I03 REST API design considerations

The REST APIs provide a powerful interface to browse and search media servers, and control media renderers in a home network and provide a robust way to render playlists on media renderers automatically. It abstracts much of the complexity of network access and network device control from the application development. Nonetheless the controlled devices are only accessible through a network and the application design has to take this into account. Also, it is important to collect experience with networked device control, especially UPnP & DLNA control points, media servers and media renderers, before designing the application.

Response times of network-controlled devices can pose serious issues for an application. It is strongly recommended not to couple UI implementation and web API requests directly. A good application design should be prepared to handle response times of up to several seconds. The typical design for an application accessing the REST API will provide a multi-threaded request queue to e.g. load a media server directory and provide state machines for media renderer control, to prevent the application from blocking and to allow a user to keep interacting with the application.

Another important design consideration is the amount of metadata involved. Media servers can easily have directories with more than 10000 items. For most applications, especially on mobile and embedded devices, it's impossible to store the related metadata in memory (e.g. in a list box). In addition, the loading time for all the data would be unacceptable for a user trying to interact with the application. Therefore the application design should allow to only load the small number of visible items in a list view, plus possibly a cache for next few items to allow smooth scrolling. Loading this data should be done through the above mentioned multi-threaded request queue. The application should show some intermediate text or feedback to the user when scrolling, or jumping, too fast (e.g. each line shows „...") and then refresh the displayed text or graphic as soon as the request queue has processed and the respective media item has been loaded. Invisible items in the list view should release their (test) data to recover memory. This could be done by a background thread that's checking for text items out of the currently visible windows of items.