onStarting()

Error?

initAddOns()

Error?

handleGracefulShutdown()

Yes

No

Yes

No

onError(error)

onError(error)

onStarted()

Receive SIGTERM?

Yes

Error?

Yes

No

onError(error)

registerDependencies ()

attachConfigProvider()

setTimeout(  
MAX\_STOP\_TIMEOUT)

Error?

onStopping()

Yes

sendDeadLetters()

disposeAddOns()

onStopped()

onError(error)

exitNow?

Yes

Timeout?

check(  
MAX\_STOP\_TIMEOUT)

No

Yes

No

UserController

UserService

UserRepository

@inject(USR\_SVC)

@inject(USR\_REP)

resolve

resolve

resolve

resolve

Profile Service

Account Service

Auth Service

Product Service

Order Service

Profile Service

Account Service

Auth Service

Product Service

Order Service

**Message**

**Broker**

`

function **add**(a, b) {

return a + b

}

add(2, 5)

local call

**Process**

function **add**(a, b) {

return RPC.call(  
 ‘add’, a, b)

}

RPC  
caller

function **add**(a, b) {

return a + b

}

add(2, 5)

RPC handler

local call

remote call

**Process A**

**Process B**

interface **IUserProvider**

class **RemoteUserProvider**

implements **IUserProvider**

@controller()  
class UserController

@inject() *provider*: **IUserProvider**

*provider*.getList(pageIndex, pageSize)

local call

**Micro Fleet service A**

inject

RPC caller

interface **IUserProvider**

class **UserProvider**

implements **IUserProvider**

**// Real implementation here**

@directController()  
@mediateController()  
class UserController

@inject() *provider*: **IUserProvider**

*provider*.getList(pageIndex, pageSize)

local call

**Micro Fleet service B**

inject

RPC

Handler

local call

Remote communication (direct / mediate)

Client side

HTTP request

**Server side**