used on the end-user device. The supported render target formats include standard 2D setups such as stereo or 5.1, 3D setups with height speakers such as 4+7+0 as well as the option of binaural output.

The implemented solution does have some caveats as well. The MPEG-H renderer expects the metadata in a format different from ADM and a direct format conversion presents a complex challenge. This is still being investigated at Fraunhofer. Currently only a very simple conversion is performed, but more complex features e.g. switching language versions are not implemented yet. Also the renderer comes with the limitations of MPEG-H in accordance with the corresponding complexity level. This affects a number of factors e.g. number of objects rendered in parallel.

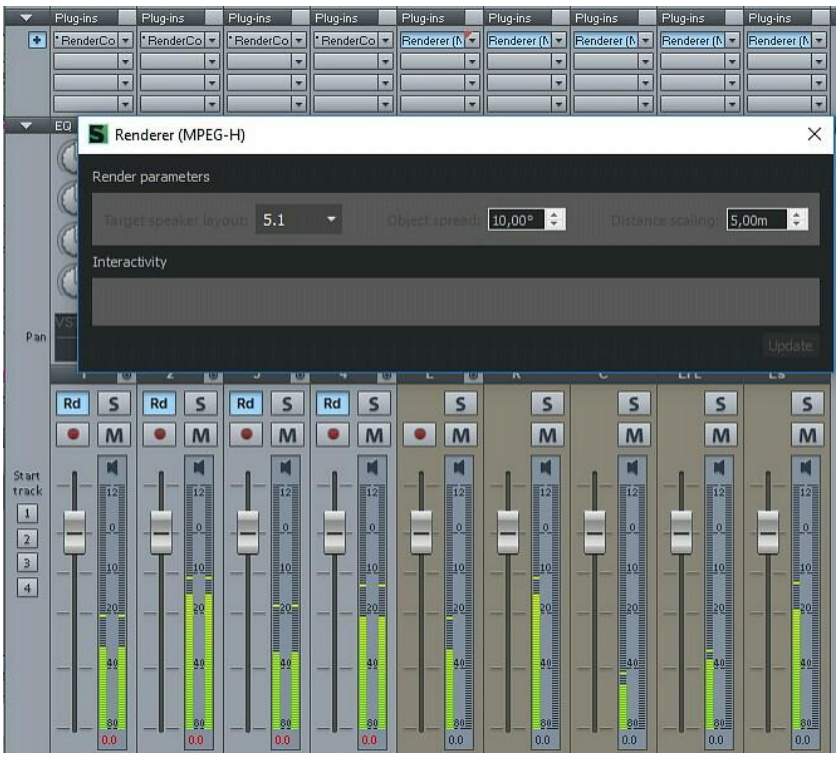


Figure 10: Rendering plugin based on MPEG-H

2.6 Further ADM metadata editing

While it was possible to find mappings of some parts of the ADM to Sequoia's internal structures, there are many more, which don't have an obvious equivalent in the DAW. A separate solution had to be found.

In Sequoia it is possible for the user to add comments to clips, tracks or whole projects. This functionality was created as a means of easily taking quick notes e.g. about mixing decisions that were made. Having this kind of information saved right within a project file is a convenient way of relaying thoughts or plans to another user in a collaborative setting or keeping this information for when a project is reopened years later.

The track comment field can now be used to access or add metadata to ADM objects by including certain tags as well. For multichannel objects, such tags need to be included on the first associated track only.

Feature	Tagging format	ADM relation
Language association	[language=<desired language>]	This tag sets the language of the ADM audioProgram, and, if multiple different language tags are used, creates additional audioPrograms for alternative language versions with the corresponding objects tied to them.
Foreground/background association	[foreground]	This is represented by generating additional objects labelled 'foreground' and 'background' without any referenced audio. They only serve an additional way of grouping the actual audio objects, which are nested below them. Only

		foreground objects need to be tagged. All other objects are assigned to the background implicitly. It should be noted that this is not a standardized field in the ADM, but rather a specific way of using it. The feature was added nevertheless, since it was deemed an important use case.
Importance	[importance=<0-10>]	This assigns an importance value to an object. Using this information a renderer might choose to discard an unimportant object e.g. when the load of processing gets too high.
Interactive muting	[onOffInteract]	This tag indicates that a user may mute the associated object interactively. A preprocessing tool might also use this information to render a project to a channel bed, while preserving interactive objects as separate elements.

Table 1: ADM tags supported by Sequoia

This overall approach works quite well, provided the user has the required background knowledge. Also, it is somewhat prone to error, since precise text input is required. A much-improved plan for editing metadata has been laid out in section 2.11, though the implementation process is still ongoing.

2.7 Introducing Variable Length

For pilot phase 2 of the project the ORPHEUS team decided to implement and demonstrate an important interactive use case: Programs with a variable length. In practice this means, that, given a certain level of interest, the same program can be listened to at different degrees of content depth and therefore different lengths.

During production the content is to be sectioned and tagged to indicate which sections fit to the following levels of interest:

- Level 1: A very short “headline” version, containing only the core message
- Level 2: A slightly longer version including all key information
- Level 3: An even longer version with additional background information
- Level 4: A more complete version, adding further details
- Level 5: The full version including all details, associations, etc.

These descriptions won’t necessarily fit for any production, but they provide a sense of how the levels are expected to be used.

The task of the DAW is to provide the user with means of editing levels of interest. For Sequoia this was realized in the form of markers on the project timeline, which can be set from the “marker manager”. A marker indicates that all content between its location and the location of the next marker are part of a certain interest level. By using keyboard shortcuts, they can be added easily during playback. For an example with added markers see Figure 11.

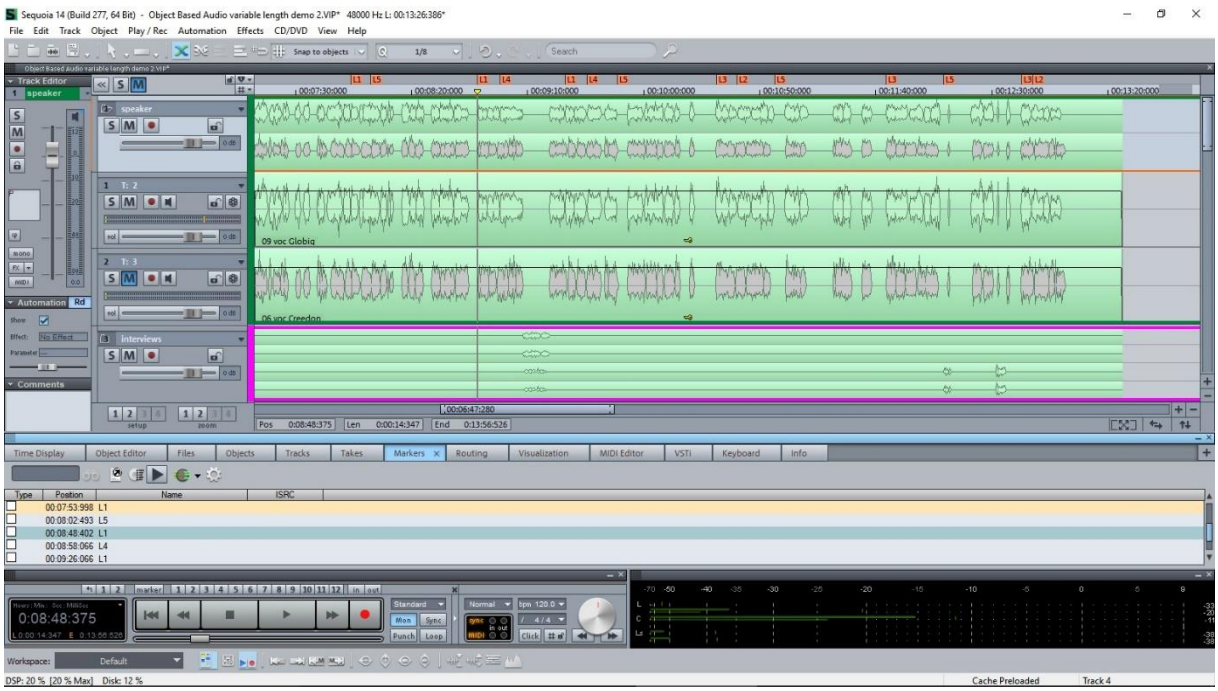


Figure 11: "Experience object-based audio" created by BR with added markers for interest level

In addition, a preview mode has been implemented. By selecting a marker of any interest level, only the sections of that level and levels below it will be played. Sections marked with a higher interest level will be skipped.

For the ADM export these markers will be stored in the metadata chunk of the BW64 file conformant to the EBUCore definition[27]. This information will then need to be transported further along the broadcast chain to eventually be used on the end-user device.

As a future enhancement it seems to be useful and possible to mark also complete tracks / ADM objects with interest levels. This could be used for improved handling of background music or atmosphere audio.

2.8 Performance optimization

One of the most important features of any DAW in the audio production workflow is precise and stable playback. The nature of ADM projects introduced some challenges for the real-time play engine of Sequoia:

- 3D audio recordings with 32 channels in combination with CPU intensive plugins per channel
- handling large numbers of tracks with lots of automation data
- real-time skipping of project parts for projects with variable length

In order to accommodate these demands the playback engine was optimized for performance in several areas.

To handle the large amount of metadata a special metadata cache was implemented. This cache gives the low latency engine fast access to the needed metadata of the actual play position.

The prefetching engine of Sequoia was improved to better utilise multi-core processors. This results in a smoother playback of large projects, provided a CPU with 4 or more cores is being used.

Thanks to these intensive optimization efforts all ADM projects of the ORPHEUS project can be edited comfortably and played back without issues on a typical Windows 10 Quadcore desktop PC using ASIO low latency audio.

2.9 Outlook: Metadata editor

While the current mechanisms for editing metadata work fine in the context of a research project, they present several issues, which make them unsuitable for a product release:

- Information that will contribute to ADM metadata generation is spread between several places in a project.
- Using free text input requires a lot of background knowledge at the user end, is sensitive to typing errors and can be difficult to validate.
- Currently the editing is still quite clear, but only a limited set of ADM attributes are supported. Adding support for more features would make it hard to keep track everything and eventually render the process unmanageable.

To address these issues, a concept for a dedicated metadata editing module for Sequoia was created. The implementation process is currently ongoing. The following image shows a mock-up view of the planned Sequoia ADM metadata editor (see Figure 12).

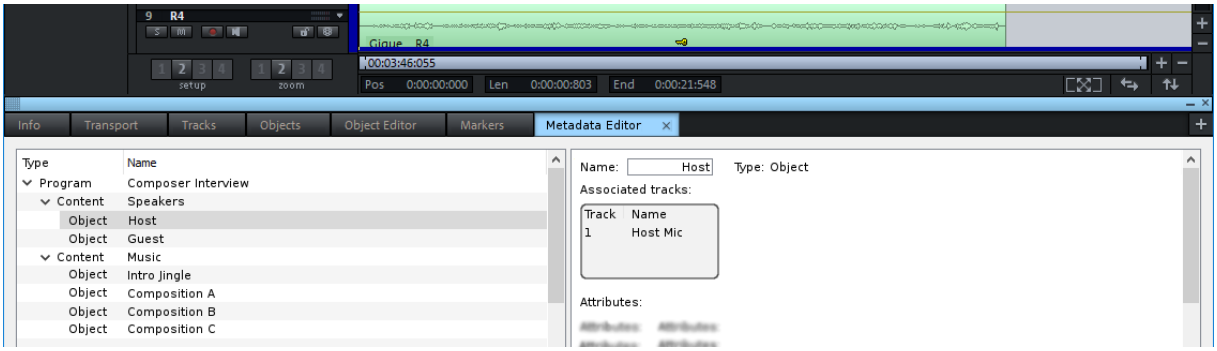


Figure 12: Metadata editor concept

The metadata editor is designed to bring the following advantages with it:

- All ADM metadata can be accessed and edited in one place.
- No special background knowledge of keywords is required. User input is directed by the GUI.
- The editor presents the object hierarchy directly and tracks of the DAW can be referenced by the objects.
- Logical checks and warnings as well as ADM templates can be integrated.
- The editor can be easily maintained and expanded for future ADM developments.

3 IP Studio: An Object-Based Audio Production tool

This section introduces and details the implementation of an object-based audio control interface of BBC's IP Studio software, which was designed and used for ORPHEUS pilot phase 1A.

3.1 Object-Based Media Production

When producing an audio programme traditionally, an audio mixing console within a studio receives various incoming audio sources. These could be real-time sources (such as from microphones) or pre-recorded sources provided by some form of playout system or cart player. The console combines these sources in to a single audio signal according to the control input of the console operator. This is an irreversible process without employing complex source separation techniques, which may still have very limited success. Essentially this means that all consumers receive the same, fixed audio signal, regardless the consumers listening preferences or their device capabilities.

With object-based audio production, each audio source is sent to the consumer individually accompanied by a stream of metadata. In essence, the 'mixing console' is no longer required to mix audio, but simply to generate the representative metadata. The responsibility of actually mixing or 'rendering' the audio is then shifted to the consumer's device. This means the mix can be adapted as required for that particular consumer. The audio may be rendered according to the representative metadata and/or options under the consumers' control and/or their device capabilities.

As an example of device-oriented rendering, a device which provides only a headphone output would render the audio in stereo, or perhaps binaurally to best recreate the three-dimensional soundscape. For multichannel speaker layouts, the audio would be rendered to make optimal use of the speaker configuration in order to replicate the soundscape as accurately as possible. This capability of object-based audio delivers a significant advantage over existing workflows since the sound engineer no longer needs to provide a bespoke mix for each of the channel configurations they wish to support.

An example of consumer-controlled rendering would be providing the ability to affect the contribution of individual elements of the audio mix through UI controls. A consumer who is hard of hearing may choose to turn down the music or ambience in a drama in order to hear the dialogue more clearly, or perhaps a Welsh consumer may choose to swap the English commentary of a football game with Welsh commentary.

The same 'object-based' concept can be applied to other types of media, such as video (for example, the BBC R&D 'Forecaster' project [[4]]) or even text (such as in the 'Atomised News' project [[5]]).

3.2 Challenges

The concept of object-based audio has been used for decades in the computer games industry and many projects have utilised object-based principles for the customisation of other experiences in the past. The major challenge in this case was to establish a functioning broadcast chain to deliver object-based audio experiences live. This is entirely unprecedented and marks a significant step towards being able to deliver next generation audio and new content experiences to our consumers.

As with all projects, BBC R&D were keen to use existing standards where possible.

3.3 Interface Design

The interface operates as a single page web application. It is intended to be used on a touchscreen in full screen mode at a display resolution of 1920x1080 or greater. However, it is designed to be responsive and so will make best use of the screen space available.

The interface is divided in to three horizontal sections as illustrated by **Error! Reference source not found.**.

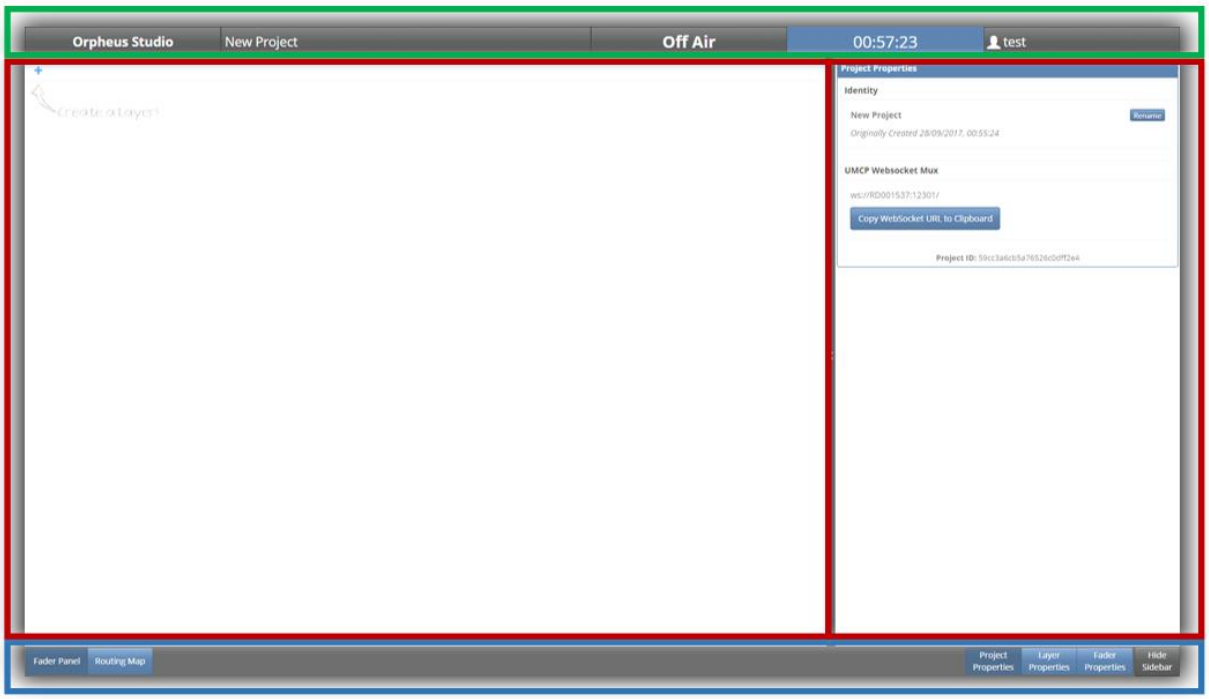


Figure 13: Interface Layout

The header (outlined in green) displays the selected studio, the open project, the broadcast state, the time, and the current user. All of these components are clickable.

- The studio component presents a menu to select or configure a studio.
- The project component presents a project selection menu.
- The state component presents a menu to select between 'On Air' 'Rehearse' and 'Off Air'.
- The time component will momentarily show the date.
- The user component presents a menu to log out or access user options.

The midsection (outlined in red) can be split in to two sections, showing a sidebar to the right and a main section to the left. Alternatively, the sidebar can be hidden so that the main section can occupy the entire width of the screen. The main section has different modes which present different user interfaces to configure and control the project. The sidebar also has various modes for configuration of faders, layers or the project itself. The midsection is used in its entirety when presenting the Studio Configuration or User Options screen.

The purpose of the footer (outlined in blue) is to set the mode of the midsection. Buttons to the left of the footer control the mode of the main section, and buttons to the right of the footer control the mode and the visibility of the sidebar.

3.3.1 Fader Panel mode

This interface layout is intended to feel familiar to a sound engineer or desk operator by mimicking a traditional mixing interface. A screenshot is provided in Figure 14.

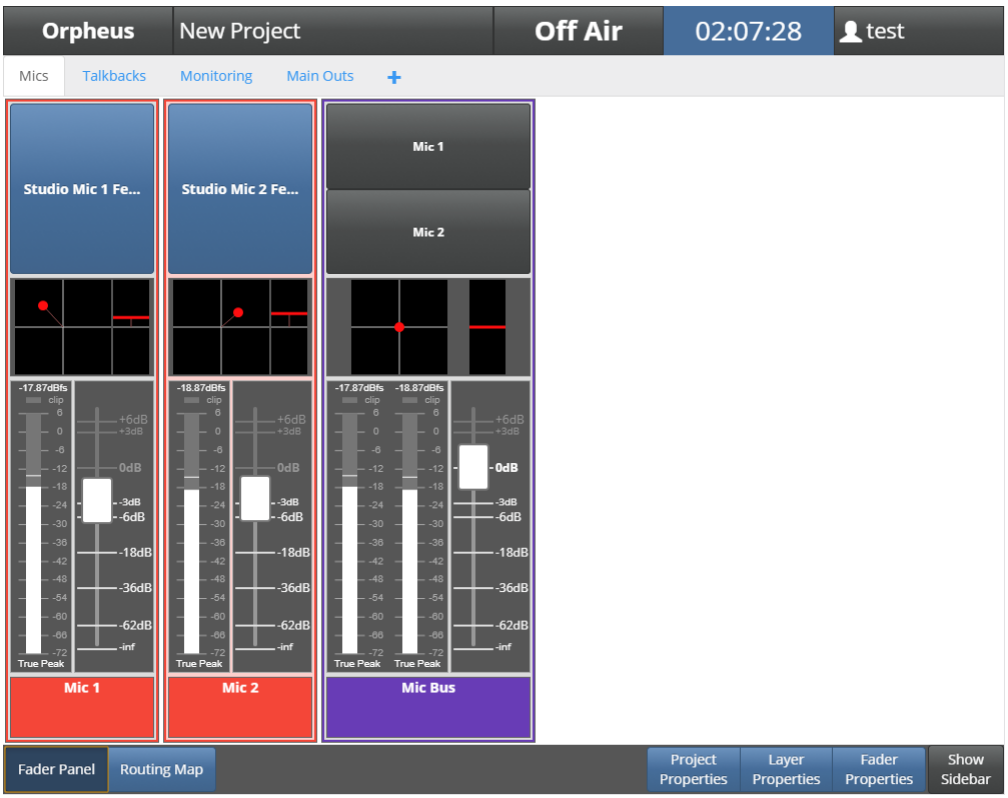


Figure 14: Fader Panel interface

Faders are presented in channel strips with accompanying spatial position controls. The input sources are presented as buttons at the top of each channel strip allowing each source to be switched in and out of the faders mix. These buttons are colour-coded according to the type of source and the source status; black indicates a feed from the output of another fader, blue indicates a real-time audio feed (such as a microphone), and a green indicates a cart (pre-recorded audio clip). A red source warns the user that the source cannot be located, and orange warns that the tool cannot communicate with the playback application assigned for a pre-recorded source. When a pre-recorded source is playing, the associated button will also display a progress bar.

As with many physical mixing consoles, faders can organised across different layers to simplify the workspace. Faders can even exist across multiple layers.

3.3.2 Routing Map mode

This mode is designed to assist with the routing of audio objects within a project by providing a visual representation of the routing graph and the ability to edit routes by simply interacting with the graph. A screenshot is provided in Figure 15.

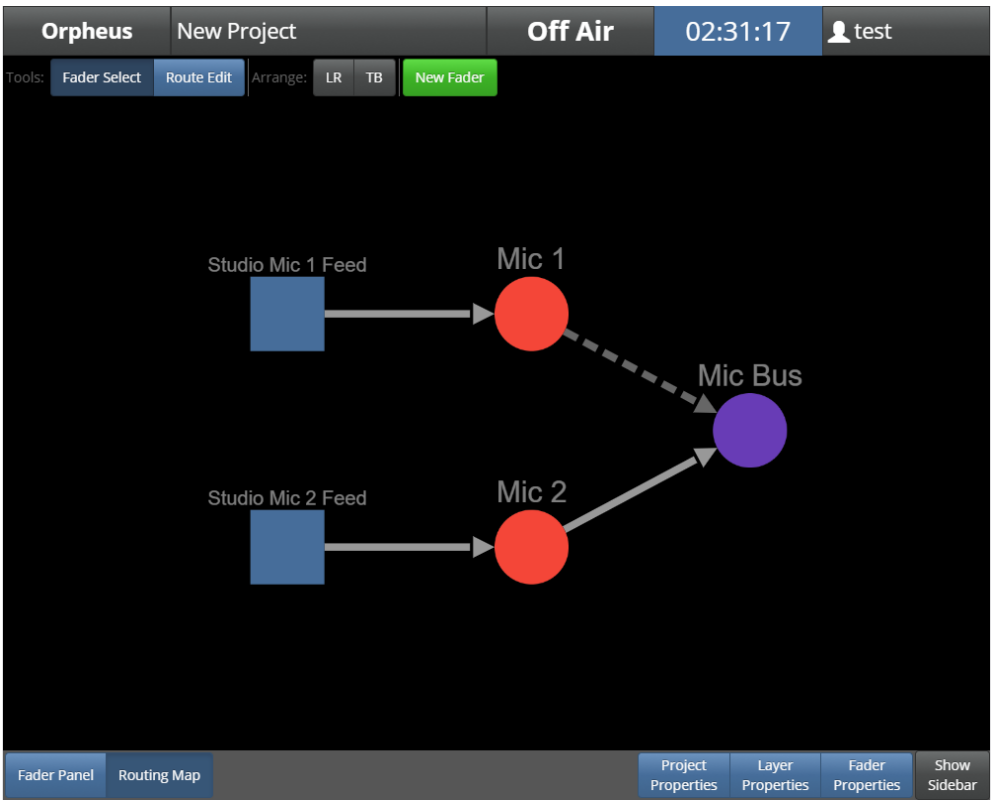


Figure 15: Routing Map interface

Circular nodes on the graph indicate faders. Square nodes indicate audio sources, with blue representing a real-time source and green indicating a cart (pre-recorded source).

The graph is interactive; users can drag-and-drop nodes, the pan the view by dragging, and zoom by using either a scroll-wheel or by using a pinch gesture on a touchscreen display.

With the Fader Select tool active, edges between nodes are solid if the source on the destination fader is active, otherwise they will be dashed. Clicking on a fader using this tool will bring up the Fader Properties sidebar. Clicking on an edge will toggle the active state of the associated source on the destination fader, and clicking on a cart source node will active the source and begin playback.

Edges will turn red as a warning when the Route Edit tool is active. Using this tool, routes can be removed by long-pressing/clicking on an edge. Routes can also be created by tapping a source node (which will begin flashing to indicate that it is selected) followed by the destination node. Tapping the source node again whilst it is selected (flashing) will unselect it and no route will be made.

3.3.3 Sidebar

The sidebar is a resizable pane positioned to the right of the interface which can be hidden when not in use. The sidebar is used for configuration and customisation purposes, therefore providing three modes; Project Properties, Layer Properties and Fader Properties as shown by the screenshots in Figure 16

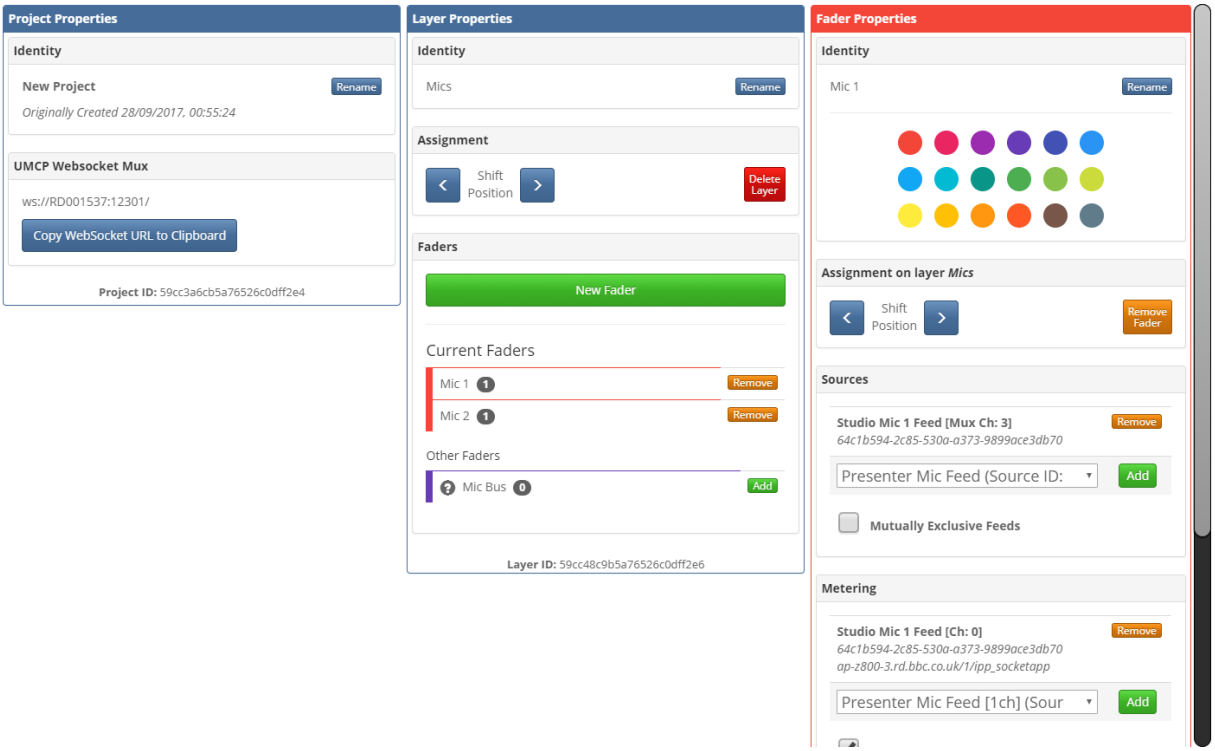


Figure 16: Sidebar modes

These panels are quite self-explanatory, but include some useful features worth noting.

The Fader Properties panel allows faders to be colour-coded for ease of identification. This colouring is reflected in the panel border. Sources and meters can be added to faders by selecting them in the provided dropdown boxes and clicking on the 'Add' button. Added sources and meters are listed in the panel and can be removed with the buttons provided. Options in the dropdown boxes are colour-coded similarly to source buttons. In the sources pane, a checkbox is provided to toggle mutually exclusive behaviour of sources on a fader. When checked, activating one source on a fader will deactivate all of the others. The metering pane has a feature can list meters per source, or per source channel. There is also the option to add 'triggerables' to the fader.

The Layer Properties panel provides a simple, easy to use method to populate the current layer with faders and to organise them. New faders can be created on the layer using the button provided. All existing faders within the project are listed in rows, separated according to whether they exist on the current layer or not. Each row provides the label of the fader and is are colour-coded accordingly. The number next to each fader label indicates the number of layers that the fader is present on. Faders can be add to, or removed from the current layer with the buttons provided. Faders on the layer can be rearranged by simply dragging-and-dropping rows within the list.

3.4 Working Practices

To simplify workflows as much as possible, it was decided early on that the production interface wouldn't provide any specific signal bussing or grouping tools. Instead, faders would simply support multiple input sources to achieve the same functionality. Additionally, a fader would be able to feed another fader as a source. This enables complex routing graphs to be formed using just faders.

Faders can even act as signal switchers since any source on a fader can be activated or deactivated on the fly. The 'mutually exclusive' mode allows only one source to be active at a time, providing automatic clean switching.

3.5 Technical Information

The production tool was developed in two halves; a backend and the user interface (frontend). These are stored in the orpheus-audio-control-backend and the orpheus-audio-control-interface GitHub repositories respectively [[6]][[7]].

The backend application manages the project database, performs user authorisation, tracks client sessions, and incorporates a web server (HTTP) and a WebSocket server. The user interface is delivered to the client via this web server on port 90.

The frontend is responsible for;

- presenting a functional user interface,
- establishing and managing a WebSocket connection to the backend,
- sending commands to the backend and acting on received messages/responses,
- and connecting to the UMCP API and generating relevant composition data.

3.6 System Architecture

BBC R&D's IP Studio is used as the underlying platform for the Orpheus project. It is used for source acquisition and discovery, signal analysis (metering), signal routing, format conversion, and in-studio rendering for monitoring purposes.

As Figure 17 illustrates, the production tool communicates with various IP Studio services and APIs in order to discover available sources, receive metering data, and to control the playback of pre-recorded audio clips.

The production tool also uses the UMCP (Universal Media Composition Protocol) API to build the object-based production and to stream and store the representative metadata. For the Orpheus project, this metadata stream was received by an IP Studio rendering processor for in-studio monitoring and also translated to a serialised version of ADM (Audio Definition Model) [[8]] for transmission to consumers.

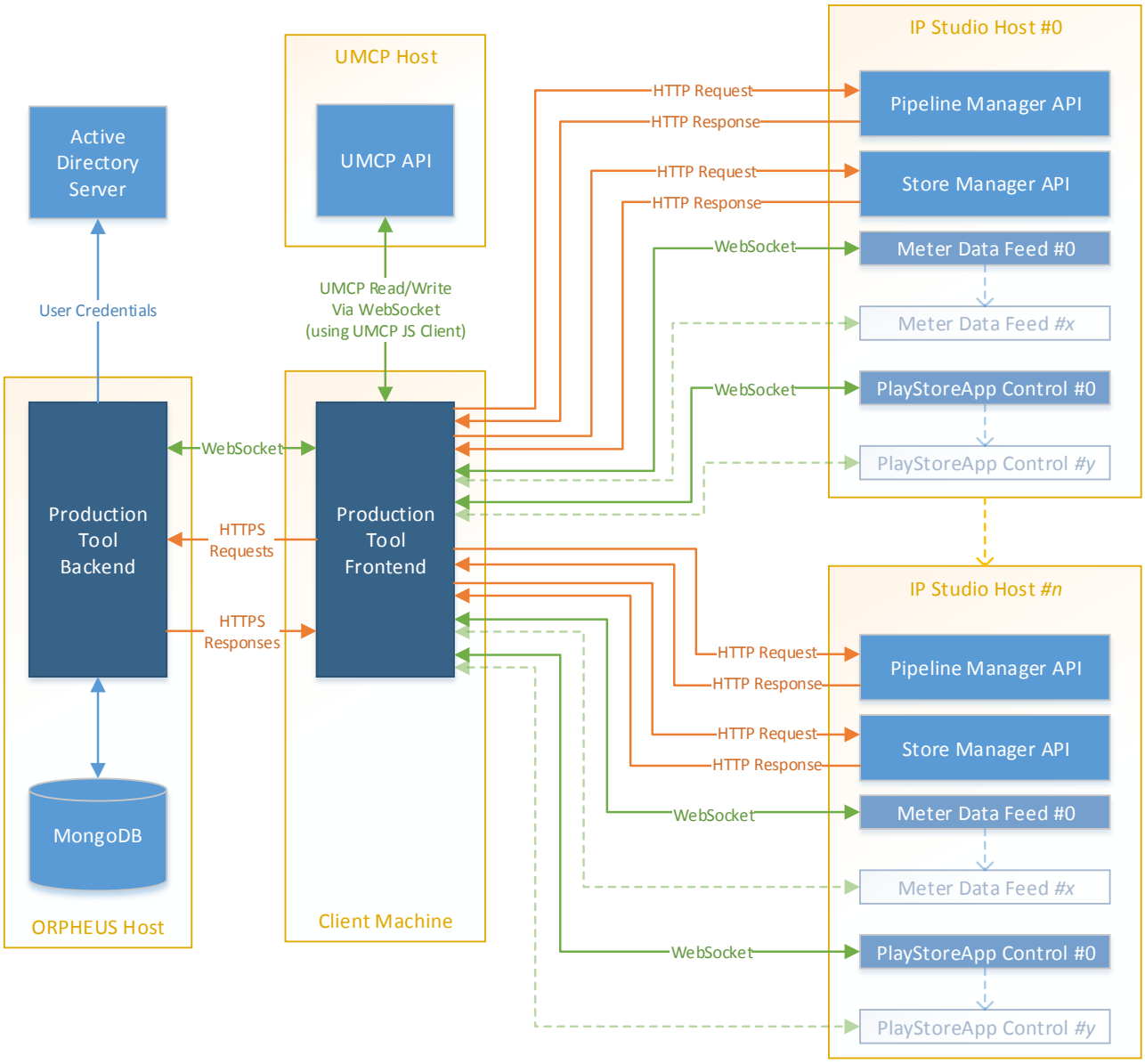


Figure 17: System Architecture

It could be argued that the backend should communicate with these hosts, thus abstracting the frontend away from other APIs and reducing workload. However, since minimal latency is essential for metering data, playback control, and particularly for UMCP data writes, it was decided that the frontend should communicate with the host machines directly.

3.7 Languages, Frameworks and Libraries

Both the backend and frontend are written in JavaScript. More specifically, the frontend uses ES2016. In both cases, a Babel transform plugin is used to support asynchronous functions for cleaner promise-based code (expected to be supported in the currently unpublished ES2017 specification). Node Package Manager (NPM) [[9]] is used for package management and Webpack [[10]] is used for module bundling and running the necessary transpilers and plugins.

3.7.1 Backend

The backend runs on Node JS and integrates with MongoDB. It also has a number of additional dependencies;

- "*activedirectory*" authenticates user credentials (BBC login data) with Microsoft Active Directory.
- "*express*" provides web server functionality, "*express-session*" tracks client sessions, and "*express-ws*" provides a WebSocket server.
- "*mongodb*" provides an API to a MongoDB database to store studio, project and user data.
- "*orpheus-ws-mux*" [[11]] was developed in BBC R&D to combine NMOS event grains received via multiple WebSockets from various sources and flows into one new flow on one single WebSocket endpoint.

3.7.2 Frontend

The frontend runs in the browser. It was developed for and tested in Chrome 60. Other browsers may be supported but are untested. The user interface is built using React [[12]] to manage dynamic elements within the web application. Styling is provided by the Bootstrap framework [[13]] using the Spacelab theme [[14]]. Among the other dependencies are;

- "*cytoscape*" is a JavaScript implementation of the popular Cytoscape graphing tool [[15]]. This is used in the *Routing Map* mode, discussed in section 3.3.2. "*cytoscape-dagre*" produces the graph layout – a hierarchical, tree-based, directed, acyclic graph.
- "*rc-slider*" is used for the fader controls in each channel strip.
- "*react-color*" provides the colour palette used in the *Fader Properties* sidebar, discussed in section 3.3.3.
- "*react-copy-to-clipboard*" provides 'copy' functionality for text data.
- "*uuid*" generates Universally Unique IDentifiers used frequently with UMCP.
- "*tai-timestamp-js*" [[16]] is a JavaScript class originally written by Matt Firth and Matthew Shotton at BBC R&D. It is used for handling and converting NMOS (Networked Media Open Specification [[17]]) timestamps efficiently to nanosecond accuracy without introducing floating-point precision error. These NMOS timestamps are used by UMCP and throughout IP Studio.
- A prebuilt version of "*umcp-client*" [[18]] at version 1.5.1 is included in the source code directory of the interface. This provides a developer-friendly interface to the UMCP API [[19]] in JavaScript.
- A modified version of *meterws.js* (as used in the IP Studio Audio Meter Bridge web application) provides visualisations of metering data generated by IP Studio True-Peak Meter processors.

4 ADMix tools suite

The ADMix tool suite of IRCAM has been already introduced in deliverable D3.5. In the following sections a more detailed description is provided together with current limitations. It is largely inspired from a paper written for the ICSA (International Conference on Spatial Audio) that took place in Graz in September 2017. Originally developed for experimenting with incorporating reverb into an object-based production workflow, the ADMix tool suite can be used in a more general context to create, load and play back audio files containing ADM metadata. In its current state, only a subset of the ADM specifications is supported, but many common use cases are already covered.

The ADMix tool suite is composed of four stand-alone components: Player, Renderer, Recorder and ExtractXML. They are available for MacOS and MS Windows and can be freely downloaded from Ircam forumnet (http://forumnet.ircam.fr/fr/produit/spat/admix-en).

The ADMix Player and ADMix Recorder can communicate with other programs via Open Sound Control (OSC) [21]. The ADMix tools use an ad-hoc OSC syntax, which facilitates the communication with other OSC-aware audio devices or software application. The ToscA plugin, developed before the ORPHEUS project, can serve for instance as an ancillary tool to connect a Digital Audio Workstation (DAW) to the ADMix Recorder. The ToscA plugin is also freely available from Ircam forumnet (http://forumnet.ircam.fr/fr/produit/spat/tosca-en).

4.1 ADMix Player and ADMix Renderer

The ADMix Player can load WAV/BWF/BW64 files and read ADM metadata and channel assignments from their respective chunks in the file. Upon loading a file, it performs a number of "sanity checks" on the ADM metadata. This is helpful to check whether a given ADM file is compliant. The ADMix Player plays back the audio content of the different audio objects contained in the file as separate tracks and produces synchronized control messages based on the ADM metadata obtained from the file.

The ADMix Renderer embeds an ADMix Player and exploits the audio object tracks and associated metadata to calculate appropriate signals for real-time reproduction over headphones or loudspeaker setups. The main user interface of the ADMix Renderer is shown in Figure 18. Summary information about the content of the currently open file (number of objects, number of tracks, duration, etc.) is displayed in the list on the right 1. Controls for playback (start/stop/resume/loop/seek) are located on the left 2. A set of meters displays the level of the different audio object tracks (outputs from the embedded ADMix Player) 3 whereas another set of meters displays the level of the rendered output channels 4. On the lower right, the rendering settings can be chosen 5.

The ‘view scene’ button opens a window displaying the current position of the different audio objects and allows moving them interactively as long as there is no position metadata associated to the object at current time (Figure 19). The ‘mute/solo’ button opens a window showing the different audio objects and associated components (packs, channels). When clicking on a given object (here the ‘Dialogue’ object), all linked components (content, pack, channels…) are highlighted (Figure 19).

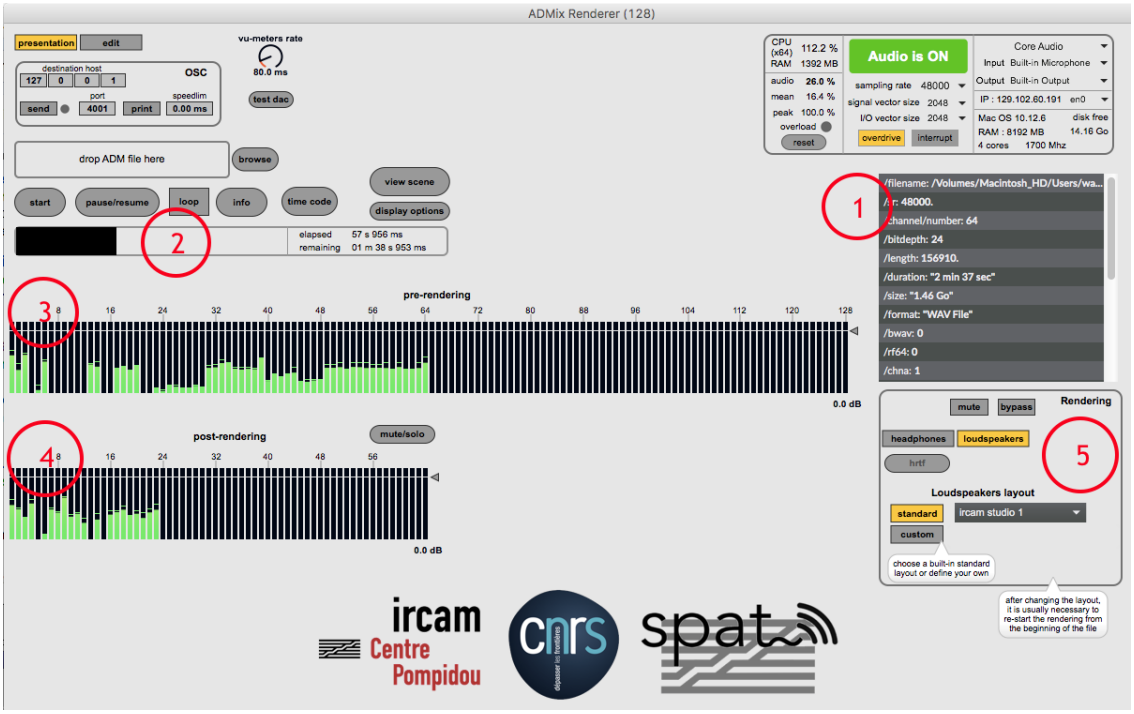


Figure 18: Main user interface of the ADMix Renderer

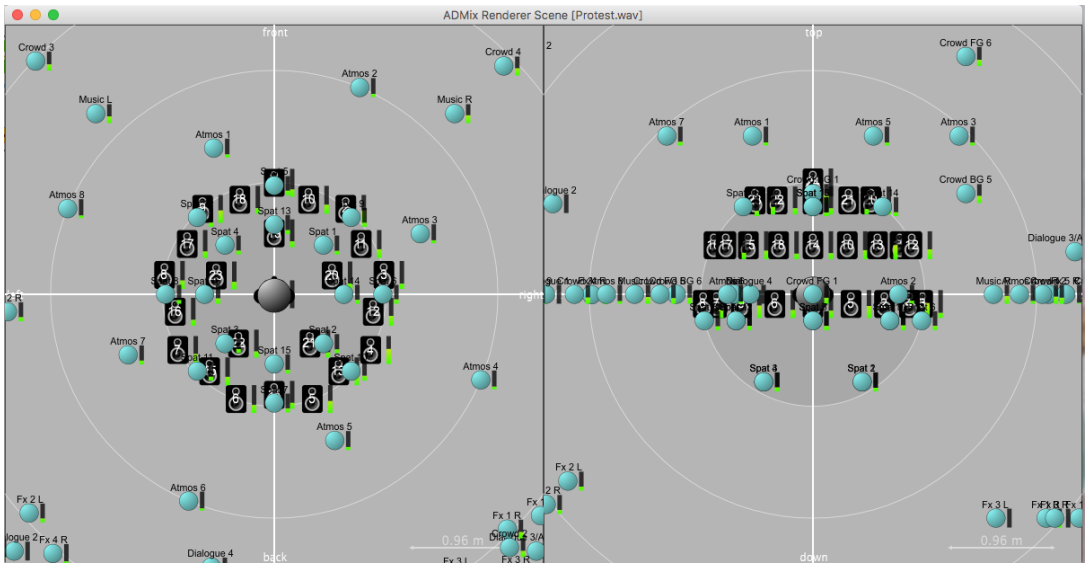


Figure 19: Scene viewer of the ADMix Renderer displaying the different objects together with the loudspeakers of the rendering setup

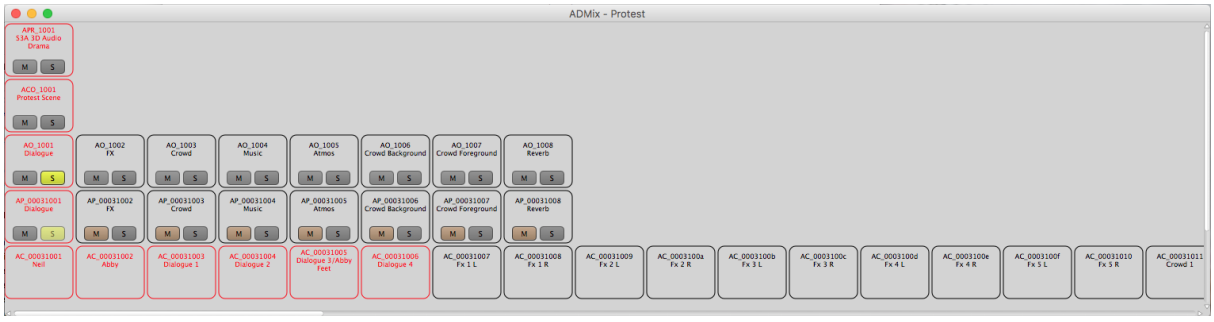


Figure 20: Display of the programme content of the ADM file with Mute/Solo interaction on the objects

For the rendering of “object” type objects, vector based amplitude panning (VBAP) is used [20]. The loudspeaker setup can be chosen from a predefined set of common loudspeaker setups (5.0/5.1, 6.0/6.1 … 22.0/22.2, 4.7.0) or by manually specifying custom loudspeaker positions (Figure 21). Up to 64 loudspeakers are supported. For binaural rendering, Head-Related Transfer Functions (HRTFs) can be selected from HRTF datasets using the SOFA/AES69 format [22]. These HRTF can be accessed either from datasets stored locally or through public web databases.

The rendering of “HOA” type objects is also implemented. Supported encoded HOA normalisation and channel number conventions are N3D-ACN and SN3D-ACN. The HOA decoder follows the Energy Preserving approach [23].

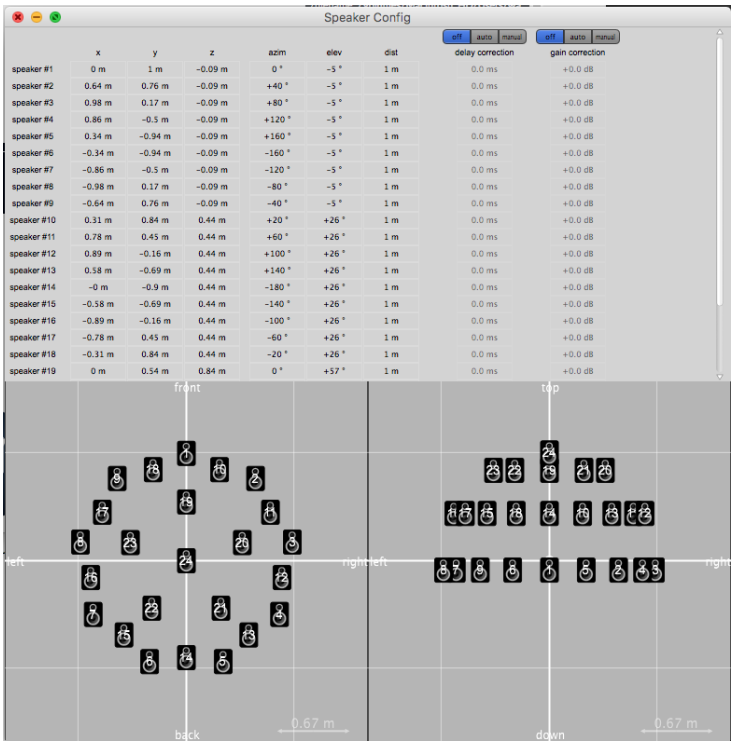


Figure 21: User interface allowing to edit and visualize the position of the loudspeaker setup

4.1.1 Current limitations

The current version of the ADMix Player and Renderer is limited to single Programme and single Content ADM files, and only supports basic object attributes such as position and gain. The renderer does not yet implement position interpolation, i.e. the objects jump immediately to the position specified in the current AudioBlockFormat. Ongoing work is dedicated to the rendering of the “diffuse” attribute, support for multiple Programme.

4.2 ADMix ExtractXML

Given a sound file with ADM metadata, *ADMix ExtractXML* exports an XML file containing only the ADM metadata (i.e. the contents of the <axml> chunk). The content of the <chna> chunk is also stored in a separate file. Like the *ADMix Player*, this tool runs several "sanity checks" on the metadata to make sure it complies with the ADM specification. *ADMix ExtractXML* also creates structural representations of the XML elements using the DOT graph description language [24]. An example of such graph visualized with the Graph Visualization Software [25] is depicted in Figure 22.

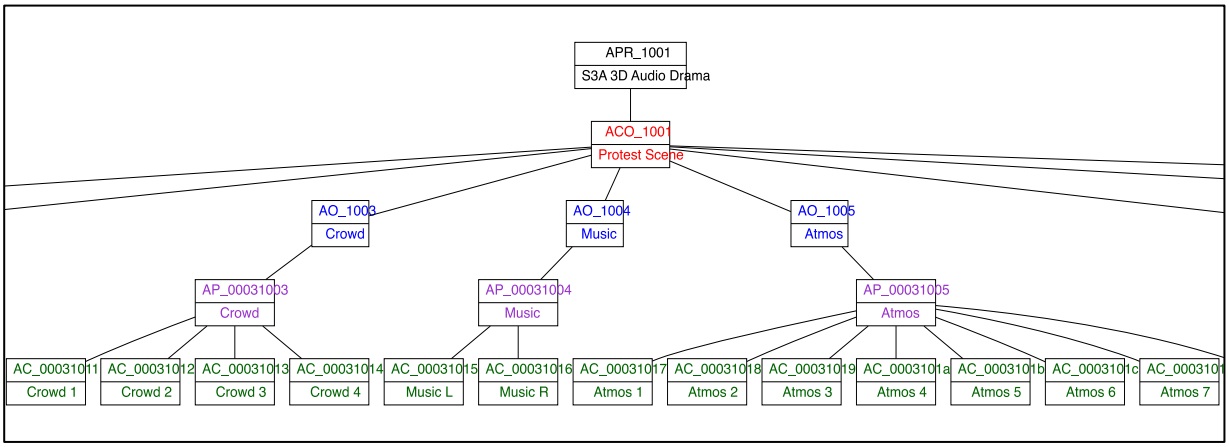


Figure 22: Partial view of the XML metadata structure of and ADM file. The IDs shown in the boxes are used internally to create connections between ADM elements (objects, packs, channels…)

4.3 ADMix Recorder

The *ADMix Recorder* is used to create, configure and record an ADM file. It can record up to 128 audio channels into a BWF/BW64 file. For each of these channels, it can also record automation data that is stored as ADM metadata in the same file. The *ADMix Recorder* has a built-in renderer that can be used for monitoring.

Figure 23 shows the main window of the ADM Recorder. The monitoring settings 1 in the bottom right of the window are the same as for the *ADMix Renderer* described above. In the top left of the main window, host and port settings 3 can be configured for receiving and sending OSC messages that can transport ADM metadata between applications or between computers. This can be used typically to exchange position or gain automation data with a Digital Audio Workstation (see section 4.4). Messages that are received via this interface are stored as XML metadata in the recorded ADM. The "configure" button 2 opens a window for editing the ADM metadata.

On the top of this configuration window (Figure 24), the programme and content name can be specified 1. Below that, a list of "audio packs" 2 and 3 can be set up. Each pack can contain one or more "channels". Each channel has a unique channel number that can be used together with the routing matrix on the left side of the window 4. Supported ADM object types are “object”, “HOA”, “Direct Speakers” and “binaural”. For the "DirectSpeakers" type, one of a set of pre-defined channel-based setups (e.g. stereo, 5.1, 22.2, 4.7.0 …) can be selected which populates automatically the channels appropriately. For the “HO” type, the user is invited to specify the order (up to order 10) and the normalization/channel order convention (SN3D-ACN, N3D-ACN). A "Binaural" pack automatically contains two channels, one for the left and one for the right ear.

N.B. The current state of the ADMix Recorder only supports “flat structures”, i.e. no nested objects. It does not yet support multiple programmes, multiple contents or ‘complementary objects’ (used for instance to describe mutually exclusive languages).

Once the packs and channels are set up as desired, the “arm” button can be used to enable recording. This will open a scene viewer window (Figure 25), which allows configuring the initial positions of all audio objects. The scene window also allows moving objects with the mouse during recording, which will record the object trajectories in the resulting ADM file.

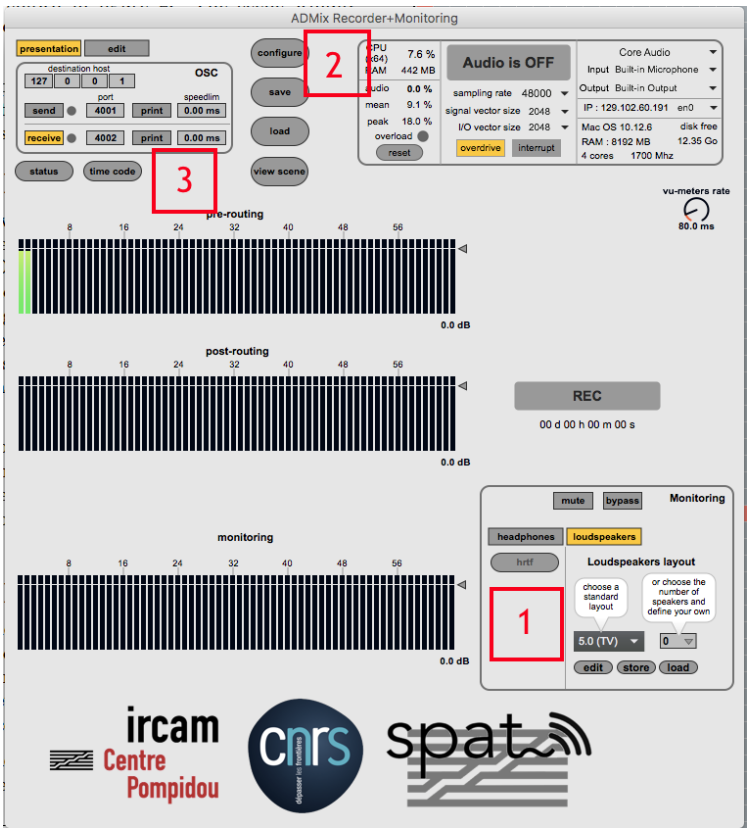


Figure 23: Main window of the ADMix Recorder

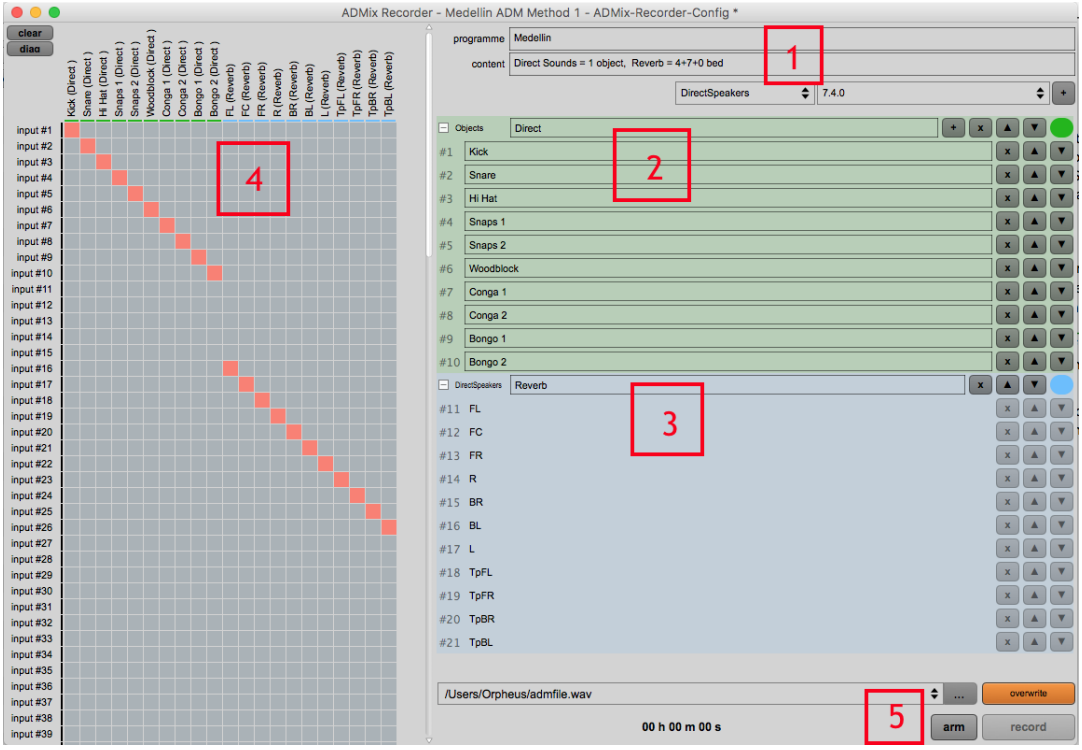


Figure 24: Metadata editor and routing matrix of the ADMix Recorder

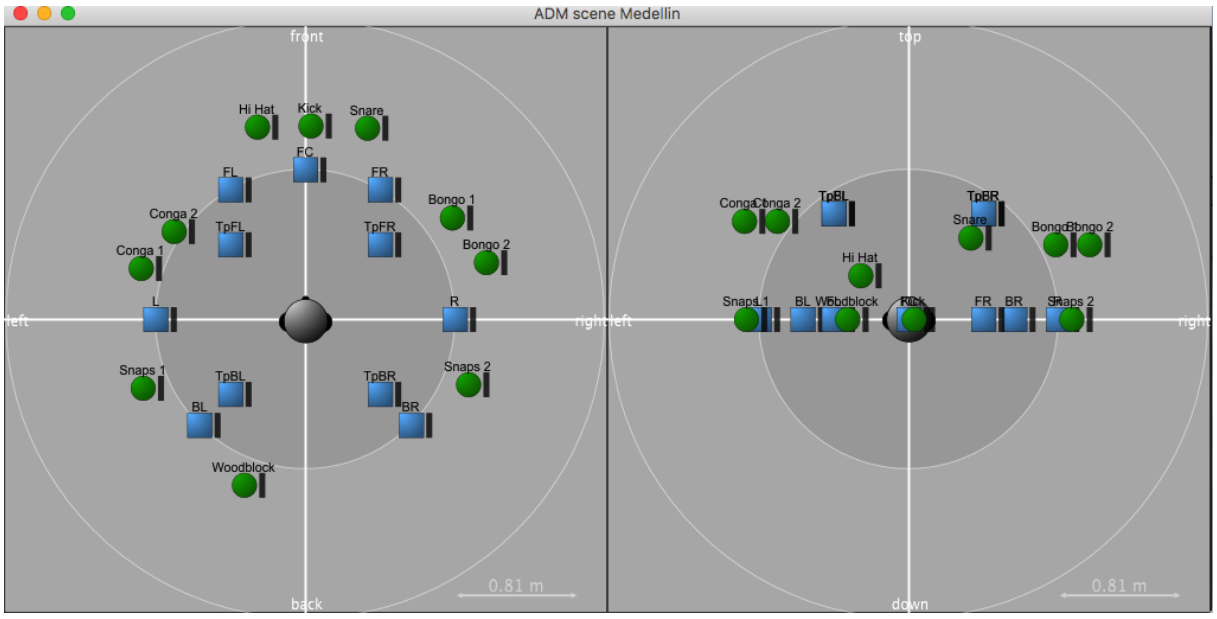


Figure 25: Scene viewer of the ADMix Recorder used to set the initial position of the objects and to create / monitor their automation

4.4 DAW Automation with ToscA

In practice, it will be more convenient to connect the *ADMix Recorder* to a Digital Audio Workstation (DAW), which can record automation data for multiple objects (possibly one after another). The DAW can then play back all automation tracks at once, which in turn can be recorded by the *ADMix Recorder*. A track with an LTC time code signal can be used to synchronize start and stop times between the DAW and the ADM Recorder.

ToscA [21] is a plugin for Digital Audio Workstations (DAWs) that allows the recording of automation tracks for arbitrary parameters that can be sent and received as OSC messages via a network interface. Primarily developed for the remote control of spatial audio processors, this tool also turns out to be very handy for "linking" the *ADMix tools* with a DAW. It can be used together with the *ADMix Recorder* to record movements and changing gain values as automation tracks in a DAW.

An instance of the ToscA plugin has to be inserted into any track that is supposed to contain automation data, see figure [26]. Each ToscA instance can then be configured with a channel ID that is used to associate tracks in the DAW with channels defined in the ADM Recorder. The *ADMix Recorder* will in turn store the recorded automation data in the ADM file.

The audio outputs of the DAW tracks can be provided to the ADM Recorder either via hardware connections or via virtual sound drivers.

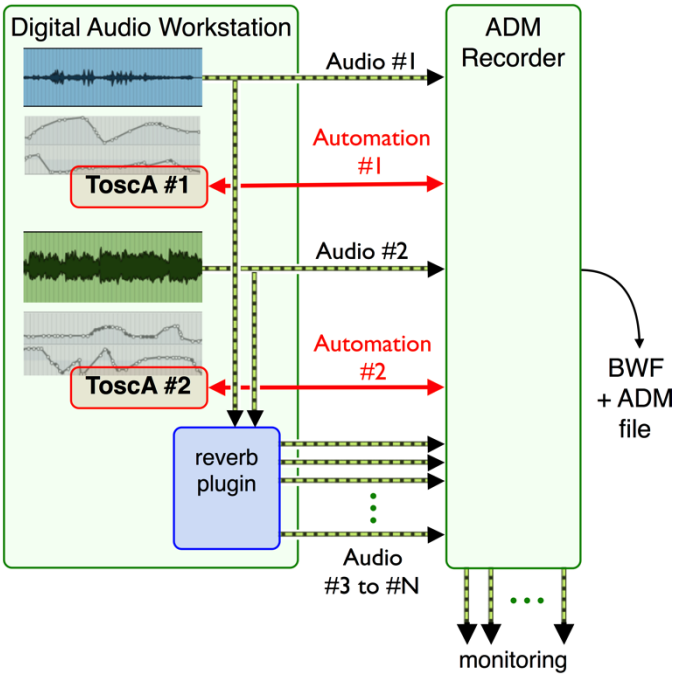


Figure 26: Using the ToscA plugin to connect a DAW with the ADMix Recorder

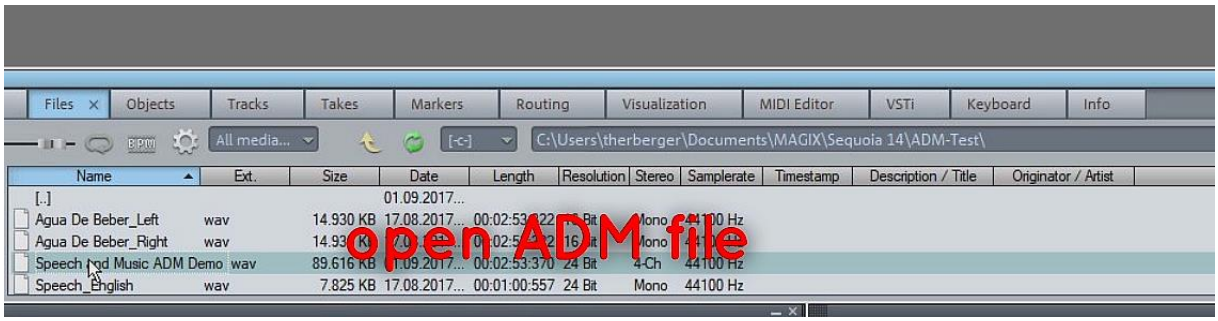
Appendix A Description of Sequoia Workflows

This appendix provides a quick introduction to the various new workflows, which were introduced to deal with ADM files and 3D audio recordings in Sequoia. The explanations are also available in the form of a tutorial video[28].

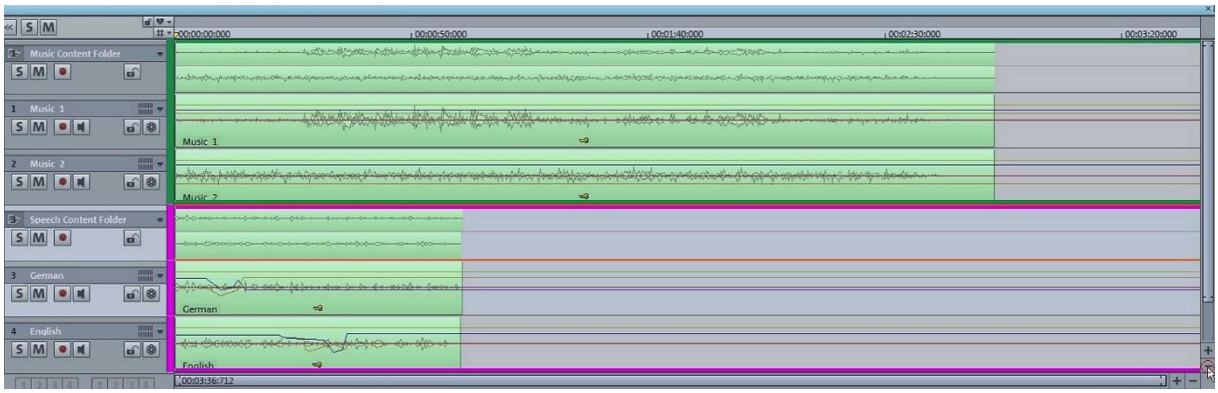
A.1 Import and export of ADM files

To import an existing ADM file, edit its content and export it as a new file, a Sequoia user needs to follow these steps:

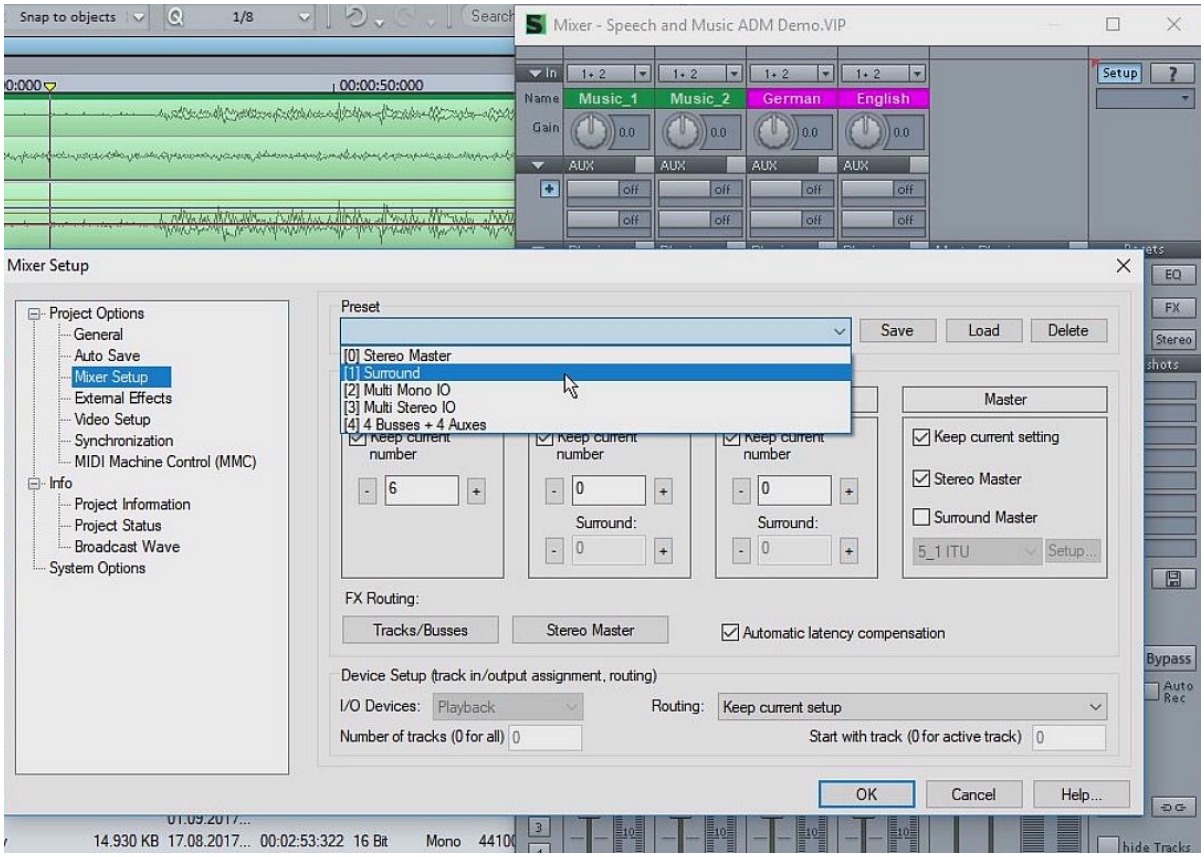
(1) Open ADM file using menu “File > Open” or “Files” Tab in the manager bar



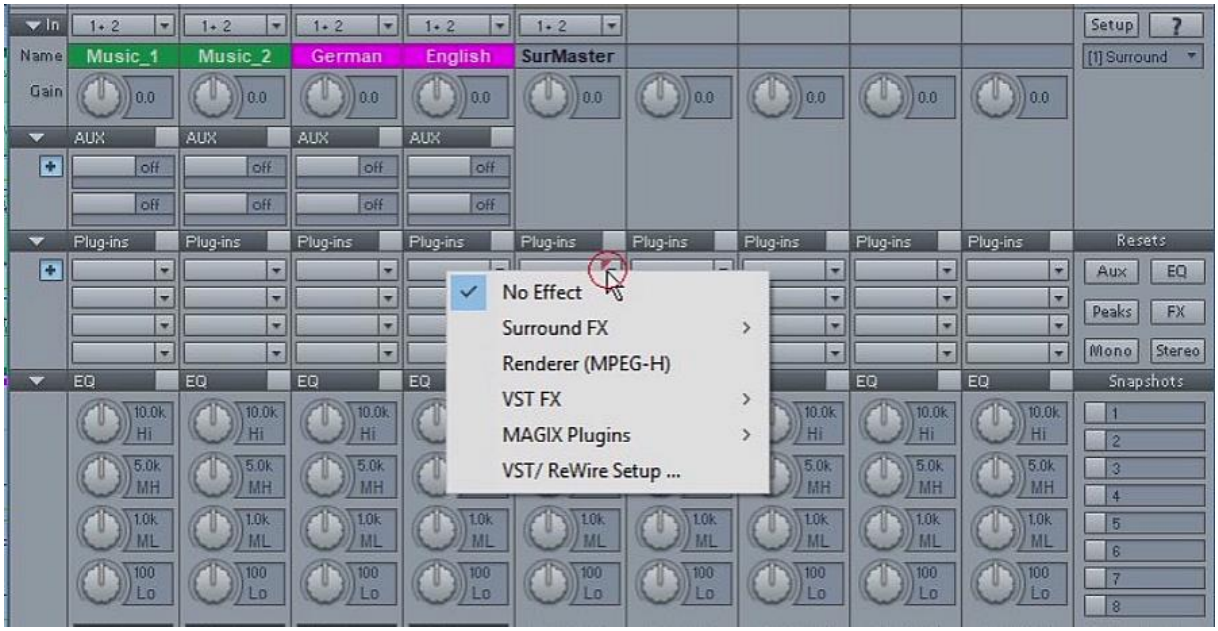
(2) Zoom vertically to view all tracks of the ADM file



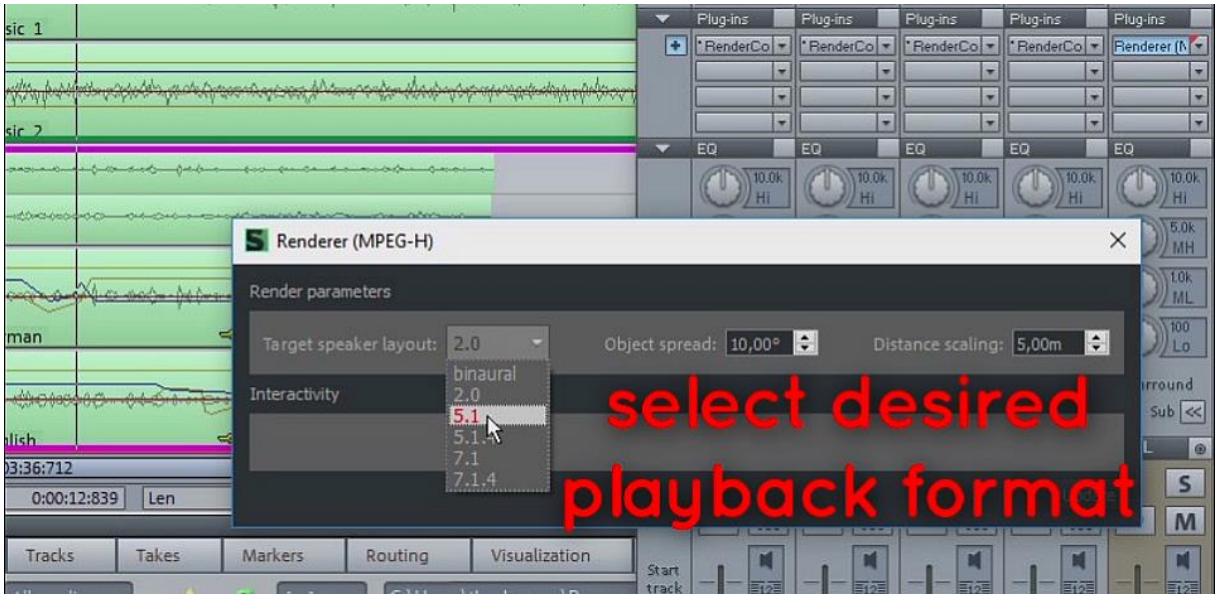
(3) Open mixer using key “m” and set mixer to surround mode



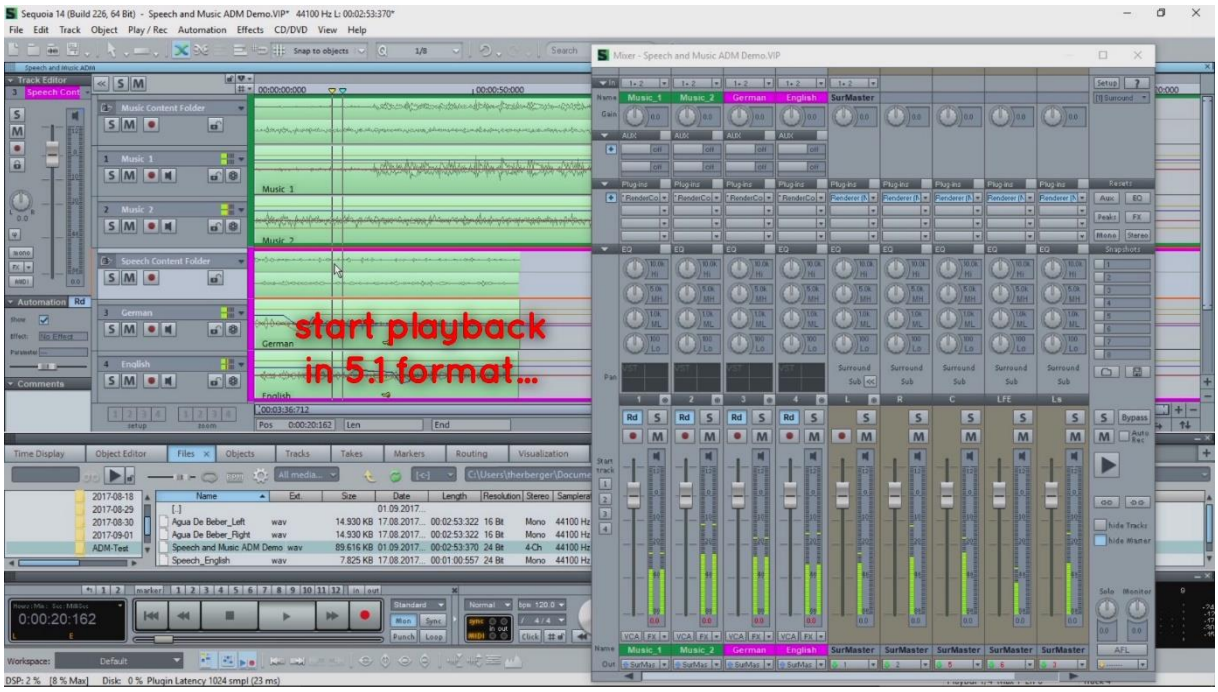
(4) Insert renderer plugin on the surround master bus



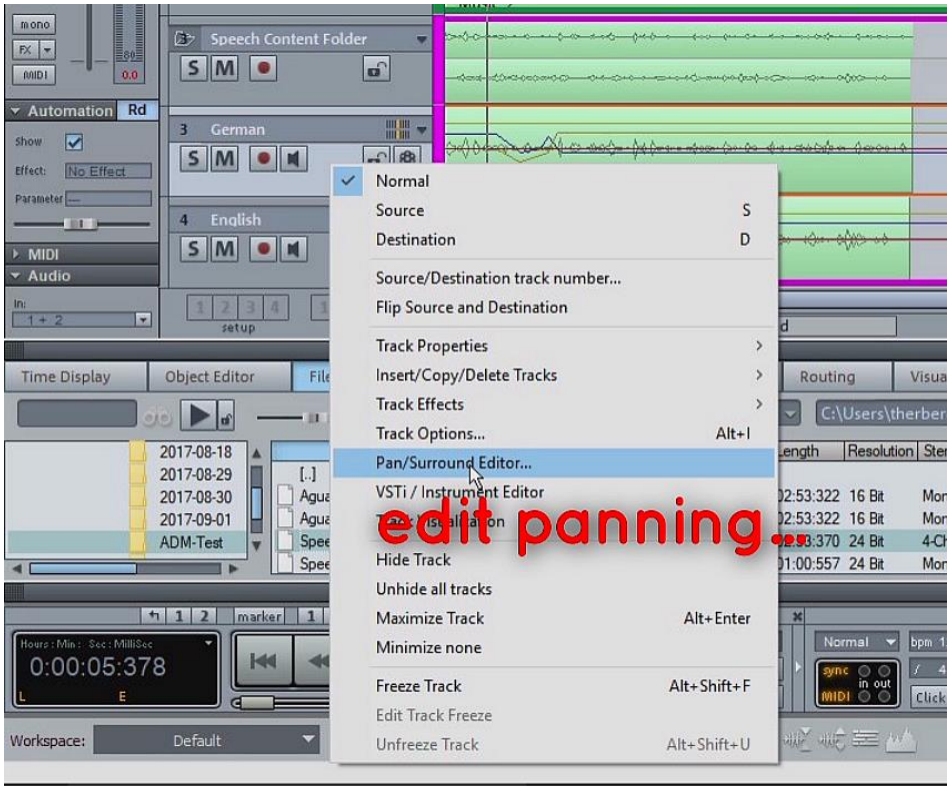
(5) Select target format in renderer plugin (e.g. 5.1 surround, stereo or binaural…)



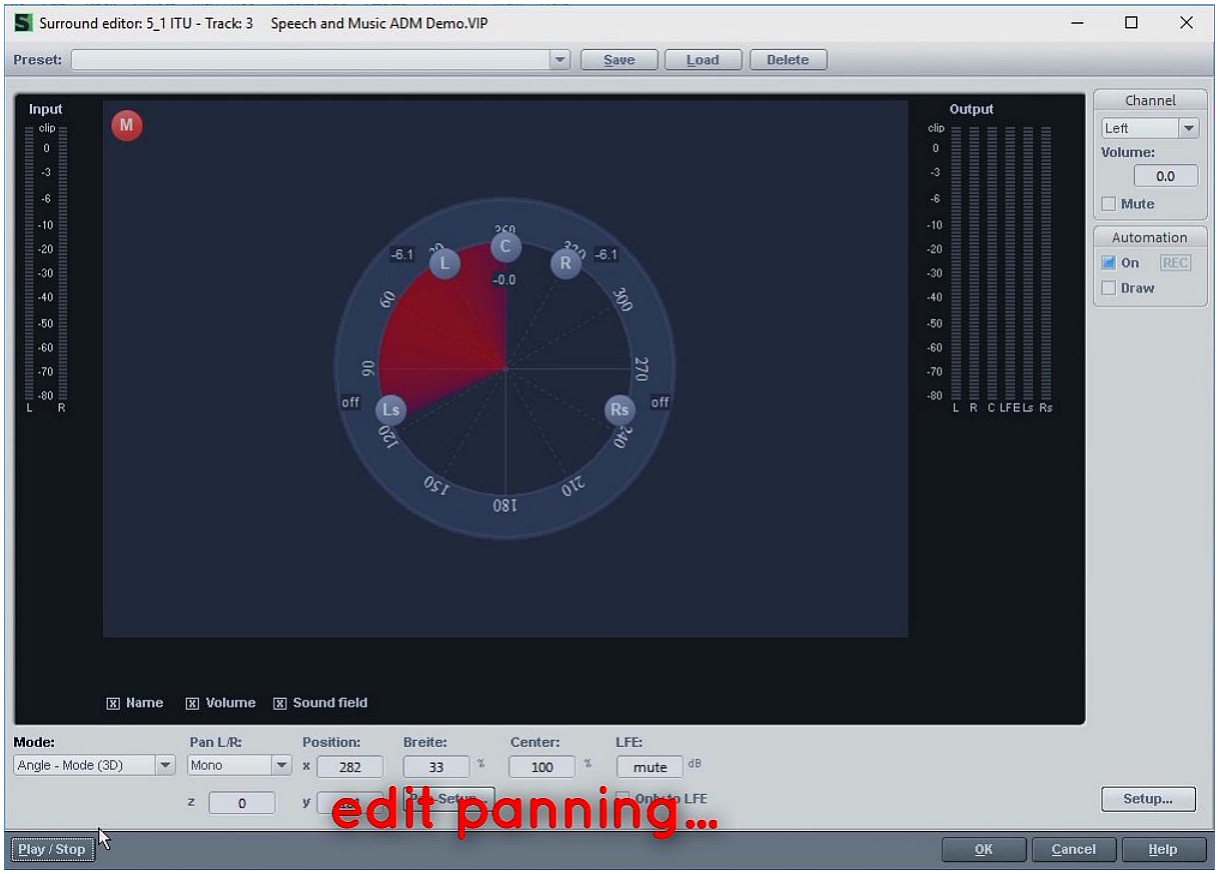
(6) Start playback using space key or play button in transport control



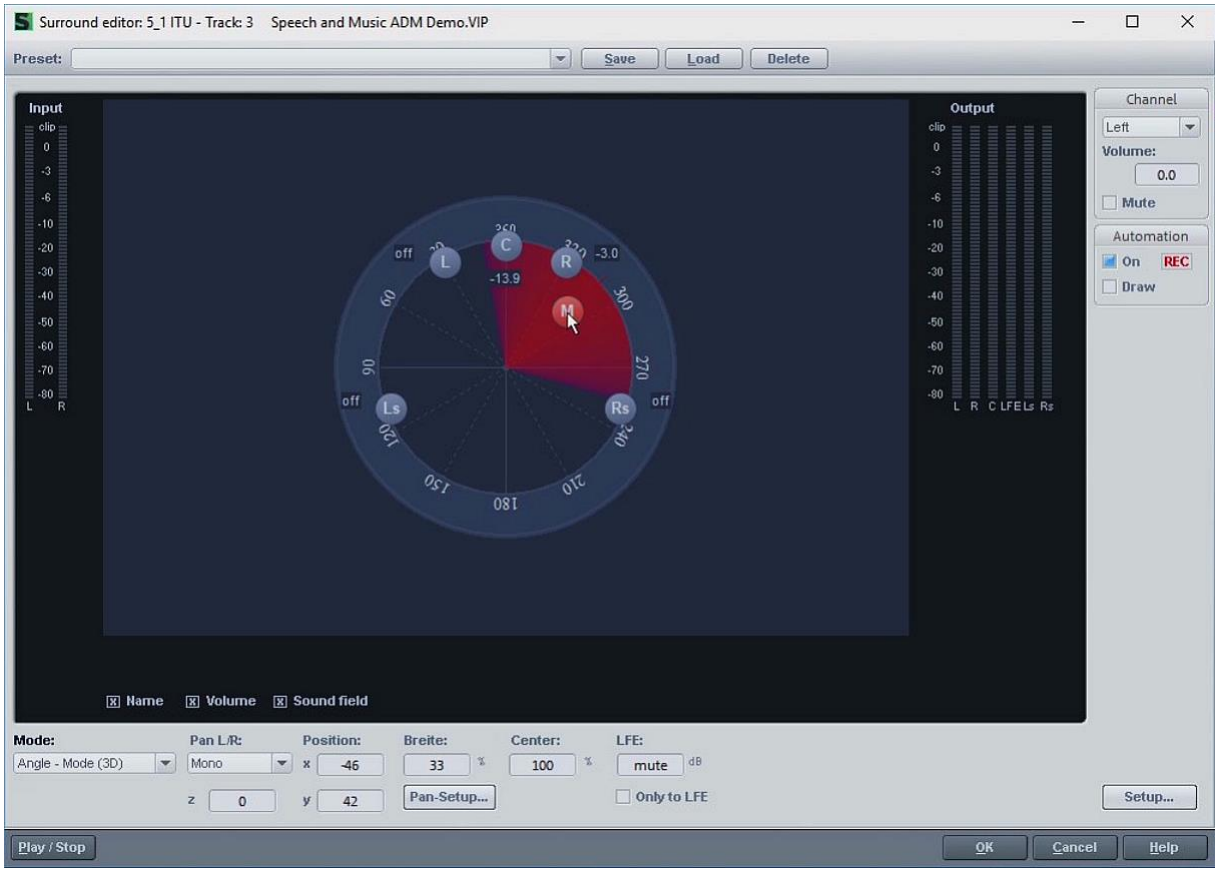
(7) Open panning dialog using track pop up menu “Pan / Surround Editor”



(8) Edit static 3D panning in surround editor dialog

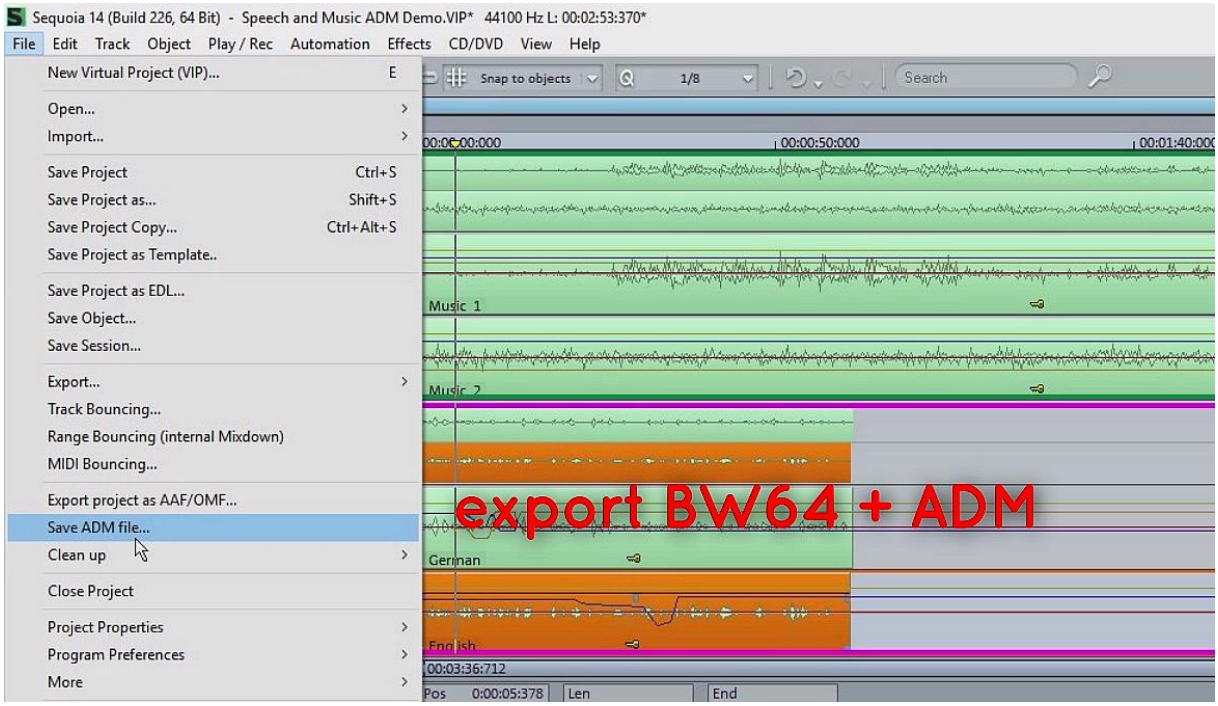


(9) Edit dynamic (automated) panning using the “Automation REC” button

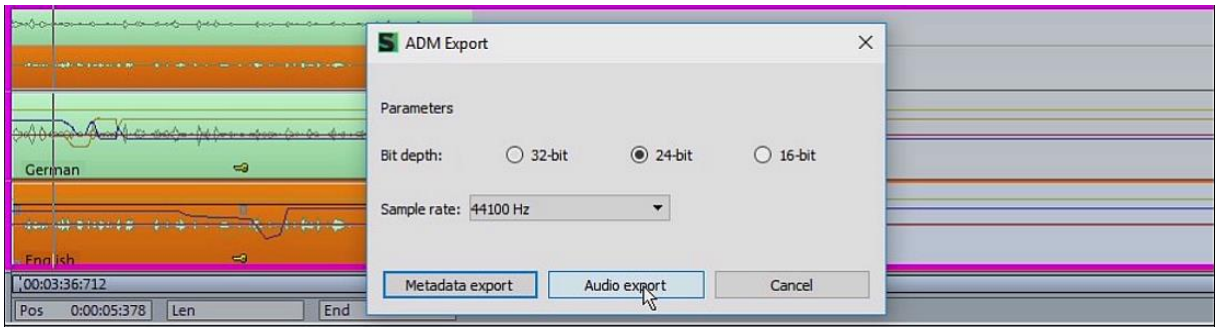


(10) Make any other edits as described in the Sequoia Manual

(11) Export the project as an ADM file using menu “File > Save ADM file…”



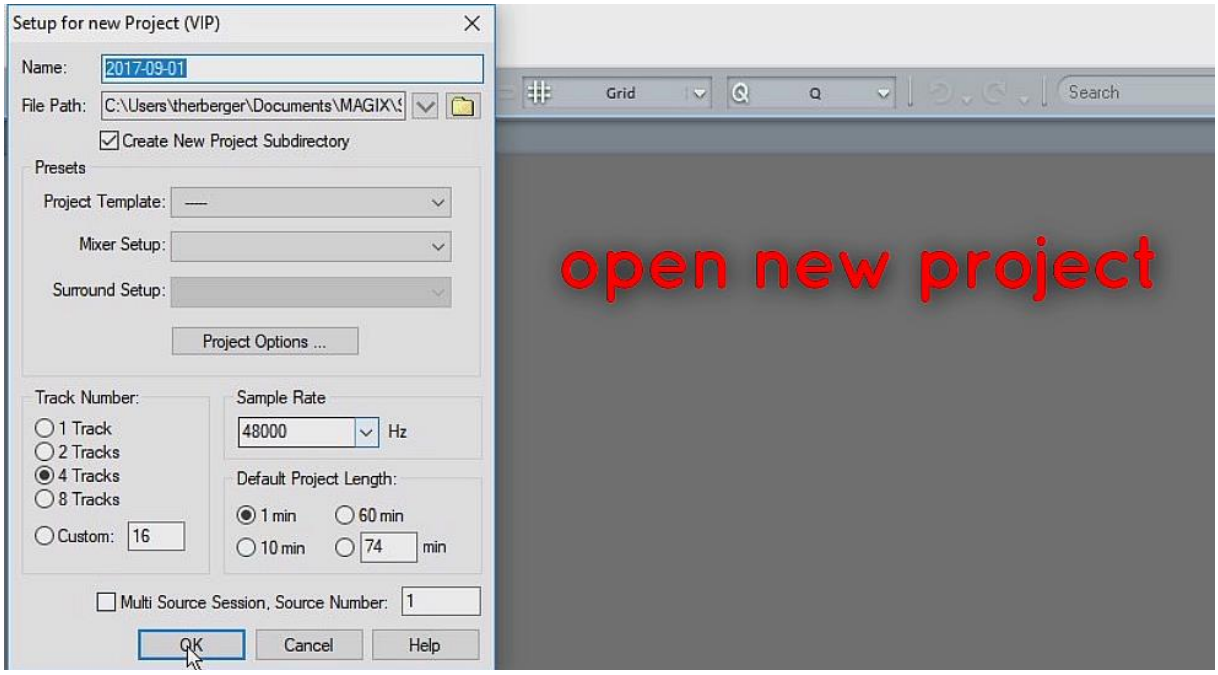
(12) Select desired sampling format and click on “Audio export” to save audio and ADM metadata to a new BW64 file



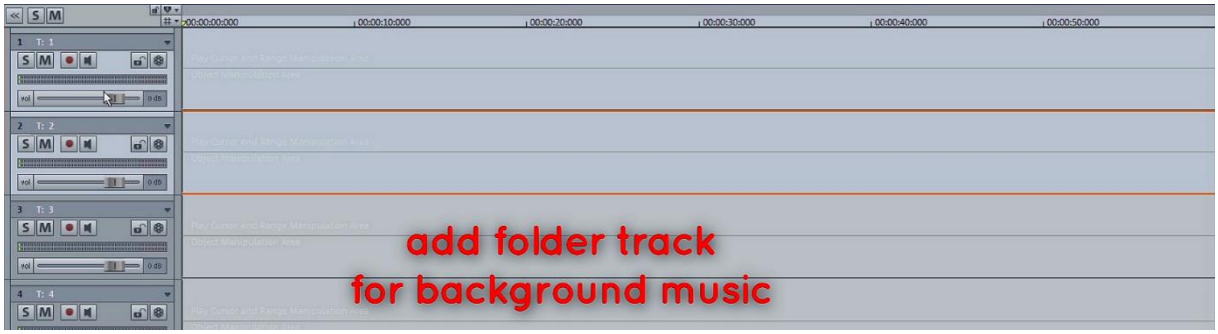
A.2 Creating ADM files from scratch

To create an ADM file from scratch and edit metadata for interactive language selection and foreground / background grouping a Sequoia user needs to follow these steps:

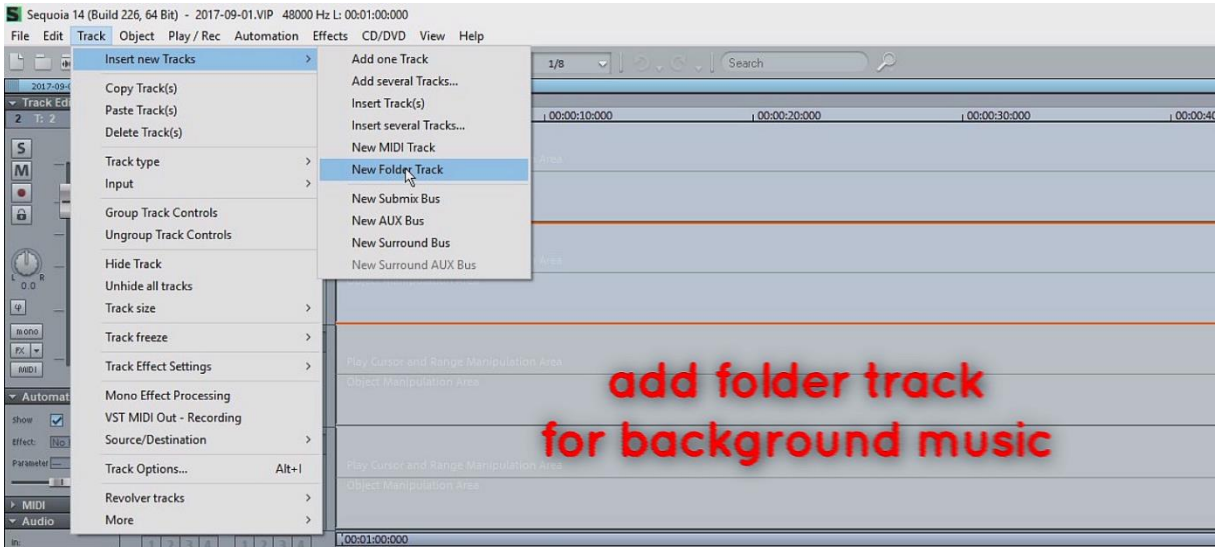
(1) Open new project with the desired track number and sample rate (can be changed later at any time)



(2) Select the desired number of tracks (by Ctrl-Click) for the first content folder, e.g. for background music



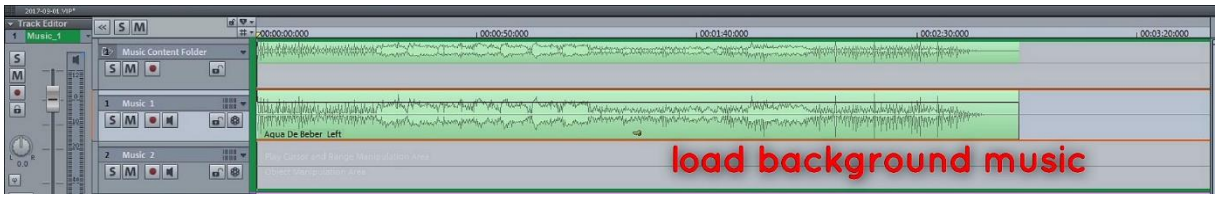
(3) Add folder track for first audio content using menu “Track > Insert new Tracks > New Folder Track” (currently selected tracks are automatically added to the new folder track)



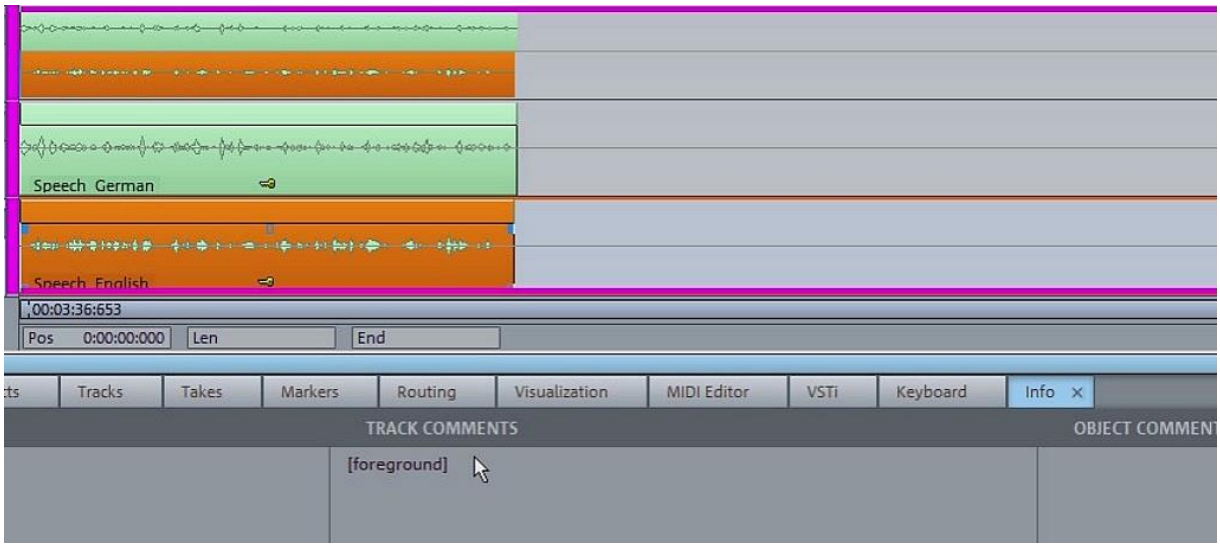
(4) Rename tracks of multi-channel objects with using the suffixes _1, _2 etc.

(5) Repeat these steps for new content folders, e.g. for speech tracks in multiple languages

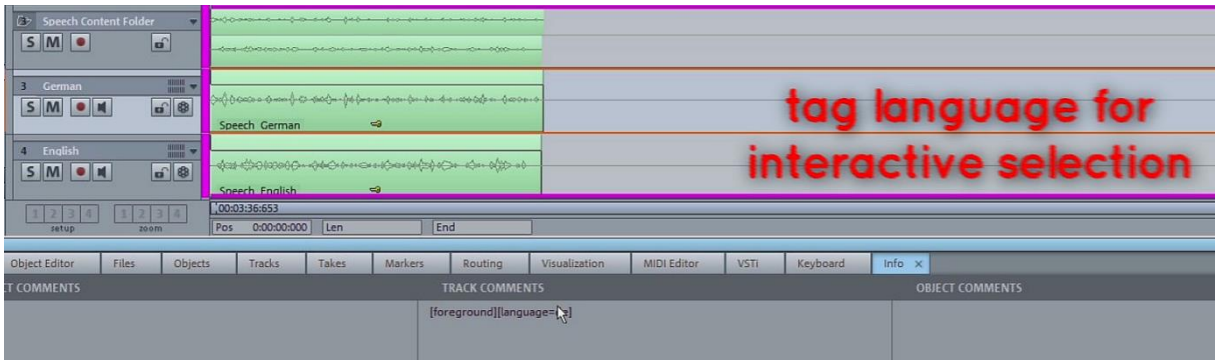
(6) Load music and speech audio files into the prepared tracks using “Files” tab in the manager section of menu “File > Import”



(6) Tag foreground (speech) tracks with the keyword [foreground] in the track comment section of the “Info” manager tab

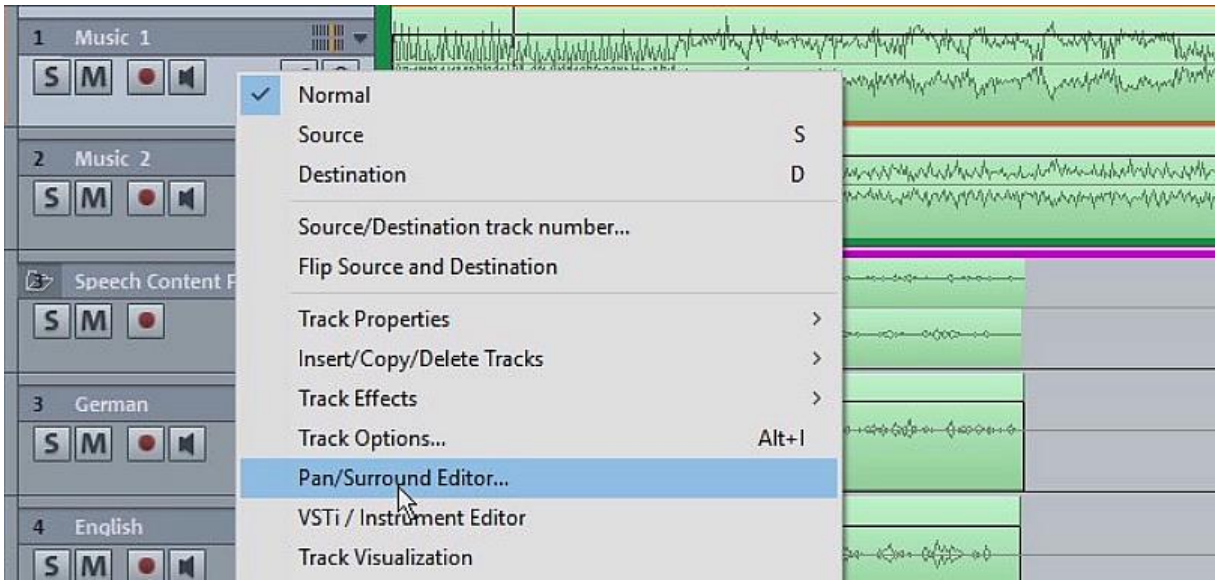


(7) Tag language per track in the same way using the keyword [language=de/en/fr]



(8) Setup renderer plugin as described in section A.1 (steps 3 – 6)

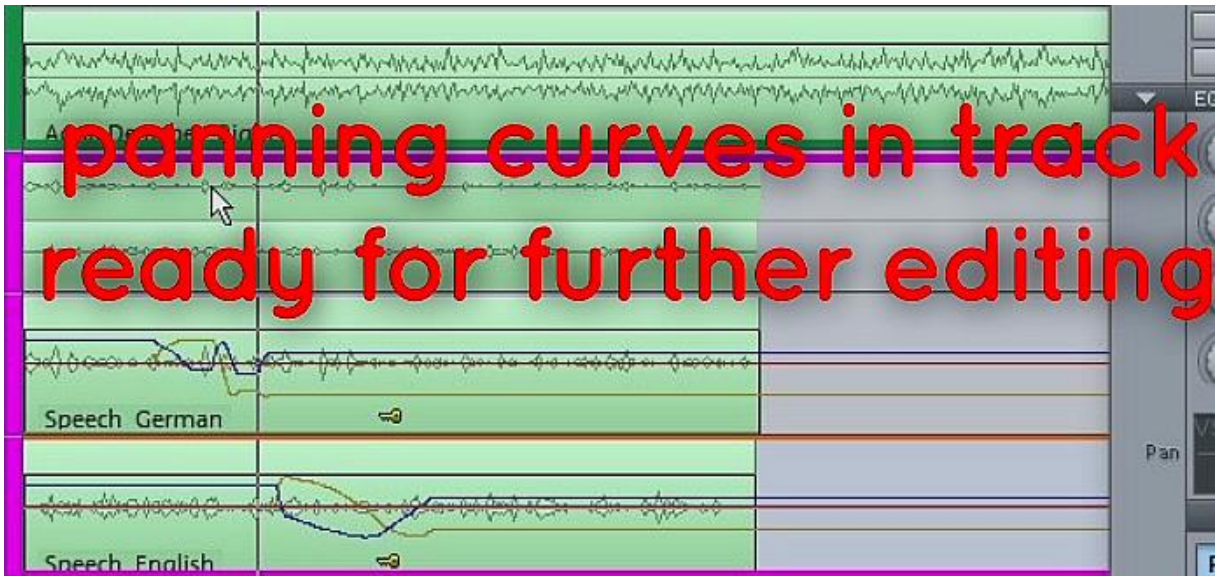
(9) Open panning dialog using the track menu “Pan/Surround Editor”



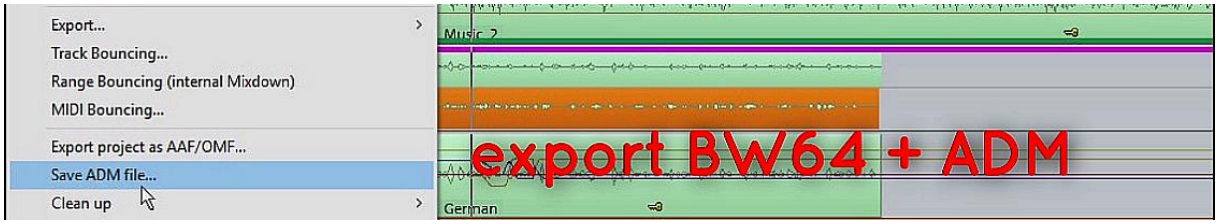
(10) Switch to “3D” panning mode , “Mono” tracks and drag panning position to the desired location

(11) Add dynamic panning using the “Automation Record” section. Draw panning automation during playback.

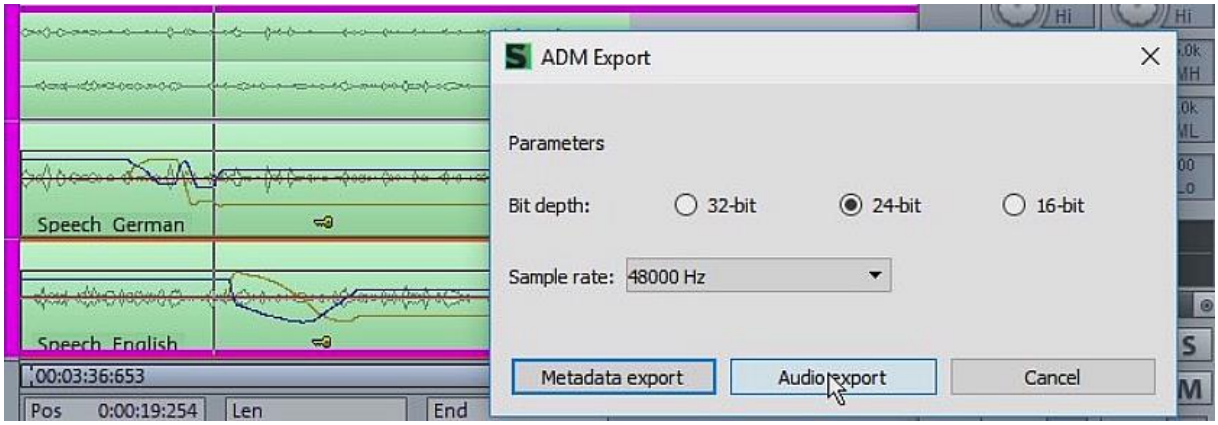
(12) Edit panning curves in the tracks manually if needed



(13) Export the project via “File > Save ADM file”



(14) Set desired sampling format and click “Audio export” to save project into BW64 file with ADM metadata



A.3 Recording to ADM

(1) Create a project with 32 tracks to directly record or import existing “Eigenmike” microphone signals

(2) Set the project to surround from the mixer interface and select the 32.0 preset combined with the routing option “Assign tracks to surround channels” which will automatically route each track to one sub-bus

(3) Insert b<>com “MicProcessor” plugin on the surround master bus, which will convert microphone signals to HOA

(4) Insert b<>com “RenderHOA2Spk” plugin on the surround master bus, which will convert HOA signals to your target speaker format

(5) Create a new folder track containing the track count of the selection target speaker format (e.g. 6 tracks for 5.1 speaker format) and pan the tracks to the corresponding positions

(6) Label the folder track following the special naming convention “##RECORDING”

(7) Use menu File > ADM Export to create an ADM file containing the rendered output combined with metadata generated from the “dummy” folder track

References

[1] ORPHEUS, https://orpheus-audio.eu/

[2] Fraunhofer IIS, https://www.iis.fraunhofer.de/en.html

[3] Horizon 2020, European Commission. https://ec.europa.eu/programmes/horizon2020/

[4] Forecaster: our experimental object-based weather forecast, December 2015, Leonard, M. http://www.bbc.co.uk/rd/blog/2015-11-forecaster-our-experimental-object-based-weather-forecast

[5] Atomised News - with BBC R&D, http://bbcnewslabs.co.uk/projects/atomised-news/

[6] orpheus-audio-control-backend, https://github.com/bbc/orpheus-audio-control-backend

[7] orpheus-audio-control-interface, https://github.com/bbc/orpheus-audio-control-interface

[8] Audio Definition Model (ADM): BS.2076-1, June 2017, ITU. https://www.itu.int/rec/R-REC-BS.2076-1-201706-I/en

[9] NPM, https://www.npmjs.com/

[10] Webpack, https://webpack.js.org/

[11] orpheus-ws-mux, https://github.com/bbc/ips-orpheus-ws-mux

[12] React, Facebook. https://facebook.github.io/react/

[13] Bootstrap Framework, http://getbootstrap.com/

[14] Spacelab Theme, Bootswatch, https://bootswatch.com/spacelab/

[15] Cytoscape, http://www.cytoscape.org/

[16] tai-timestamp-js, https://github.com/bbc/tai-timestamp-js

[17] Networked Media Open Specifications (NMOS), https://nmos.tv/

[18] umcp-client, https://github.com/bbc/umcp-client

[19] umcp-production-api, https://github.com/bbc/umcp-production-api

[20] Pulkki, V., Virtual sound source positioning using Vector Base Amplitude Panning, Journal of the Audio Engineering Society, 45(6):456-466, 1997.

[21] Carpentier, T., ToscA: An OSC communication plugin for object-oriented spatialization authoring, In 41st International Computer Music Conference, 2015, https://hal.archives-ouvertes.fr/hal-01247588v1/

[22] Majdak, P., Iwaya, Y., Carpentier, T., Nicol, R., Parmentier, M., Roginska, A., Suzuki, Y., Watanabe, K., Wierstorf, H., Ziegelwanger, H. and Noisternig, M., Spatially Oriented Format for Acoustics: A data exchange format representing Head-Related Transfer Functions, In 134th AES Convention, 2013

[23] Zotter, F., Pomberger, H., Noisternig, M., Energy-Preserving Ambisonic Decoding, Acta Acustica United with Acustica, Vol. 98:37-47, 2012

[24] Gansner, E. and North, S., An open graph visualization system and its applications to software engineering. Software: Practice and Experience, 30(11):1203{1233, September 2000

[25] Graph Vizualization Software, http://www.graphviz.org

[26] Geier, M., Carpentier, T., Noisternig, M., Warusfel, O., Software tools for object-based audio production using the Audio Definition Model, in Proceedings of Int. Conf. on Spatial Audio, Graz, September 2017. http://hal.archives-ouvertes.fr/hal-01574183

[27] EBUcore Metadata Set: Tech 3293, October 2017, EBU. https://tech.ebu.ch/MetadataEbuCore

[28] ADM workflows in Sequoia, MAGIX. https://youtu.be/lmUbNZYQmSw