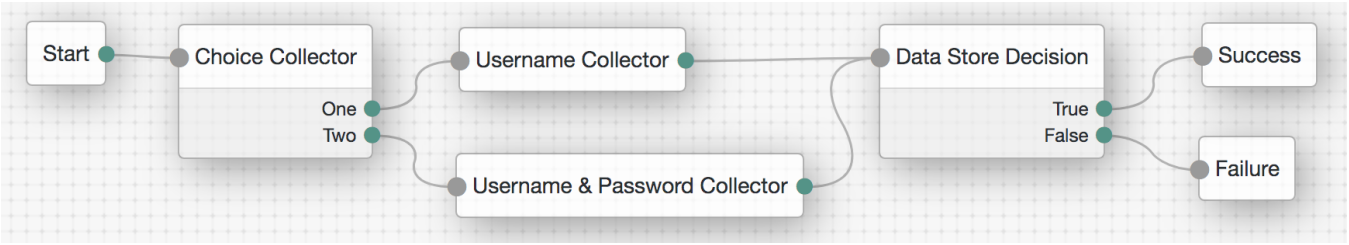
# Auth Tree Dependency Checking




In the tree above, the first collector collects only the username and the second collects both the username and the password. The DataStoreDecision node requires both attributes to succeed and both are required. The tree above creates an error if the "One" path is chosen from the ChoiceCollector node.

Tree validation starts at the first node in the tree, in this case the ChoiceCollector, and traverses all possible paths to the terminus nodes, i.e. Success and Failure. If we number the nodes above, beginning with Start, as 1, 2, 3 (for the Username collector), 4 (for the Username & password collector), 5, 6 (for Success) and 7 (for Failure), the possible paths are:

1 > 2 > 3 > 5 > 6

1 > 2 > 3 > 5 > 7

1 > 2 > 4 > 5 > 6

1 > 2 > 4 > 5 > 7

Validation checks each node's requirements to ensure that the nodes which came before provided the required data. To do this on a node-by-node basis the paths above are broken down further such that each node in the tree (except Success and Failure) is a "terminus" for the path. For the tree above this results in:

1 > 2

1 > 2 > 3

1 > 2 > 4

1 > 2 > 3 > 5

1 > 2 > 4 > 5

(note that Success and Failure are not calculated as they have no requirements)

For each of these "subtrees" a list of unsatisfied requirements is generated. For the tree and paths above:

1 > 2 : no unsatisfied requirements

1 > 2 > 3 : no unsatisfied requirements

1 > 2 > 4 : no unsatisfied requirements

1 > 2 > 3 > 5 : "password" is unsatisfied

1 > 2 > 4 > 5 : no unsatisfied requirements

The full set of possible paths and subpaths is reported to the API caller, complete with the list of unsatisfied requirements for each path.

Beyond hard requirements each node may declare inputs that will be consumed if present but which will not prevent proper operation of the node if they are absent. Likewise, each node may declare outputs that are made available only for certain "outcomes" (One and Two in the Choice Collector example, for instance). The path and requirement analysis takes these optional/outcome-dependent values into account when calculating requirements. Unsatisfied optional "requirements" will result in an informational notice in the results rather than an error notice.