Server Plugin

frp server plugin is aimed to extend frp's ability without modifying the Golang code.

An external server should run in a different process receiving RPC calls from frps. Before frps is doing some operations, it will send RPC requests to notify the external RPC server and act according to its response.

RPC request

RPC requests are based on JSON over HTTP.

When a server plugin accepts an operation request, it can respond with three different responses:

- Reject operation and return a reason.
- Allow operation and keep original content.
- Allow operation and return modified content.

Interface

HTTP path can be configured for each manage plugin in frps. We'll assume for this example that it's `/handler`.

A request to the RPC server will look like:

```
POST /handler?version=0.1.0&op=Login
{
    "version": "0.1.0",
    "op": "Login",
    "content": {
        ... // Operation info
    }
}

Request Header:
X-Frp-Reqid: for tracing
```

The response can look like any of the following:

- Non-200 HTTP response status code (this will automatically tell frps that the request should fail)
- Reject operation:

```
{
    "reject": true,
    "reject_reason": "invalid user"
}
```

- Allow operation and keep original content:

```
{
    "reject": false,
    "unchange": true
}
```

- Allow operation and modify content

```
{
    "unchange": "false",
    "content": {
        ... // Replaced content
    }
}
```

Operation

Currently `Login`, `NewProxy`, `CloseProxy`, `Ping`, `NewWorkConn` and `NewUserConn` operations are supported.

Login Client login operation

```
{
    "content": {
        "version": <string>,
        "hostname": <string>,
        "os": <string>,
        "arch": <string>,
        "user": <string>,
        "timestamp": <int64>,
        "privilege_key": <string>,
        "run_id": <string>,
        "pool_count": <int>,
        "metas": map<string>string,
        "client_address": <string>
    }
}
```

NewProxy Create new proxy

```
{
    "content": {
        "user": {
            "user": <string>,
            "metas": map<string>string
            "run_id": <string>
        },
        "proxy_name": <string>,
        "proxy_type": <string>,
```

```
        "use_encryption": <bool>,
        "use_compression": <bool>,
        "group": <string>,
        "group_key": <string>,

        // tcp and udp only
        "remote_port": <int>,

        // http and https only
        "custom_domains": []<string>,
        "subdomain": <string>,
        "locations": <string>,
        "http_user": <string>,
        "http_pwd": <string>,
        "host_header_rewrite": <string>,
        "headers": map<string>string,

        // stcp only
        "sk": <string>,

        // tcpmux only
        "multiplexer": <string>

        "metas": map<string>string
    }
}
```

CloseProxy A previously created proxy is closed.

Please note that one request will be sent for every proxy that is closed, do **NOT** use this if you have too many proxies bound to a single client, as this may exhaust the server's resources.

```
{
    "content": {
        "user": {
            "user": <string>,
            "metas": map<string>string
            "run_id": <string>
        },
        "proxy_name": <string>
    }
}
```

Ping Heartbeat from frpc

```
{
```

```
    "content": {
        "user": {
            "user": <string>,
            "metas": map<string>string
            "run_id": <string>
        },
        "timestamp": <int64>,
        "privilege_key": <string>
    }
}
```

NewWorkConn New work connection received from frpc (RPC sent after `run_id` is matched with an existing frp connection)

```
{
    "content": {
        "user": {
            "user": <string>,
            "metas": map<string>string
            "run_id": <string>
        },
        "run_id": <string>
        "timestamp": <int64>,
        "privilege_key": <string>
    }
}
```

NewUserConn New user connection received from proxy (support `tcp`, `stcp`, `https` and `tcpmux`) .

```
{
    "content": {
        "user": {
            "user": <string>,
            "metas": map<string>string
            "run_id": <string>
        },
        "proxy_name": <string>,
        "proxy_type": <string>,
        "remote_addr": <string>
    }
}
```

Server Plugin Configuration

```
# frps.ini
[common]
```

```
bind_port = 7000

[plugin.user-manager]
addr = 127.0.0.1:9000
path = /handler
ops = Login

[plugin.port-manager]
addr = 127.0.0.1:9001
path = /handler
ops = NewProxy
```

- addr: the address where the external RPC service listens. Defaults to http. For https, specify the schema: `addr = https://127.0.0.1:9001`.
- path: http request url path for the POST request.
- ops: operations plugin needs to handle (e.g. "Login", "NewProxy", ...).
- tls_verify: When the schema is https, we verify by default. Set this value to false if you want to skip verification.

Metadata

Metadata will be sent to the server plugin in each RPC request.

There are 2 types of metadata entries - 1 under `[common]` and the other under each proxy configuration. Metadata entries under `[common]` will be sent in `Login` under the key `metas`, and in any other RPC request under `user.metas`. Metadata entries under each proxy configuration will be sent in `NewProxy` op only, under `metas`.

Metadata entries start with `meta_`. This is an example of metadata entries in `[common]` and under the proxy named `[ssh]`:

```
# frpc.ini
[common]
server_addr = 127.0.0.1
server_port = 7000
user = fake
meta_token = fake
meta_version = 1.0.0

[ssh]
type = tcp
local_port = 22
remote_port = 6000
meta_id = 123
```