GoVPP

Golang VPP Management Toolset

Rastislav Szabo
April 2017

GoVPP

- Golang toolset for VPP management
- API based on the Go bindings generated from the VPP binary API (JSON) and Go channels

- Currently used in:
  - Cloud-native VNF Agent (to be open-sourced)
  - Contiv-VPP integration

Project Scope

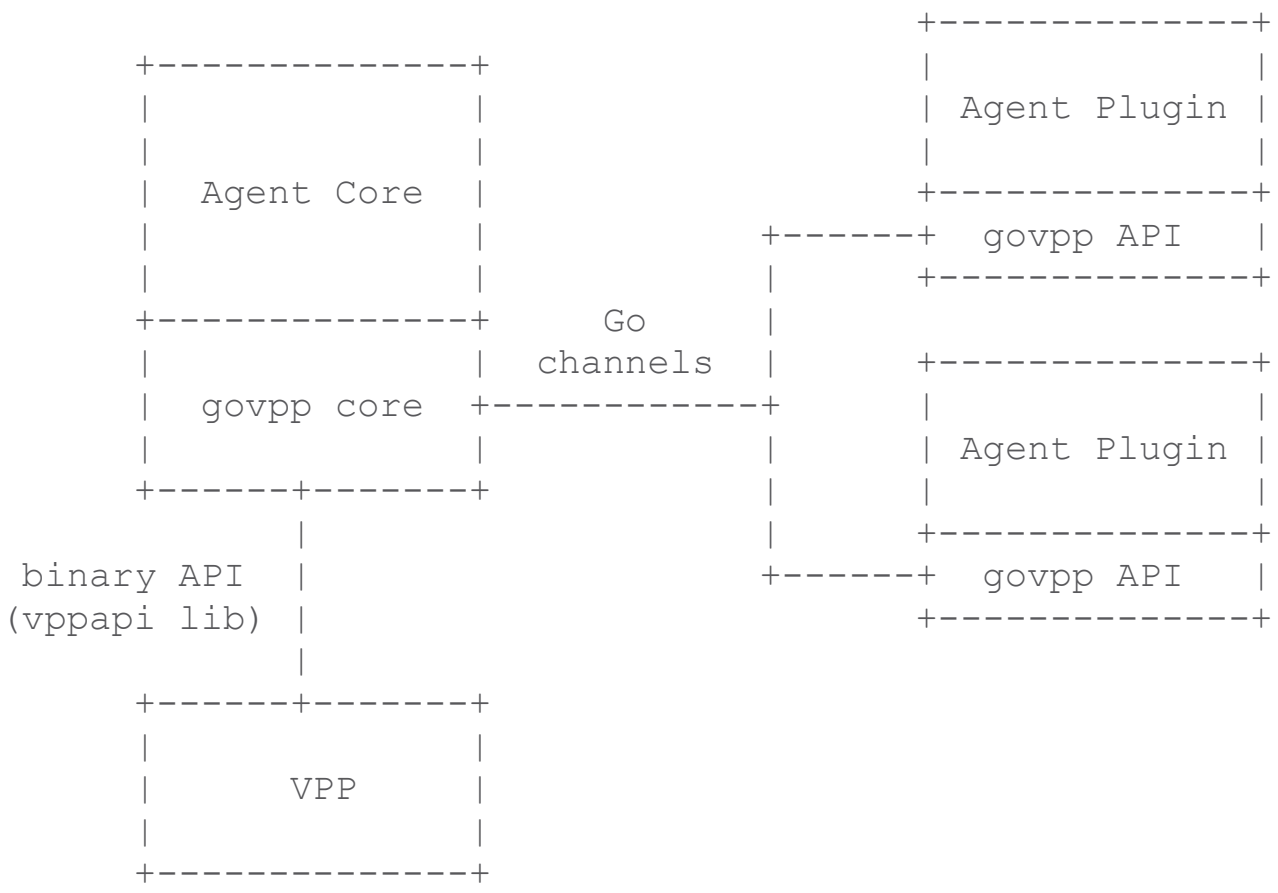
- Go bindings generator
  - JSON -> Go structs with annotations
- GoVPP API packages
  - API, core, VPP adapter
- Supporting code
  - Examples, unit tests, VPP mocker for unit testing

Out of scope:

- “vppapiclient” library (the former “pneum” library) used to communicate with VPP via CGO
- Binary API to JSON generator

Internal Structure

Ready for plugin-based infrastructure of the management agents:



Example of Generated Go Bindings

```
// ACLRule represents the VPP binary API data type
'acl_rule'.
// Generated from 'bin_api/acl.api.json', line 3:
//
//        ["acl_rule",
//            ["u8", "is_permit"],
//            ["u8", "is_ipv6"],
//            ["u8", "src_ip_addr", 16],
//            ["u8", "src_ip_prefix_len"],
//            ["u8", "dst_ip_addr", 16],
//            ["u8", "dst_ip_prefix_len"],
//            ["u8", "proto"],
//            ["u16", "srcport_or_icmptype_first"],
//            ["u16", "srcport_or_icmptype_last"],
//            ["u16", "dstport_or_icmpcode_first"],
//            ["u16", "dstport_or_icmpcode_last"],
//            ["u8", "tcp_flags_mask"],
//            ["u8", "tcp_flags_value"],
//            {"crc" : "0x2715e1c0"}
//        ],
//
type ACLRule struct {
   IsPermit                uint8
   IsIpv6                  uint8
   SrcIPAddr               []byte `struc:"[16]byte"`
   SrcIPPrefixLen          uint8
   DstIPAddr               []byte `struc:"[16]byte"`
   DstIPPrefixLen          uint8
   Proto                   uint8
   SrcportOrIcmptypeFirst uint16
   SrcportOrIcmptypeLast  uint16
   DstportOrIcmpcodeFirst uint16
   DstportOrIcmpcodeLast  uint16
   TCPFlagsMask            uint8
   TCPFlagsValue           uint8
}
```

```
// ACLAddReplace represents the VPP binary API message
'acl_add_replace'.
// Generated from 'bin_api/acl.api.json', line 43:
//
//        ["acl_add_replace",
//            ["u16", "_vl_msg_id"],
//            ["u32", "client_index"],
//            ["u32", "context"],
//            ["u32", "acl_index"],
//            ["u8", "tag", 64],
//            ["u32", "count"],
//            ["vl_api_acl_rule_t", "r", 0, "count"],
//            {"crc" : "0x3c317936"}
//        ],
//
type ACLAddReplace struct {
   ACLIndex uint32
   Tag      []byte `struc:"[64]byte"`
   Count    uint32 `struc:"sizeof=R"`
   R        []ACLRule
}
```

Example Usage

```
req := &acl.ACLAddReplace{
  ACLIndex: ^uint32(0),
  Tag:      []byte("access list 1"),
  R: []acl.ACLRule{
    {
      IsPermit:       1,
      SrcIPAddr:      net.ParseIP("10.0.0.0").To4(),
      SrcIPPrefixLen: 8,
      DstIPAddr:      net.ParseIP("192.168.1.0").To4(),
      DstIPPrefixLen: 24,
      Proto:          6,
    },
    {
      IsPermit:       1,
      SrcIPAddr:      net.ParseIP("8.8.8.8").To4(),
      SrcIPPrefixLen: 32,
      DstIPAddr:      net.ParseIP("172.16.0.0").To4(),
      DstIPPrefixLen: 16,
      Proto:          6,
    },
  },
}
reply := &acl.ACLAddReplaceReply{}

err := ch.SendRequest(req).ReceiveReply(reply)
fmt.Printf("%+v\n", reply)
```

Also supported:

- Go channel-based API
- Multipart replies
- Notifications
- Counters
  (+ specific package for interface counters)

Why a Separate Project?

- Go toolset is build on remote import paths & remote access to the repositories of the imported packages:

In Go code:

```
import "gerrit.fd.io/r/govpp"
```

To install GoVPP:

```
go get gerrit.fd.io/r/govpp
```

(“go get” effectively does “git clone” into GOPATH – we don’t want to be downloading all the VPP source code into GOPATH)

Initial Committers

Main Contact:

- Jan Medved (jmedved@cisco.com)
- Rastislav Szabo (raszabo@cisco.com)

Other Committers:

- Jozef Slezak (joslezak@cisco.com)
- Keith Burns (krb@cisco.com)
- Nikos Bregiannis (nbregian@cisco.com)


CISCO