Node: Public Virtual Methods

```
/** Called after construction to give the node a chance to modify its topology.
    This should return true if any changes were made to the topology as this
    indicates that the method may need to be called again after other nodes have
    had their topology changed.
*/
virtual bool transform (Node& /*rootNode*/) { return false; }

/** Should return all the inputs directly feeding in to this node. */
virtual std::vector<Node*> getDirectInputNodes() { return {}; }

/** Should return the properties of the node.
    This should not be called until after initialise.
*/
virtual NodeProperties getNodeProperties() = 0;

/** Should return true when this node is ready to be processed.
    This is usually when its input's output buffers are ready.
*/
virtual bool isReadyToProcess() = 0;
```

```cpp
/** Holds some really basic properties of a node */
struct NodeProperties
{
    bool hasAudio = false;
    bool hasMidi = false;
    int numberOfChannels = 0;
    int latencyNumSamples = 0;
    size_t nodeID = 0;
};
```

Node: Public Virtual Methods

```
/** Called after construction to give the node a chance to modify its topology.
    This should return true if any changes were made to the topology as this
    indicates that the method may need to be called again after other nodes have
    had their topology changed.
*/
virtual bool transform (Node& /*rootNode*/) { return false; }

/** Should return all the inputs directly feeding in to this node. */
virtual std::vector<Node*> getDirectInputNodes() { return {}; }

/** Should return the properties of the node.
    This should not be called until after initialise.
*/
virtual NodeProperties getNodeProperties() = 0;

/** Should return true when this node is ready to be processed.
    This is usually when its input's output buffers are ready.
*/
virtual bool isReadyToProcess() = 0;
```

Node: Protected Virtual Methods

```
/** Called once before playback begins for each node.
    Use this to allocate buffers etc.
    This step can be used to modify the topology of the graph (i.e. add/remove nodes).
    However, if you do this, you must make sure to call initialise on them so they are
    fully prepared for processing.
*/
virtual void prepareToPlay (const PlaybackInitialisationInfo&) {}

/** Called once on all Nodes before they are processed.
    This can be used to prefetch audio data or update mute statuses etc..
*/
virtual void prefetchBlock (juce::Range<int64_t> /*referenceSampleRange*/) {}

/** Called when the node is to be processed.
    This should add in to the buffers available making sure not to change their size at all.
*/
virtual void process (const ProcessContext&) = 0;
```