NodeLib – Functional Specification

Timothy Place, Pascal Baltazar

2009-04-16

Abstract

1 Overview

The **NodeLib** is a library within the Jamoma Modular framework for locating and communicating with nodes in the Jamoma system. A node is any point in the system whose name would occur between slashes when represented using Open Sound Control nomenclature.

This spec is not complete.

This work should be done in the experimental branch.

2 Use Cases

2.1 Phase 1 Use Cases

2.1.1 Case 1: Pascal the message sender

Pascal, besides being a brilliant mathematician and developing his famous triangle for showing binomial coefficients, is obsessed with the efficiency of sending messages to nodes. Without the NodeLib, messages are broadcast to all modules and then filtered at each module. If there are 1653 modules, then to send one message will entail sending it once, receiving it 1653 times, adn then filtering it 1653 times – even though it is only intended for one module. In reality the filtering is much more expensive, particularly where wildcard patterns are involved. Pascal knows this and wishes the message could be sent directly.

2.1.2 Case 2: Cartier the explorer

Not happy with discovering Canada, Cartier wishes to explore the topology of the nodes in the Jamoma namespace. To do this Cartier needs to know how to access the root of the namespace's tree structure. Then any node within the tree structure (including the root) needs to be able to report what children it has (that it is aware of). Any given node also needs to know some things about itself and be able to report them.

2.1.3 Case 3: Bob the tracker

Bob, a sound engineer, is used to using tracks for grouping effects, and would like to find the same thing in Jamoma. He would then organize his setup by having a series of /track nodes between his groups of modules (e.g. /track.2/filter.1, /track.3/player.1, etc...)

2.2 Phase 2 Use Cases

2.2.1 Case 4: ???? the instance manager

???? wants to manage instances in his Jamoma setup. For example, he has a collection of player modules (named /player.1 to /player.n), and want to play a sound on one of them. As he is a lazy guy, and likes to outsource his brain activity to computers, ???? would like the system to automatically manage the instance to the first available player in this group. This could be the case as well for a module having sub-modules (consider e.g. a mixing console module, with several tracks) This would mean that modules (or nodes) would have to have a @busy attribute.

2.2.2 Case 5: ???? the wildcard player

/module.*/parameter, /*/parameter, /module.2/track.*/parameter
/module.[2-4]/parameter
need to keep track of the namespace in a hashtab ?

3 Features

3.1 Initial Features

Initially focused on working only for local contexts (e.g. inside of a running Max process)

- Feedback suppression: need the ability to control feedback, including feedback through networks (suggests even serial numbers that are passed?)
- Lightweight query system and syntax
- jcom.send and jcom.receive
- wildcards
- fast hash-table based daemon running in the main thread

3.2 Later Features

Including expansion to working with remote contexts (other processes or remote machines).

- [Aliases] ability to create nodes which are aliases to other nodes
- jcom.send= and jcom.receive=
- running the daemon in another thread?
- communicating with a network

A list of comments, notes and feature requests can be found in the Jamoma WIKI under `http://groupware.bek.no/groups/jamoma/wiki/1f98f/NodeLib.html`.