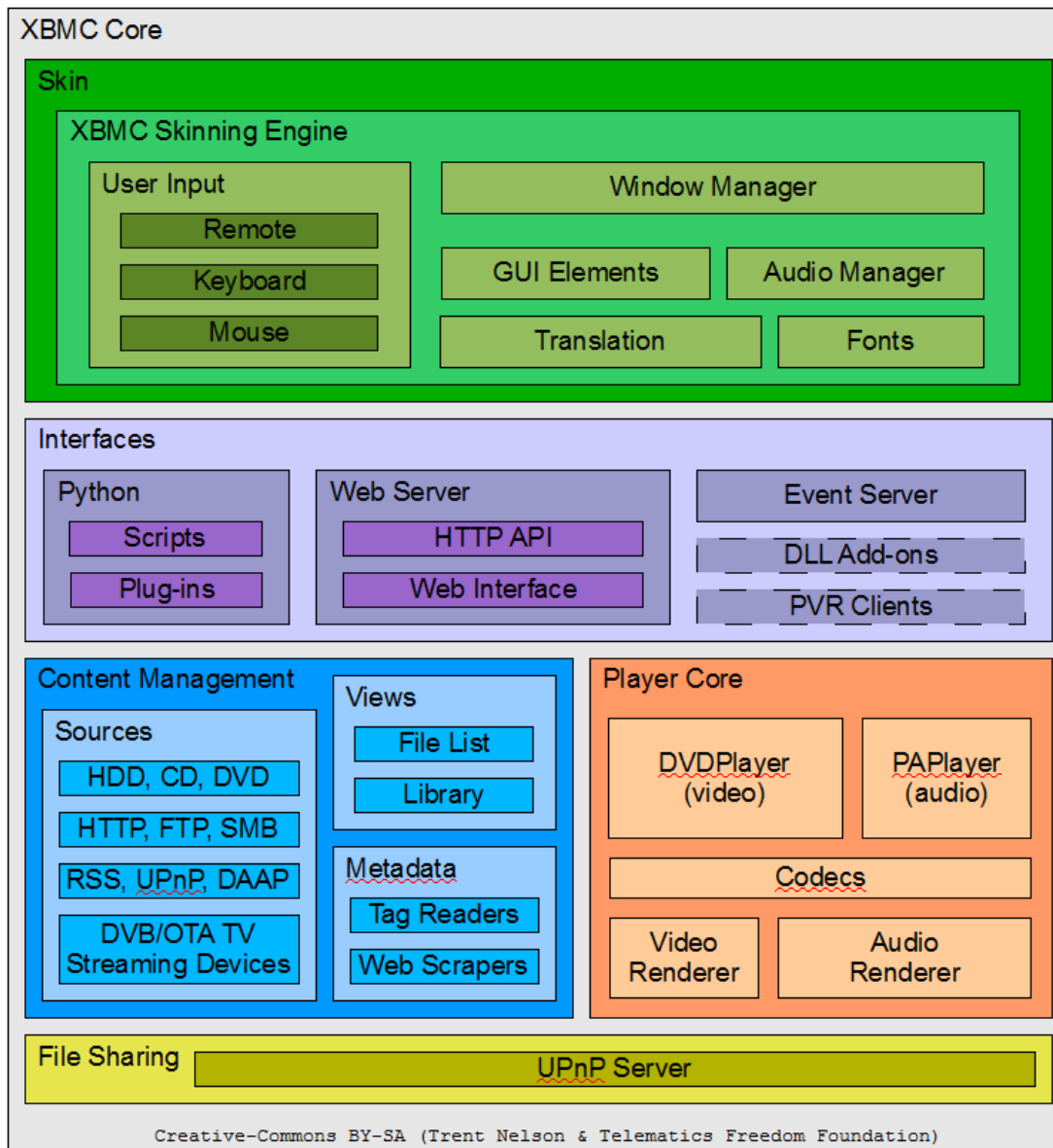


XBMC Architecture Overview



XBMC Media Center

XBMC Media Center is your ultimate multimedia hub. From the stunning interface, down to the helpful and enthusiastic community, it has everything you need to take your media enjoyment experience to the next level. Under the hood lies an extremely powerful and versatile custom skinning engine which allows skimmers to define nearly every aspect of the application and make their wildest dreams a reality. XBMC's library mode will take that mess of media files you have collected over the years and present them to you in immaculate organization. It taps several meta data sources for a plethora of information to enrich your experience. In addition to media on your computer, XBMC can stream content from a multitude of on-line resources for your enjoyment. A python interface lets you customize behavior and adds yet another source of content via scripts and plug-ins. Playback is handled by a pair of in-house players. One for music and another for video, each leveraging the very best of open source codecs ensuring XBMC can handle anything you throw at it. These features and many more make XBMC your best choice in home theater computer software.

Graphical User Interface and Skinning Engine

Dubbed *guilib* internally, XBMC's skinning engine is one of the most powerful available in the HTPC market. The XBMC core defines a minimum set of controls for each window to do its job but the skin is free to present these and any additional controls as they wish. Skin authors have all of the usual GUI elements at their disposal; labels, buttons, radio buttons, text boxes, spin boxes, sliders, scroll bars, control lists and control groups are all completely customizable. In addition, *guilib* takes advantage of today's 3D graphics hardware to provide silky smooth, animations and transitions.

Skins are defined by text files formatted in the XML mark-up language and images. There is one XML file to define each window and dialog defined by the core application. In this file, the position, dimensions, visibility, images and navigation details of each control are specified, as well as any animations and transitions for the window. Skins can also add more controls to each

window that execute XBMC built-in functions and scripts, raise specific windows, or query player and system information. Control visibility can be conditional based on what the application is doing at the time and can be delayed if need be. There is also the ability for skins to add their own settings to control aspects of the look.

The skin engine also contains several other parts of the interface. There is a window manager which keeps track of view states of each window such as list sort orders and currently selected control. An audio manager plays navigation sounds on cursor movement, click and back actions. XBMC has been translated to dozens of languages. *guilib* handles all string localization and also lets skins declare their own strings with translations. The last responsibility of *guilib* is user input, taking events from a remote control, keyboard, mouse or some other device through an event client and passing them on to the application.

Interfaces

XBMC offers several interfaces to control, modify behavior and add content. The built in Python interpreter provides two of these interfaces. The first, scripts, are mini programs designed to stand alone from XBMC. The *xbmc* and *xbmcgui* modules define the Python API into XBMC. Functions for controlling players, querying meta data, manipulating settings and the like are all available through the *xbmc* module. Script graphical interfaces are designed using the *xbmcgui* module, all of the controls available in the skinning engine can also be used for scripting. Using the WindowXML class in the *xbmcgui* module, the script can be skinned in an identical fashion to XBMC itself. Scripts can perform their task in the background with no user interactivity at all, or obtain a fully customizable window for a clean graphical interface.

The Python interpreter also provides file system (or directory) plug-ins, commonly referred to simply as “plugins”. These are special python scripts designed to fill a directory listing with content which may not be easily described otherwise and fit it in to the normal categories provided by XBMC (music, video, pictures and programs). Plug-ins take their visual queues

from the active XBMC skin so no specific skinning is necessary or available. They can also add items to the XBMC context menu when the plug-in is active, as well as define their own settings. File system plug-ins are often used to integrate content from websites (eg. YouTube) into XBMC.

Remote access to XBMC is provided via an embedded web server. It serves a web page offering the ability to browse your media and control the XBMC session from your web browser. The web server also provides the XBMC HTTP-API for developers to add XBMC control to their applications. It exposes full control over XBMC as well as information regarding what is currently playing and querying the library databases. Commands are executed by making a simple HTTP GET request and any information or status codes are returned in HTML format. Adding new functions to this interface is very simple in the XBMC code base.

A recently added interface into XBMC is the Event Server. The aim of this API is to allow developers to add support for various types of input devices. It listens on a UDP socket for Event Clients to connect and send events. The clients run separately from XBMC and can be written in any programming language that supports UDP networking. They commonly leverage other libraries written to handle the target input device. Their main interaction is sending button presses or joystick action to XBMC, but can also send notification messages for events such as battery status. One popular event client adds support for the Sony Playstation 3 bluetooth six axis game pad.

There are two new APIs currently in development branches of XBMC. The first, DLL Add-ons, aims to add features similar to the Python interpreter. The difference will be that the applications will be written in C or C++ and compiled into modules, removing the python interpreter overhead. XBMC can then load and execute them at the user's request. Secondly, PVR Clients, will allow interaction with PVR back end applications such as MythTV and VDR. This will give XBMC OTA, DVB, etc card access without having to actually support any

hardware itself. When completed, users will be able to fully control features of supported PVR applications, including; live TV, record timers, time shifting and on-screen guides.

Content Management

In order for content to be visible in XBMC a source must be created to point to it. There are several available sources for media content supported out of box. Locally, any content on the users hard disk, CD-ROM or DVD-ROM can be added. Remote sources, though, are where XBMC really gets interesting. Users can point XBMC at another computer serving their media via SMB (Windows File Sharing), HTTP, FTP, UPnP and DAAP (iTunes 6 sharing). Additionally sources for some service can be configured without support software such as Last.fm and shoutcast radio streams and OTA/DVB TV devices, such as dbox, tuxbox and MythTV live TV, which export the video stream over the local network.

Once a source is added, all of its contents is immediately available via the file view. This view presents media as you would expect in a file browser, with an icon, the file name, and a few other details pertinent to the selected sort order. If source contains static data (ie. the media isn't a stream), it can now be scanned into the library. This involves selecting a content type for videos; one of TV, Movies or Music Videos, selecting a website to scrape meta data from and answering a few questions as to how the files are laid out. XBMC will now read any tags and hunt out meta data on-line, storing this information in a database for quick easy access, later.

Library view is now available, providing a far richer experience. Fanart (backgrounds that relate to the focused media item) will be displayed in some view modes. In the music section, artist biographies and album reviews are available. For videos, you can now see play times, air dates, ratings, season and episode numbers, actors and a multitude of other information. This meta data is then used to organize media files and present them to in a friendly fashion. Another nice feature of the library is smart play lists. These are automatically generate play lists based on user selected values of meta data fields.

Player Core

Playback quality is paramount to XBMC's success. To that end, a pair of custom players are employed to ensure the utmost performance. DVDPlayer is used for all video playback. It is based off of the FFMpeg multimedia decoding library and features playback of all popular audio and video codecs, several subtitle formats as well as DVD menu support. Music playback is handled by PAPlayer. It takes advantage of several open source audio codecs and features gap-less playback and cross-fade. There is even a special codec that calls in to DVDPlayer on the off chance that PAPlayer doesn't recognize a file type. PAPlayer also has a facility to export the raw audio samples supporting reactive visualizations.

File Sharing

XBMC provides local network file sharing via a UPnP server. It relies on the library features, and allows for the content and meta-data in the library to be provided to other UPnP clients on your network. The server strives to be DLNA compliant, so it will work with popular clients such as Windows Media Player, Xbox360 and Playstation3 assuming they support the format being requested.