Details ✓ Valid go.mod file ? ✓ Redistributable license ? ✓ Tagged version ?

✓ Stable version ?

Learn more

Repository github.com/go-zookeeper/zk

Overview

README

Discover Packages > github.com/go-zookeeper/zk

zk

package module

License

3-clause BSD. See LICENSE file.

Collapse ▲

Documentation

Overview

Package zk is a native Go client library for the ZooKeeper orchestration service.

Index

func (s State) String() string

Constants

View Source

```
const (
    FlagEphemeral = 1
    FlagSequence  = 2
    FlagTTL       = 4
)
```

View Source

```
const (
    PermRead = 1 << iota
    PermWrite
    PermCreate
    PermDelete
    PermAdmin
    PermAll = 0x1f
)
```

Constants for ACL permissions

View Source

```
const (
    DefaultPort = 2181
)
```

Variables

View Source

```
var (
    ErrConnectionClosed        = errors.New("zk: connection closed")
    ErrUnknown                 = errors.New("zk: unknown error")
    ErrAPIError                = errors.New("zk: api error")
    ErrNoNode                  = errors.New("zk: node does not exist")
    ErrNoAuth                  = errors.New("zk: not authenticated")
    ErrBadVersion              = errors.New("zk: version conflict")
    ErrNoChildrenForEphemerals = errors.New("zk: ephemeral nodes may not have chi
    ErrNodeExists              = errors.New("zk: node already exists")
    ErrNotEmpty                = errors.New("zk: node has children")
    ErrSessionExpired          = errors.New("zk: session has been expired by the
    ErrInvalidACL              = errors.New("zk: invalid ACL specified")
    ErrInvalidFlags            = errors.New("zk: invalid flags specified")
    ErrAuthFailed              = errors.New("zk: client authentication failed")
    ErrClosing                 = errors.New("zk: zookeeper is closing")
    ErrNothing                 = errors.New("zk: no server responsees to process"
    ErrSessionMoved            = errors.New("zk: session moved to another server,
    ErrReconfigDisabled        = errors.New("attempts to perform a reconfiguratio
    ErrBadArguments            = errors.New("invalid arguments")
)
```

View Source

```
var (
    // ErrDeadlock is returned by Lock when trying to lock twice without unlockin
    ErrDeadlock = errors.New("zk: trying to acquire a lock twice")
    // ErrNotLocked is returned by Unlock when trying to release a lock that has
    ErrNotLocked = errors.New("zk: not locked")
)
```

View Source

```
var (
    ErrUnhandledFieldType = errors.New("zk: unhandled field type")
    ErrPtrExpected        = errors.New("zk: encode/decode expect a non-nil pointe
    ErrShortBuffer        = errors.New("zk: buffer too small")
)
```

View Source

```
var ErrInvalidPath = errors.New("zk: invalid path")
```

ErrInvalidPath indicates that an operation was being attempted on an invalid path. (e.g. empty path)

View Source

```
var ErrNoServer = errors.New("zk: could not connect to a server")
```

ErrNoServer indicates that an operation cannot be completed because attempts to connect to all servers in the list failed.

Functions

func FLWRuok

```
func FLWRuok(servers []string, timeout time.Duration) []bool
```

FLWRuok is a FourLetterWord helper function. In particular, this function pulls the ruok output from each server.

func FormatServers

```
func FormatServers(servers []string) []string
```

FormatServers takes a slice of addresses, and makes sure they are in a format that resembles <addr>:<port>. If the server has no port provided, the DefaultPort constant is added to the end.

func WithDialer

```
func WithDialer(dialer Dialer) connOption
```

WithDialer returns a connection option specifying a non-default Dialer.

func WithEventCallback

```
func WithEventCallback(cb EventCallback) connOption
```

WithEventCallback returns a connection option that specifies an event callback. The callback must not block - doing so would delay the ZK go routines.

func WithHostProvider

```
func WithHostProvider(hostProvider HostProvider) connOption
```

WithHostProvider returns a connection option specifying a non-default HostProvider.

func WithLogInfo

```
func WithLogInfo(logInfo bool) connOption
```

WithLogInfo returns a connection option specifying whether or not information messages shoud be logged.

func WithLogger

```
func WithLogger(logger Logger) connOption
```

WithLogger returns a connection option specifying a non-default Logger

func WithMaxBufferSize

```
func WithMaxBufferSize(maxBufferSize int) connOption
```

WithMaxBufferSize sets the maximum buffer size used to read and decode packets received from the Zookeeper server. The standard Zookeeper client for Java defaults to a limit of 1mb. For backwards compatibility, this Go client defaults to unbounded unless overridden via this option. A value that is zero or negative indicates that no limit is enforced.

This is meant to prevent resource exhaustion in the face of potentially malicious data in ZK. It should generally match the server setting (which also defaults ot 1mb) so that clients and servers agree on the limits for things like the size of data in an individual znode and the total size of a transaction.

For production systems, this should be set to a reasonable value (ideally that matches the server configuration). For ops tooling, it is handy to use a much larger limit, in order to do things like clean-up problematic state in the ZK tree. For example, if a single znode has a huge number of children, it is possible for the response to a "list children" operation to exceed this buffer size and cause errors in clients. The only way to subsequently clean up the tree (by removing superfluous children) is to use a client configured with a larger buffer size that can successfully query for all of the child names and then remove them. (Note there are other tools that can list all of the child names without an increased buffer size in the client, but they work by inspecting the servers' transaction logs to enumerate children instead of sending an online request to a server.

func WithMaxConnBufferSize

```
func WithMaxConnBufferSize(maxBufferSize int) connOption
```

WithMaxConnBufferSize sets maximum buffer size used to send and encode packets to Zookeeper server. The standard Zookeepeer client for java defaults to a limit of 1mb. This option should be used for non-standard server setup where znode is bigger than default 1mb.

Types

type ACL

```
type ACL struct {
    Perms  int32
    Scheme string
    ID     string
}
```

func AuthACL

```
func AuthACL(perms int32) []ACL
```

AuthACL produces an ACL list containing a single ACL which uses the provided permissions, with the scheme "auth", and ID "", which is used by ZooKeeper to represent any authenticated user.

func DigestACL

```
func DigestACL(perms int32, user, password string) []ACL
```

func WorldACL

```
func WorldACL(perms int32) []ACL
```

WorldACL produces an ACL list containing a single ACL which uses the provided permissions, with the scheme "world", and ID "anyone", which is used by ZooKeeper to represent any user at all.

type CheckVersionRequest

```
type CheckVersionRequest PathVersionRequest
```

type Conn

```
type Conn struct {
    // contains filtered or unexported fields
}
```

func Connect

```
func Connect(servers []string, sessionTimeout time.Duration, options ...connOpt
ion) (*Conn, <-chan Event, error)
```

Connect establishes a new connection to a pool of zookeeper servers. The provided session timeout sets the amount of time for which a session is considered valid after losing connection to a server. Within the session timeout it's possible to reestablish a connection to a different server and keep the same session. This is means any ephemeral nodes and watches are maintained.

func ConnectWithDialer

```
func ConnectWithDialer(servers []string, sessionTimeout time.Duration, dialer D
ialer) (*Conn, <-chan Event, error)
```

ConnectWithDialer establishes a new connection to a pool of zookeeper servers using a custom Dialer. See Connect for further information about session timeout. This method is deprecated and provided for compatibility: use the WithDialer option instead.

func (*Conn) AddAuth

```
func (c *Conn) AddAuth(scheme string, auth []byte) error
```

func (*Conn) Children

```
func (c *Conn) Children(path string) ([]string, *Stat, error)
```

func (*Conn) ChildrenW

```
func (c *Conn) ChildrenW(path string) ([]string, *Stat, <-chan Event, error)
```

func (*Conn) Close

```
func (c *Conn) Close()
```

Close will submit a close request with ZK and signal the connection to stop sending and receiving packets.

func (*Conn) Create

```
func (c *Conn) Create(path string, data []byte, flags int32, acl []ACL) (string
, error)
```

func (*Conn) CreateContainer

added in v1.0.2

```
func (c *Conn) CreateContainer(path string, data []byte, flags int32, acl []ACL
) (string, error)
```

func (*Conn) CreateProtectedEphemeralSequential

```
func (c *Conn) CreateProtectedEphemeralSequential(path string, data []byte, acl
[]ACL) (string, error)
```

CreateProtectedEphemeralSequential fixes a race condition if the server crashes after it creates the node. On reconnect the session may still be valid so the ephemeral node still exists. Therefore, on reconnect we need to check if a node with a GUID generated on create exists.

func (*Conn) CreateTTL

added in v1.0.2

```
func (c *Conn) CreateTTL(path string, data []byte, flags int32, acl []ACL, ttl
time.Duration) (string, error)
```

func (*Conn) Delete

```
func (c *Conn) Delete(path string, version int32) error
```

func (*Conn) Exists

```
func (c *Conn) Exists(path string) (bool, *Stat, error)
```

func (*Conn) ExistsW

```
func (c *Conn) ExistsW(path string) (bool, *Stat, <-chan Event, error)
```

func (*Conn) Get

```
func (c *Conn) Get(path string) ([]byte, *Stat, error)
```

func (*Conn) GetACL

```
func (c *Conn) GetACL(path string) ([]ACL, *Stat, error)
```

func (*Conn) GetW

```
func (c *Conn) GetW(path string) ([]byte, *Stat, <-chan Event, error)
```

GetW returns the contents of a znode and sets a watch

func (*Conn) IncrementalReconfig

added in v1.0.2

```
func (c *Conn) IncrementalReconfig(joining, leaving []string, version int64) (*
Stat, error)
```

IncrementalReconfig is the zookeeper reconfiguration api that allows adding and removing servers by lists of members. For more info refer to the ZK documentation.

An optional version allows for conditional reconfigurations, -1 ignores the condition.

Returns the new configuration znode stat.

func (*Conn) Multi

```
func (c *Conn) Multi(ops ...interface{}) ([]MultiResponse, error)
```

Multi executes multiple ZooKeeper operations or none of them. The provided ops must be one of *CreateRequest, *DeleteRequest, *SetDataRequest, or *CheckVersionRequest.

func (*Conn) Reconfig

added in v1.0.2

```
func (c *Conn) Reconfig(members []string, version int64) (*Stat, error)
```

Reconfig is the non-incremental update functionality for Zookeeper where the list provided is the entire new member list. For more info refer to the ZK documentation.

An optional version allows for conditional reconfigurations, -1 ignores the condition.

Returns the new configuration znode stat.

func (*Conn) Server

```
func (c *Conn) Server() string
```

Server returns the current or last-connected server name.

func (*Conn) SessionID

```
func (c *Conn) SessionID() int64
```

SessionID returns the current session id of the connection.

func (*Conn) Set

```
func (c *Conn) Set(path string, data []byte, version int32) (*Stat, error)
```

func (*Conn) SetACL

```
func (c *Conn) SetACL(path string, acl []ACL, version int32) (*Stat, error)
```

func (*Conn) SetLogger

```
func (c *Conn) SetLogger(l Logger)
```

SetLogger sets the logger to be used for printing errors. Logger is an interface provided by this package.

func (*Conn) State

```
func (c *Conn) State() State
```

State returns the current state of the connection.

func (*Conn) Sync

```
func (c *Conn) Sync(path string) (string, error)
```

type CreateContainerRequest

added in v1.0.2

```
type CreateContainerRequest CreateRequest
```

type CreateRequest

```
type CreateRequest struct {
	Path  string
	Data  []byte
	Acl   []ACL
	Flags int32
}
```

type CreateTTLRequest

added in v1.0.2

```
type CreateTTLRequest struct {
	Path  string
	Data  []byte
	Acl   []ACL
	Flags int32
	Ttl   int64 // ms
}
```

type DNSHostProvider

```
type DNSHostProvider struct {
    // contains filtered or unexported fields
}
```

DNSHostProvider is the default HostProvider. It currently matches the Java StaticHostProvider, resolving hosts from DNS once during the call to Init. It could be easily extended to re-query DNS periodically or if there is trouble connecting.

func (*DNSHostProvider) Connected

```
func (hp *DNSHostProvider) Connected()
```

Connected notifies the HostProvider of a successful connection.

func (*DNSHostProvider) Init

```
func (hp *DNSHostProvider) Init(servers []string) error
```

Init is called first, with the servers specified in the connection string. It uses DNS to look up addresses for each server, then shuffles them all together.

func (*DNSHostProvider) Len

```
func (hp *DNSHostProvider) Len() int
```

Len returns the number of servers available

func (*DNSHostProvider) Next

```
func (hp *DNSHostProvider) Next() (server string, retryStart bool)
```

Next returns the next server to connect to. retryStart will be true if we've looped through all known servers without Connected() being called.

type DeleteRequest

```
type DeleteRequest PathVersionRequest
```

type Dialer

```
type Dialer func(network, address string, timeout time.Duration) (net.Conn, error
```

type ErrCode

```
type ErrCode int32
```

type Event

```
type Event struct {
    Type   EventType
    State  State
    Path   string // For non-session events, the path of the watched node.
    Err    error
    Server string // For connection events
}
```

type EventCallback

```
type EventCallback func(Event)
```

EventCallback is a function that is called when an Event occurs.

type EventType

```
type EventType int32
```

```
const (
    EventNodeCreated         EventType = 1
    EventNodeDeleted         EventType = 2
    EventNodeDataChanged     EventType = 3
    EventNodeChildrenChanged EventType = 4

    EventSession     EventType = -1
    EventNotWatching EventType = -2
)
```

func (EventType) String

```
func (t EventType) String() string
```

type HostProvider

```
type HostProvider interface {
	// Init is called first, with the servers specified in the connection string.
	Init(servers []string) error
	// Len returns the number of servers.
	Len() int
	// Next returns the next server to connect to. retryStart will be true if we'
	// all known servers without Connected() being called.
	Next() (server string, retryStart bool)
	// Notify the HostProvider of a successful connection.
	Connected()
}
```

HostProvider is used to represent a set of hosts a ZooKeeper client should connect to. It is an analog of the Java equivalent: http://svn.apache.org/viewvc/zookeeper/trunk/src/java/main/org/apache/zookeeper/client/HostProvider.java?view=markup

type Lock

```
type Lock struct {
	// contains filtered or unexported fields
}
```

Lock is a mutual exclusion lock.

func NewLock

```
func NewLock(c *Conn, path string, acl []ACL) *Lock
```

NewLock creates a new lock instance using the provided connection, path, and acl. The path must be a node that is only used by this lock. A lock instances starts unlocked until Lock() is called.

func (*Lock) Lock

```
func (l *Lock) Lock() error
```

Lock attempts to acquire the lock. It works like LockWithData, but it doesn't write any data to the lock node.

func (*Lock) LockWithData

added in v1.0.2

```
func (l *Lock) LockWithData(data []byte) error
```

LockWithData attempts to acquire the lock, writing data into the lock node. It will wait to return until the lock is acquired or an error occurs. If this instance already has the lock then ErrDeadlock is returned.

func (*Lock) Unlock

```
func (l *Lock) Unlock() error
```

Unlock releases an acquired lock. If the lock is not currently acquired by this Lock instance than ErrNotLocked is returned.

type Logger

```
type Logger interface {
    Printf(string, ...interface{})
}
```

Logger is an interface that can be implemented to provide custom log output.

```
var DefaultLogger Logger = defaultLogger{}
```

DefaultLogger uses the stdlib log package for logging.

type Mode

```
type Mode uint8
```

Mode is used to build custom server modes (leader|follower|standalone).

```
const (
    ModeUnknown    Mode = iota
    ModeLeader     Mode = iota
    ModeFollower   Mode = iota
    ModeStandalone Mode = iota
)
```

func (Mode) String

```
func (m Mode) String() string
```

type MultiResponse

```
type MultiResponse struct {
    Stat   *Stat
    String string
    Error  error
}
```

type PathVersionRequest

```
type PathVersionRequest struct {
    Path    string
    Version int32
}
```

type ServerClient

```
type ServerClient struct {
    Queued        int64
    Received      int64
    Sent          int64
    SessionID     int64
    Lcxid         int64
    Lzxid         int64
    Timeout       int32
    LastLatency   int32
    MinLatency    int32
    AvgLatency    int32
    MaxLatency    int32
    Established   time.Time
    LastResponse  time.Time
    Addr          string
    LastOperation string // maybe?
    Error         error
}
```

ServerClient is the information for a single Zookeeper client and its session. This is used to parse/extract the output fo the `cons` command.

type ServerClients

```
type ServerClients struct {
    Clients []*ServerClient
    Error   error
}
```

ServerClients is a struct for the FLWCons() function. It's used to provide the list of Clients.

This is needed because FLWCons() takes multiple servers.

func FLWCons

```
func FLWCons(servers []string, timeout time.Duration) ([]*ServerClients, bool)
```

FLWCons is a FourLetterWord helper function. In particular, this function pulls the ruok output from each server.

As with FLWSrvr, the boolean value indicates whether one of the requests had an issue. The Clients struct has an Error value that can be checked.

type ServerStats

```
type ServerStats struct {
    Sent        int64
    Received    int64
    NodeCount   int64
    MinLatency  int64
    AvgLatency  float64
    MaxLatency  int64
    Connections int64
    Outstanding int64
    Epoch       int32
    Counter     int32
    BuildTime   time.Time
    Mode        Mode
    Version     string
    Error       error
}
```

ServerStats is the information pulled from the Zookeeper `stat` command.

func FLWSrvr

```
func FLWSrvr(servers []string, timeout time.Duration) ([]*ServerStats, bool)
```

FLWSrvr is a FourLetterWord helper function. In particular, this function pulls the srvr output from the zookeeper instances and parses the output. A slice of *ServerStats structs are returned as well as a boolean value to indicate whether this function processed successfully.

If the boolean value is false there was a problem. If the *ServerStats slice is empty or nil, then the error happened before we started to obtain 'srvr' values. Otherwise, one of the servers had an issue and the "Error" value in the struct should be inspected to determine which server had the issue.

type SetDataRequest

```
type SetDataRequest struct {
	Path    string
	Data    []byte
	Version int32
}
```

type Stat

```
type Stat struct {
	Czxid          int64 // The zxid of the change that caused this znode to be c
	Mzxid          int64 // The zxid of the change that last modified this znode.
	Ctime          int64 // The time in milliseconds from epoch when this znode w
	Mtime          int64 // The time in milliseconds from epoch when this znode w
	Version        int32 // The number of changes to the data of this znode.
	Cversion       int32 // The number of changes to the children of this znode.
	Aversion       int32 // The number of changes to the ACL of this znode.
	EphemeralOwner int64 // The session id of the owner of this znode if the znod
	DataLength     int32 // The length of the data field of this znode.
	NumChildren    int32 // The number of children of this znode.
	Pzxid          int64 // last modified children
}
```

type State

```
type State int32
```

```
const (
	StateUnknown           State = -1
	StateDisconnected      State = 0
	StateConnecting        State = 1
	StateAuthFailed        State = 4
	StateConnectedReadOnly State = 5
	StateSaslAuthenticated State = 6
	StateExpired           State = -112
```

```
    StateConnected  = State(100)
    StateHasSession = State(101)
)
```

func (State) String

```
func (s State) String() string
```

Source Files

View all

conn.go
constants.go
dnshostprovider.go
flw.go
lock.go
structs.go
util.go

Directories

_examples

Slack

r/golang

Meetup

Golang Weekly

Copyright

Terms of Service

Privacy Policy

Report an Issue

golang.org

Google