EdgeX Registry Abstraction

Decouple EdgeX services from Consul

https://github.com/edgexfoundry/edgex-go/issues/797
Existing state

**config-seed**

*init.go*
- Checks if Consul is available by attempting to access /v1/agent/self path on consul service
- If available, returns pointer to Client

*populate.go*
- PUT properties into Consul KV (legacy Java services) – Obsolete now?
- PUT configuration into Consul KV
  * Does GET on each KV prior to PUT

**Requirements**
- Check if Registry Service running
- Create NewRegistry if running
- GET configuration values from Registry Service(nil if not there)
- PUT configuration values into Registry Service

**All Others services**

*use internal/pkg/consul/client.go*
- Uses mitchellh/consulstructure for watching for configuration changes

**Requirements**
- Register service with Registry Service
- Register health check URL with Registry Service
- Load configuration from Registry Service and map values into configuration struct
- Watch for configuration changes in Registry Service and notify current service with latest configuration
- Get service endpoint information

**device-sdk-go**

*clients/init.go*
- checks if Consul service is available
- checks in dependent services are available

*internal/pkg/consul/client.go*
- Similar to version in edgex-go, but doesn't use mitchellh/consulstructure to watch for changes

**Requirements**
- Check if dependent services are available.
- Same as edgex-go microservices
What we need in an Abstraction API

• Check if Registry Service is running
• Register current service with Registry Service
• Register health check URL with Registry Service
• Load configuration from Registry Service
• Put configuration into Registry Service
• Check if a configuration value exists in Registry Service
• Get a configuration value from Registry Service
• Put a configuration value into Registry Service
• Watch for configuration changes in Registry Service
  • Load new configuration from Registry Service when changed
  • Notify app that configuration changed
• Get service endpoint information from Registry Service
• Check with Registry Service if a dependent service is available
Proposed Abstract Registry API

RegistryClient
- Struct containing Service and Registry Service information
- Similar to existing ConsulConfig
- Info for connecting to Registry Service
- Info defining configuration pathing
- etc.

NewRegistryClient(registryInfo config.RegistryInfo, serviceInfo config.ServiceInfo, serviceKey string) (*RegistryClient, error)
- Loads Registry implementation
- validate plugin conforms to required API and saving pointers to functions
- Calls NewRegistryClient(registryInfo, serviceInfo) on the implementation

(registry *RegistryClient) Register() error
- pass thru to implementation
(registry *RegistryClient) PutConfiguration(configuration interface{}) error
- pass thru to implementation
(registry *RegistryClient) GetConfiguration() (interface{}, error)
- pass thru to implementation
(registry *RegistryClient) IsRegistryRunning() bool
- pass thru to implementation
(registry *RegistryClient) ConfigurationValueExists(string name) (bool, error)
- pass thru to implementation
(registry *RegistryClient) GetConfigurationValue(string name) ([]byte, error)
- pass thru to implementation
(registry *RegistryClient) PutConfigurationValue(string name, []byte value) error
- pass thru to implementation
(registry *RegistryClient) WatchForChanges(updateChan chan<- interface{}, errChan chan<- error)
- pass thru to implementation
(registry *RegistryClient) GetServiceEndpoint(serviceld string) (ServiceEndpoint, error)
- pass thru to implementation
(registry *RegistryClient) IsServiceAvailable(serviceID string) bool
Proposed Registry implementation API

**NewRegistryClient**

```go
NewRegistryClient(registryInfo config.RegistryInfo, serviceInfo config.ServiceInfo, serviceKey string) (*RegistryClient, error)
```

- Saves the registry and service information
- Sets up connection to Registry Service
- Returns pointer to RegistryClient struct

**Register**

```go
Register(registry *RegistryClient) error
```

- Registers the service with the Registry Service
- Registers the health check callback with the Registry Service

**PutConfiguration**

```go
PutConfiguration(registry *RegistryClient, configuration interface{}) error
```

- Puts the configuration into the Registry Service using the configuration base path

**GetConfiguration**

```go
GetConfiguration(registry *RegistryClient) (interface{}, error)
```

- Gets the whole configuration from the Registry Service mapping values into the struct

**IsRegistryRunning**

```go
IsRegistryRunning(registry *RegistryClient) bool
```

- Determines if the Registry Service is running or not

**ConfigurationValueExists**

```go
ConfigurationValueExists(registry *RegistryClient, string name) (bool, error)
```

- Determines if the Registry Service has the value associated with the passed in name

**GetConfigurationValue**

```go
GetConfigurationValue(registry *RegistryClient, string name) ([]byte, error)
```

- Gets the value associated with the passed in name from the Registry Service. nil if not in Registry Service.

**PutConfigurationValue**

```go
PutConfigurationValue(registry *RegistryClient, string name, []byte value) error
```

- Puts the passed in value into the Registry Service associate with the passed in name

**WatchForChanges**

```go
WatchForChanges(registry *RegistryClient, updateChan chan<- interface{}, errChan chan<- error)
```

- Watches for configuration changes in the Registry Service.
- Changes are sent back on the updateChan channel and any errors on the errChan channel.
- Must be called as a go func

**GetServiceEndpoint**

```go
GetServiceEndpoint(registry *RegistryClient, serviceId string) (ServiceEndpoint, error)
```

- Returns the service endpoint information for the service ID passed in. nil if service not registered.

**IsServiceAvailable**

```go
IsServiceAvailable(registry *RegistryClient, serviceID string) bool
```

- Returns true if service identified by the passed in ID is currently available, false otherwise