# api\catalog.go

E:\workspace\go\consul\vendor\github.com\hashicorp\consul\api\catalog.go

## Catalog.Nodes

// Nodes is used to query all the known nodes  
**func** (c \*Catalog) Nodes(q \*QueryOptions) ([]\*Node, \*QueryMeta, error) {  
 r := c.c.newRequest("GET", "/v1/catalog/nodes")  
 r.setQueryOptions(q)  
 rtt, resp, err := requireOK(c.c.doRequest(r))  
 **if** err != nil {  
 **return** nil, nil, err  
 }  
 **defer** resp.Body.Close()  
  
 qm := &QueryMeta{}  
 parseQueryMeta(resp, qm)  
 qm.RequestTime = rtt  
  
 **var** out []\*Node  
 **if** err := decodeBody(resp, &out); err != nil {  
 **return** nil, nil, err  
 }  
 **return** out, qm, nil  
}

### 调用

#### /v1/catalog/nodes

r := c.c.newRequest("GET", "/v1/catalog/nodes")

# consul\catalog\_endpoint.go

E:\workspace\go\consul\vendor\github.com\hashicorp\consul\consul\catalog\_endpoint.go

## Catalog.ListNodes

// ListNodes is used to query the nodes in a DC  
**func** (c \*Catalog) ListNodes(args \*structs.DCSpecificRequest, reply \*structs.IndexedNodes) error {  
 **if** done, err := c.srv.forward("Catalog.ListNodes", args, args, reply); done {  
 **return** err  
 }  
  
 **return** c.srv.blockingQuery(  
 &args.QueryOptions,  
 &reply.QueryMeta,  
 **func**(ws memdb.WatchSet, state \*state.Store) error {  
 **var** index uint64  
 **var** nodes structs.Nodes  
 **var** err error  
 **if** len(args.NodeMetaFilters) > 0 {  
 index, nodes, err = state.NodesByMeta(ws, args.NodeMetaFilters)  
 } **else** {  
 index, nodes, err = state.Nodes(ws)  
 }  
 **if** err != nil {  
 **return** err  
 }  
  
 reply.Index, reply.Nodes = index, nodes  
 **if** err := c.srv.filterACL(args.Token, reply); err != nil {  
 **return** err  
 }  
 **return** c.srv.sortNodesByDistanceFrom(args.Source, reply.Nodes)  
 })  
}

### 调用

#### state.Nodes

index, nodes, err = state.Nodes(ws)

# consul\state\catalog.go

E:\workspace\go\consul\vendor\github.com\hashicorp\consul\consul\state\catalog.go

## Store.Nodes

// Nodes is used to return all of the known nodes.  
**func** (s \*Store) Nodes(ws memdb.WatchSet) (uint64, structs.Nodes, error) {  
 tx := s.db.Txn(false)  
 **defer** tx.Abort()  
  
 // Get the table index.  
 idx := maxIndexTxn(tx, "nodes")  
  
 // Retrieve all of the nodes  
 nodes, err := tx.Get("nodes", "id")  
 **if** err != nil {  
 **return** 0, nil, fmt.Errorf("failed nodes lookup: %s", err)  
 }  
 ws.Add(nodes.WatchCh())  
  
 // Create and return the nodes list.  
 **var** results structs.Nodes  
 **for** node := nodes.Next(); node != nil; node = nodes.Next() {  
 results = append(results, node.(\*structs.Node))  
 }  
 **return** idx, results, nil  
}

# command\agent\catalog\_endpoint.go

E:\workspace\go\consul\vendor\github.com\hashicorp\consul\command\agent\catalog\_endpoint.go

## HTTPServer.CatalogNodes

**func** (s \*HTTPServer) CatalogNodes(resp http.ResponseWriter, req \*http.Request) (**interface**{}, error) {  
 // Setup the request  
 args := structs.DCSpecificRequest{}  
 s.parseSource(req, &args.Source)  
 args.NodeMetaFilters = s.parseMetaFilter(req)  
 **if** done := s.parse(resp, req, &args.Datacenter, &args.QueryOptions); done {  
 **return** nil, nil  
 }  
  
 **var** out structs.IndexedNodes  
 **defer** setMeta(resp, &out.QueryMeta)  
 **if** err := s.agent.RPC("Catalog.ListNodes", &args, &out); err != nil {  
 **return** nil, err  
 }  
 translateAddresses(s.agent.config, args.Datacenter, out.Nodes)  
  
 // Use empty list instead of nil  
 **if** out.Nodes == nil {  
 out.Nodes = make(structs.Nodes, 0)  
 }  
 **return** out.Nodes, nil  
}