

Se mocker (enfin) Du Hardware

Vincent Pollet, Pierre Aubert



Unit tests
a philosophy and a help
face to its own software

Feedback on 13 years of personal
practice

LAPP / CNRS / ANNECY – 19/01/2024

Sébastien Valat

1

Sébastien Valat

Unit tests
a philosophy and a help
face to its own software

Feedback on 13 years of personal
practice

LAPP / CNRS / ANNECY – 19/01/2024

Sébastien Valat

1

Sébastien Valat

Seminar on Unit Tests

Possibility to **Mock network**

Unit tests
a philosophy and a help
face to its own software

Feedback on 13 years of personal
practice

LAPP / CNRS / ANNECY – 19/01/2024

Sébastien Valat

1

Sébastien Valat

Seminar on Unit Tests

Possibility to **Mock network**

Mock **LHCb DAq** on a **laptop**

Unit tests
a philosophy and a help
face to its own software

Feedback on 13 years of personal
practice

LAPP / CNRS / ANNECY – 19/01/2024

Sébastien Valat

1

Sébastien Valat

Seminar on Unit Tests

Possibility to **Mock network**

Mock **LHCb DAq** on a **laptop**

We could **mock only
server interactions**

Unit tests
a philosophy and a help
face to its own software

Feedback on 13 years of personal
practice

LAPP / CNRS / ANNECY – 19/01/2024

Sébastien Valat

1

Sébastien Valat

Seminar on Unit Tests

Possibility to **Mock network**

Mock **LHCb DAq** on a **laptop**

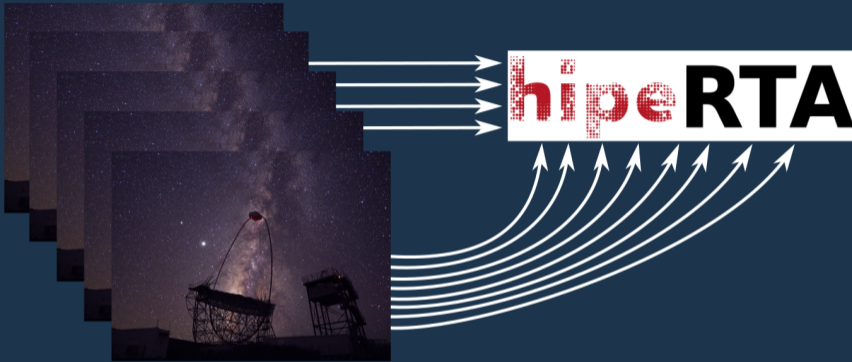
We could **mock only
server interactions**

No implementation of a
mock server **needed**

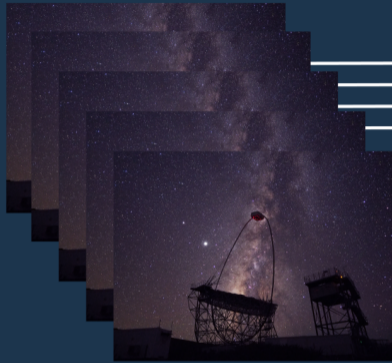




hipeRTA



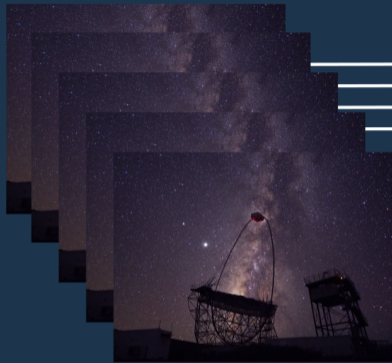
x19 telescopes North Site
x100 Telescopes South Site



hipe RTA

How To **Unit Test** It ?

x19 telescopes North Site
x100 Telescopes South Site



x19 telescopes North Site
x100 Telescopes South Site

hipe RTA

How To **Unit Test** It ?

- On our **laptops**
- On **Gitlab CI**

Mock of Sockets ?

Camera

Mock of Sockets ?



Mock of Sockets ?



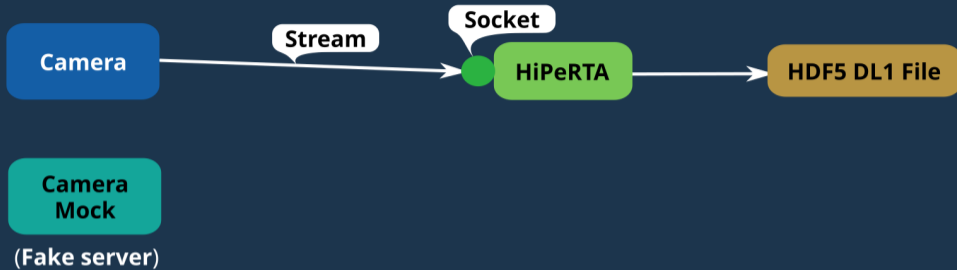
Mock of Sockets ?



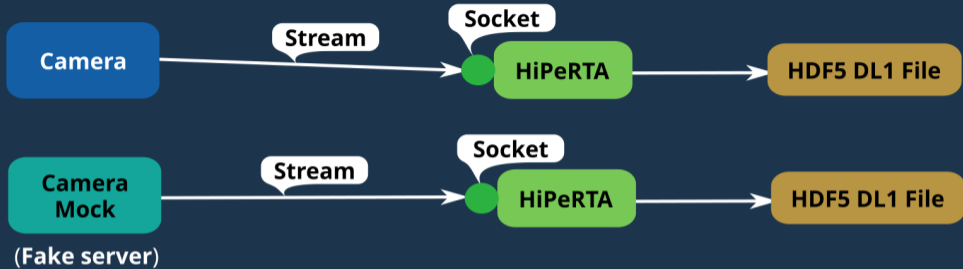
Mock of Sockets ?



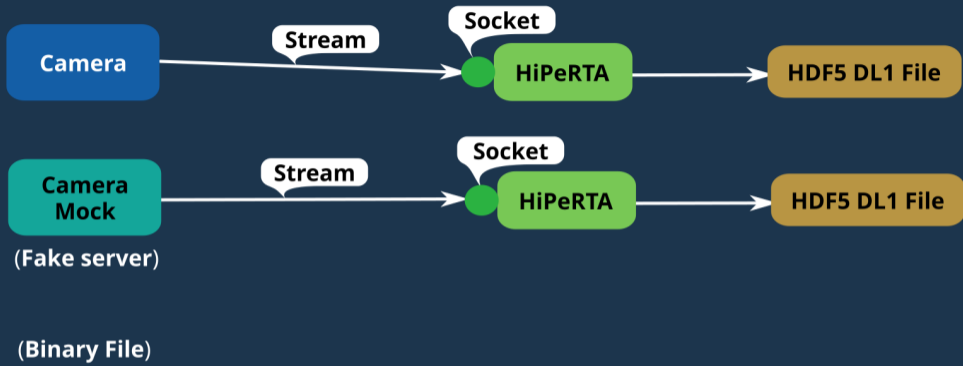
Mock of Sockets ?



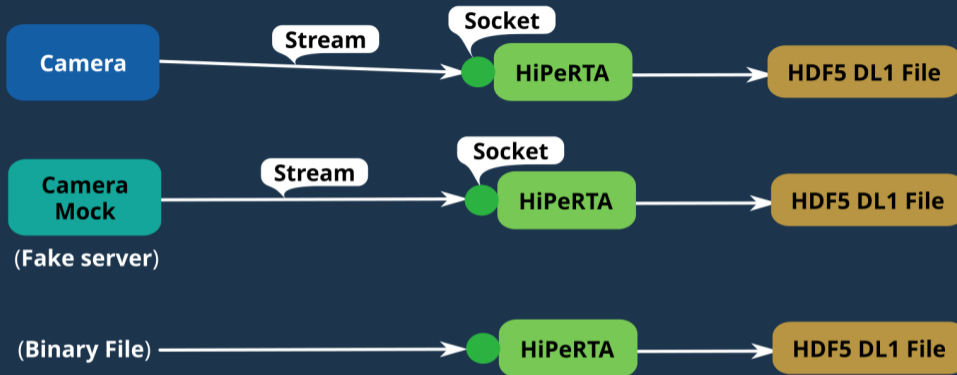
Mock of Sockets ?



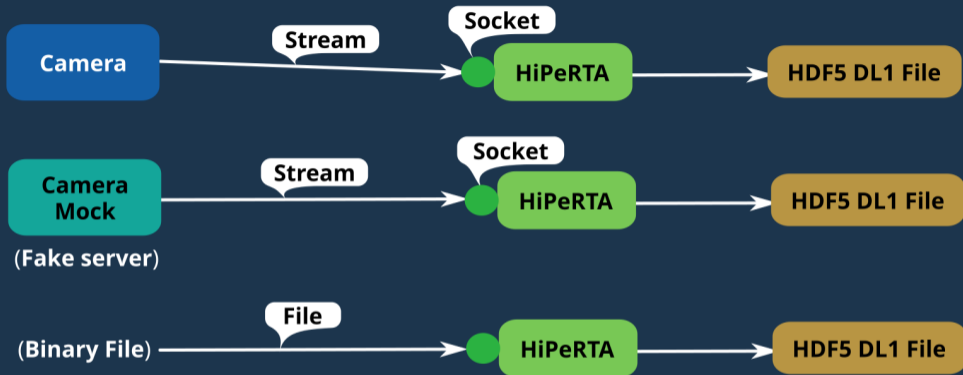
Mock of Sockets ?



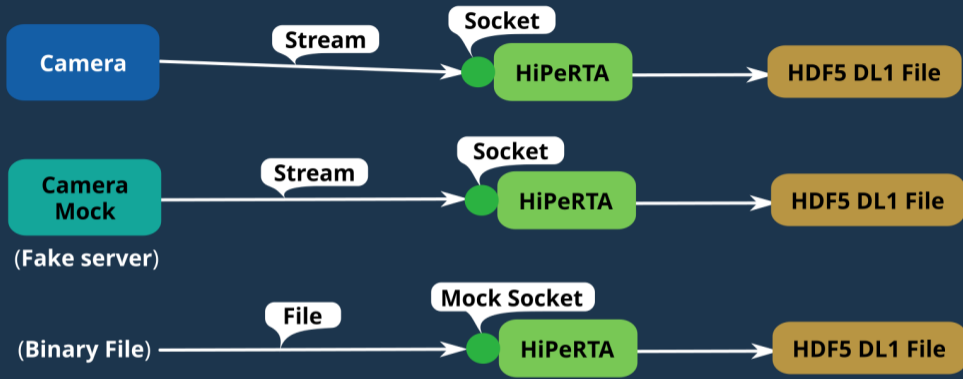
Mock of Sockets ?



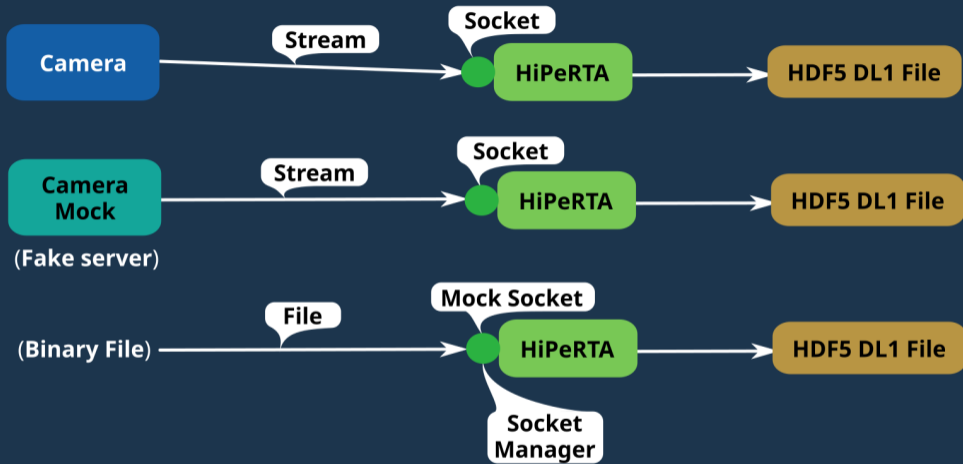
Mock of Sockets ?



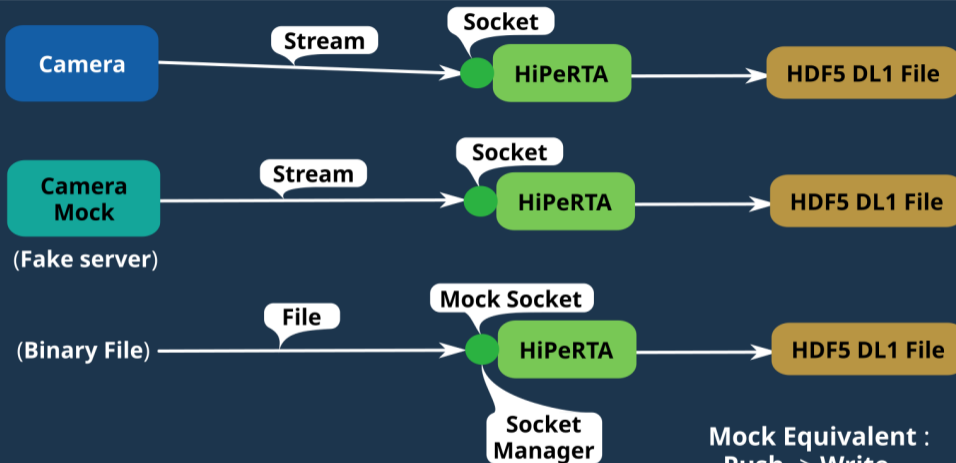
Mock of Sockets ?



Mock of Sockets ?

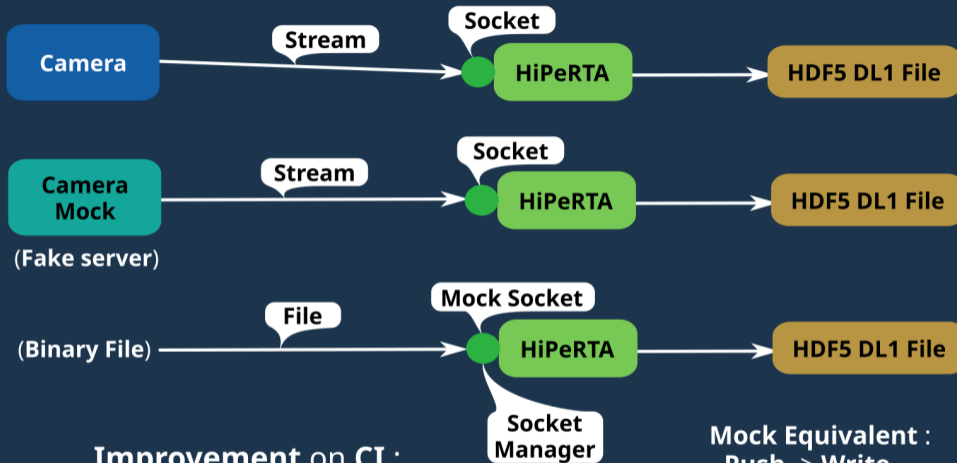


Mock of Sockets ?



Mock Equivalent :
 - Push -> Write
 - Pull -> Read

Mock of Sockets ?

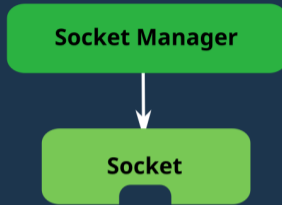


Improvement on CI :
 - Unit tests use to take **40s**
 - Now only **5s**

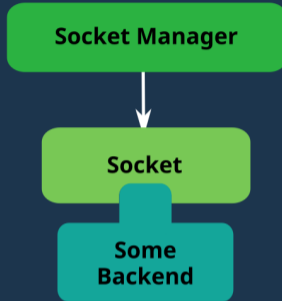
Mock Equivalent :
 - Push -> Write
 - Pull -> Read

Socket Manager

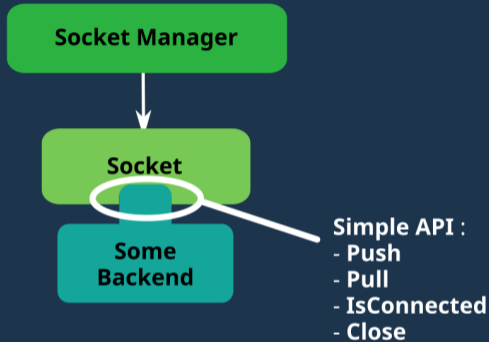
Mock Sockets Design



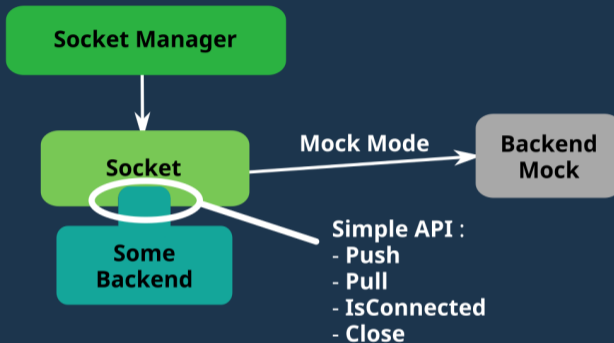
Mock Sockets Design



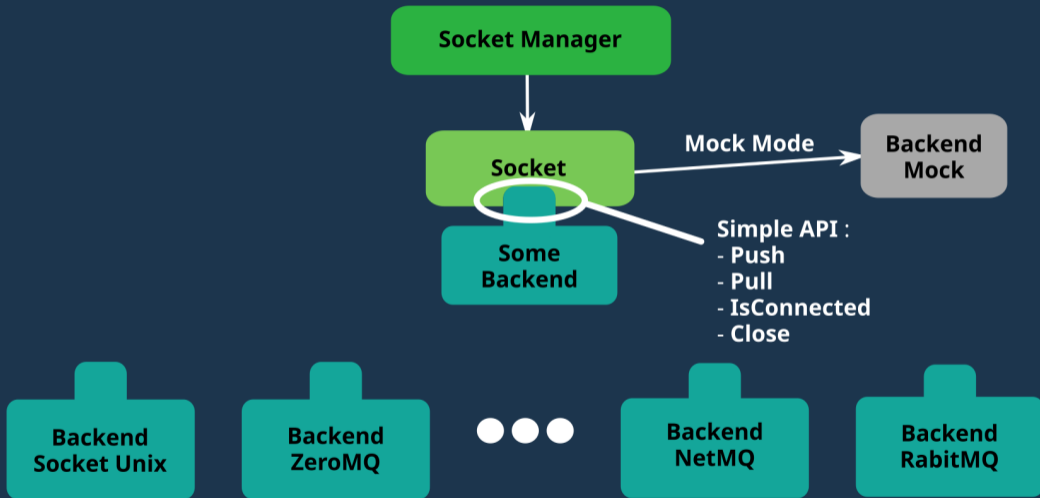
Mock Sockets Design



Mock Sockets Design

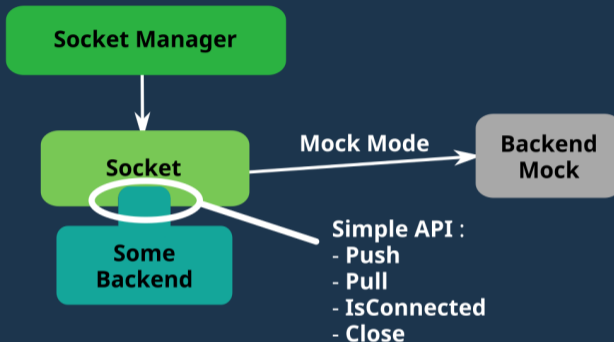


Mock Sockets Design



Mock Sockets Design

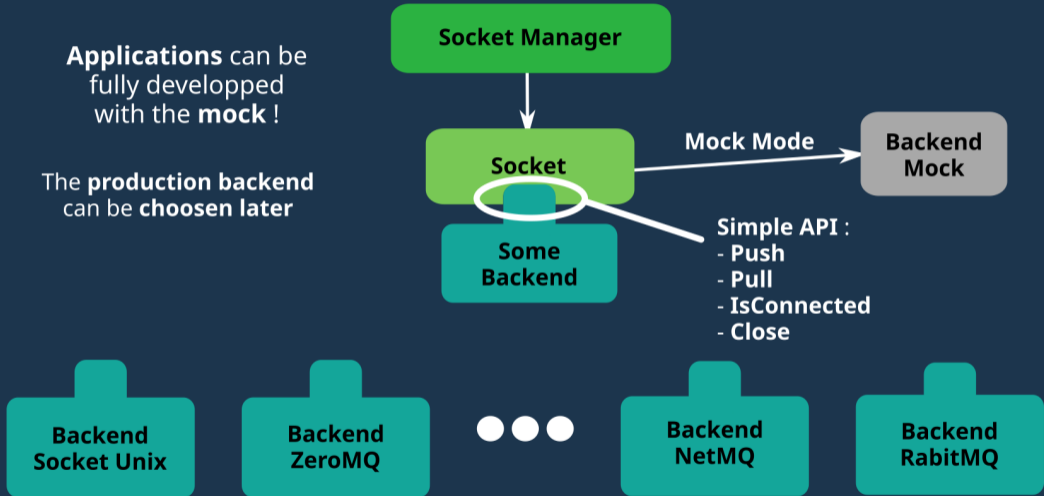
Applications can be fully developed with the **mock** !



Mock Sockets Design

Applications can be fully developed with the **mock** !

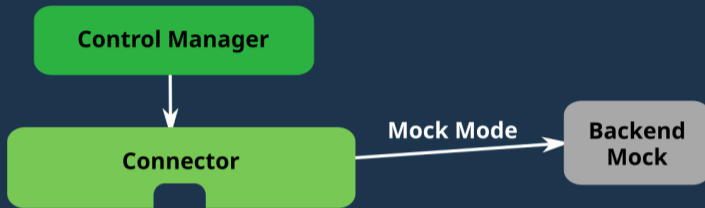
The **production backend** can be **chosen later**



We could make **Sockets**
more generic

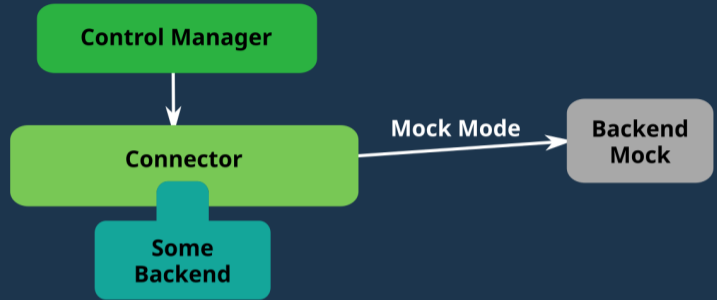
Extending Socket Backend

We could make **Sockets**
more generic



Extending Socket Backend

We could make **Sockets**
more generic

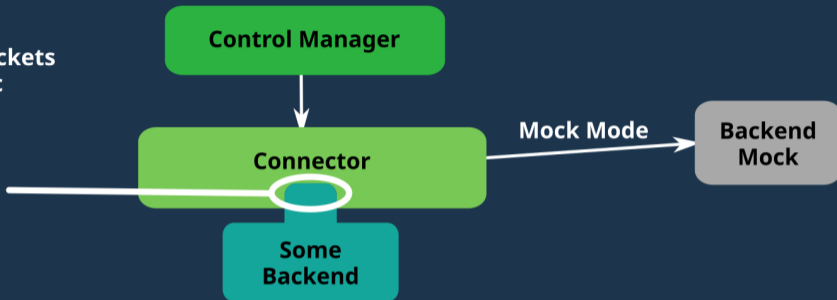


Extending Socket Backend

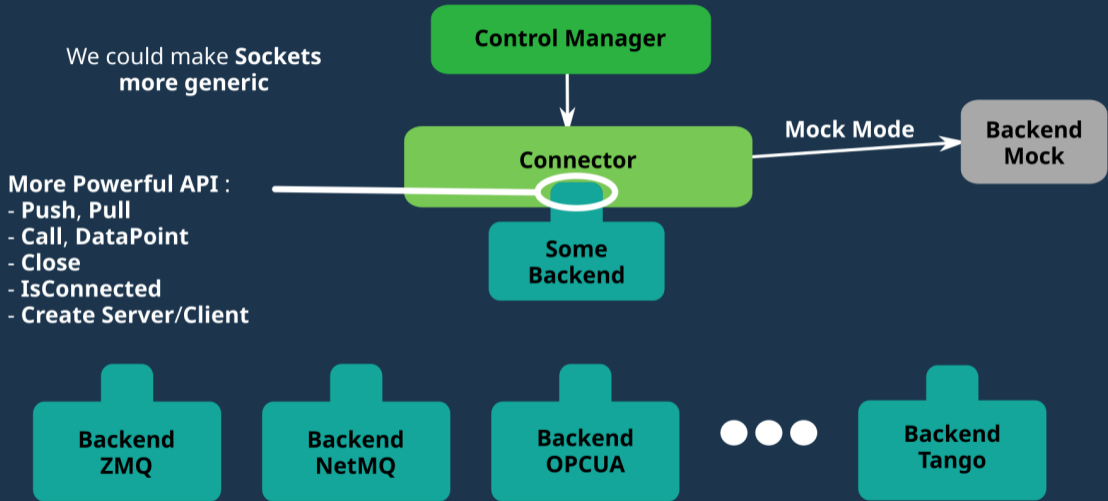
We could make **Sockets**
more generic

More Powerful API :

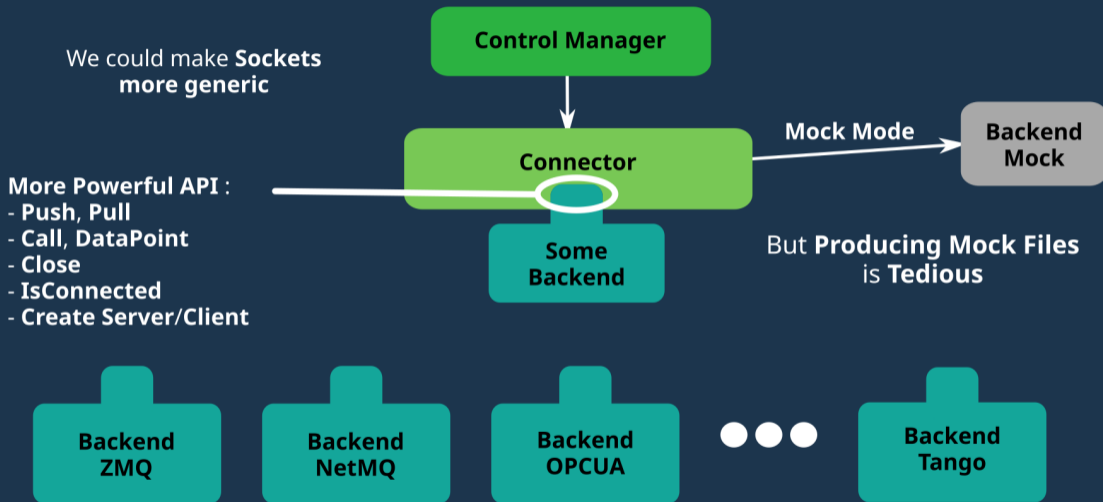
- Push, Pull
- Call, DataPoint
- Close
- IsConnected
- Create Server/Client



Extending Socket Backend

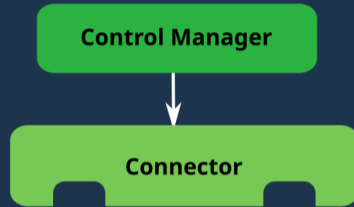


Extending Socket Backend

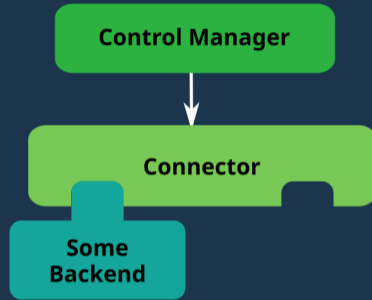


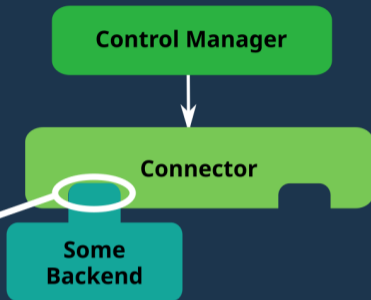
Control Manager

Mock Control Design



Mock Control Design

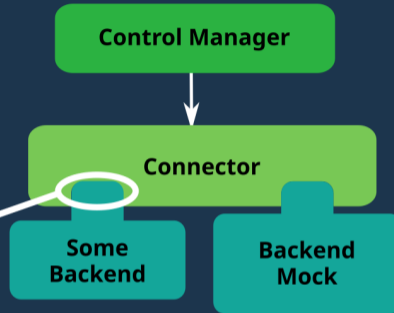




More Powerful API :

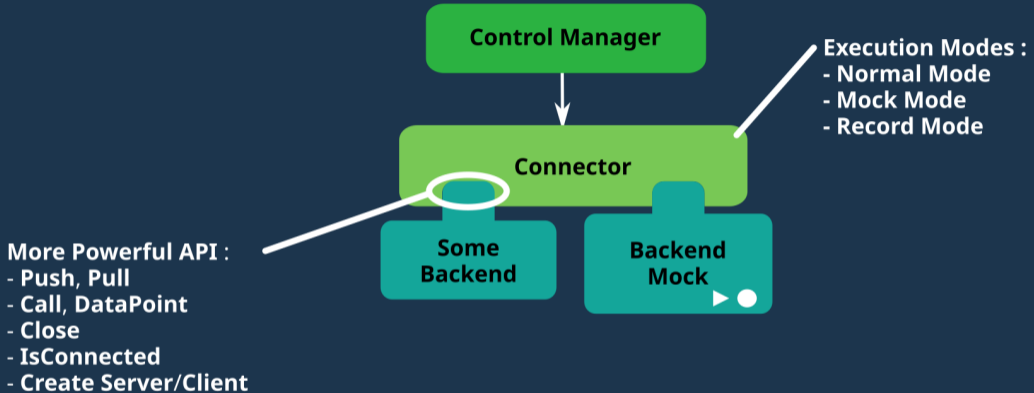
- Push, Pull
- Call, DataPoint
- Close
- IsConnected
- Create Server/Client

Mock Control Design

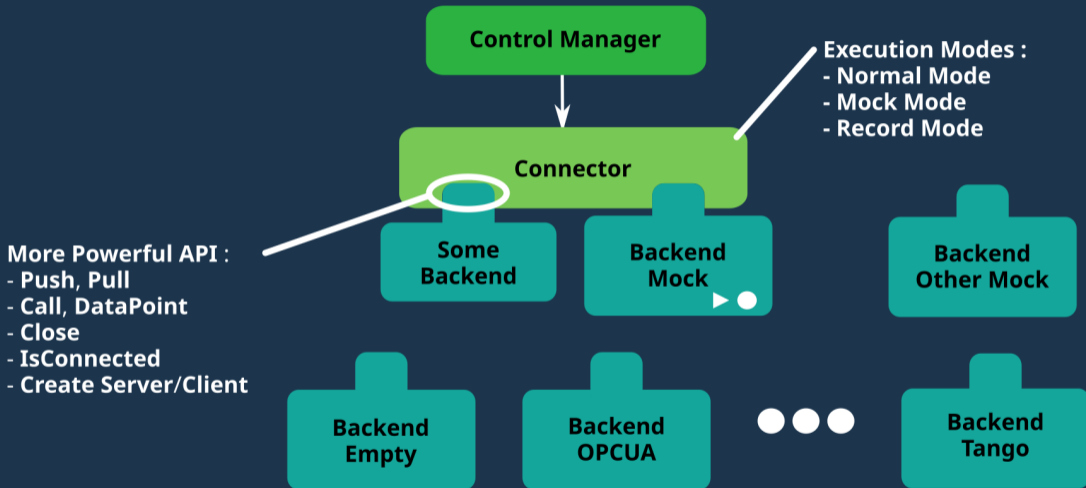


- More Powerful API :**
- Push, Pull
 - Call, DataPoint
 - Close
 - IsConnected
 - Create Server/Client

Mock Control Design



Mock Control Design



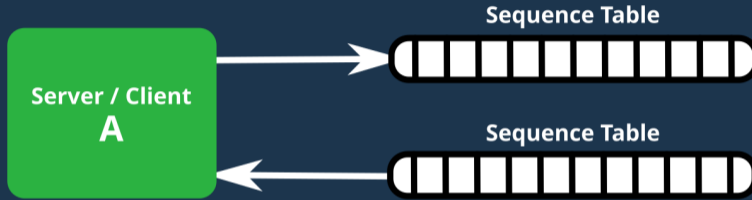
Normal Use



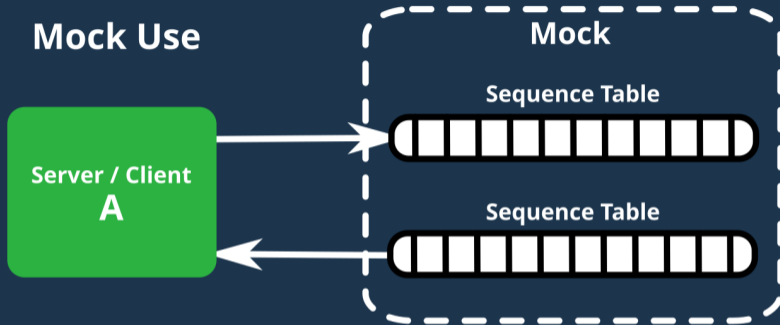
Normal Use



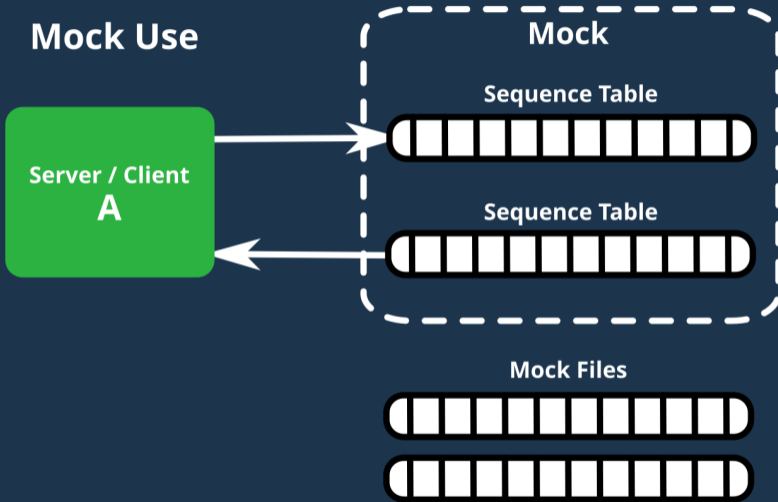
Mock Use



Normal Use and Mock

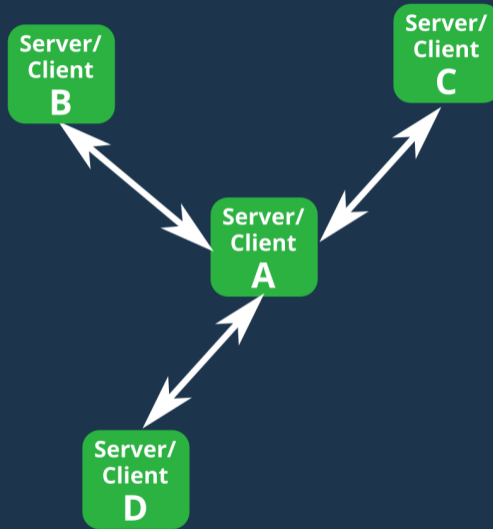


Normal Use and Mock



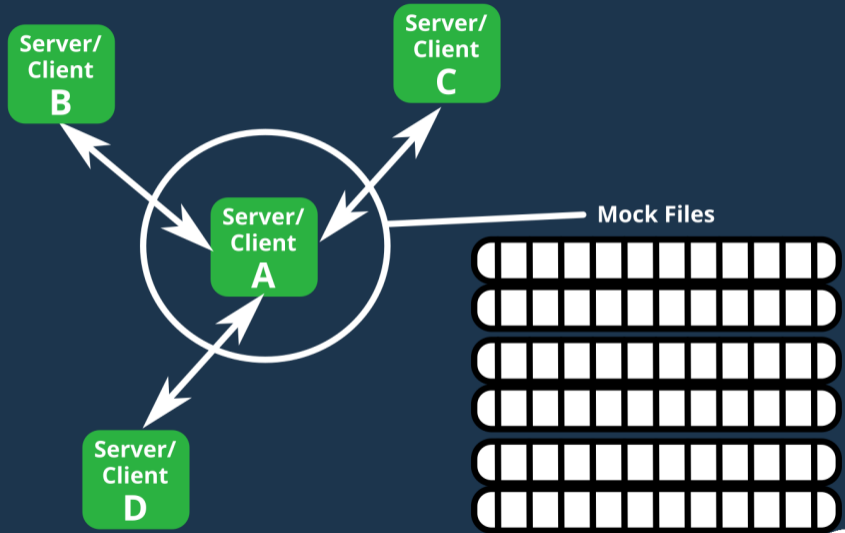
Normal Use and Mock

Normal Use



Normal Use and Mock

Mock Record

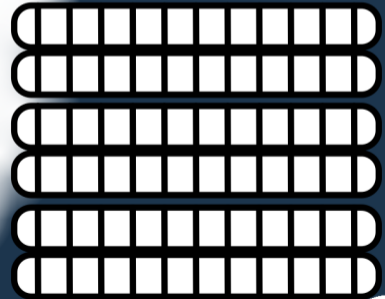


Normal Use and Mock

Mock Use

Server/
Client
A

Mock Files



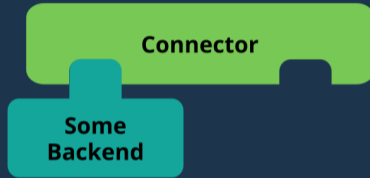
Sequence Table of A

Event Id and/or Time	State of A	Client/Sever : B			Client/Sever : C	
		Data Point B1	Calls From B of A1(X) Params	Expected Result	Calls From C of A2(X) Params	Expected Result
0	Waiting	-	-	-	-	-
1	-	B1 value	-	-	-	-
2	-	-	42	21	-	-
3	Ready	-	-	-	[63, 7]	[48, 12, 4]
28	-	-	-	-	-	-
56	Stop	-	-	-	-	-

Double Backend Design



Double Backend Design

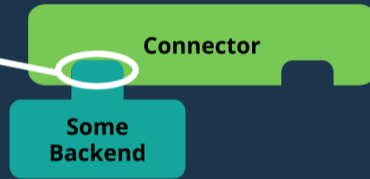


Double Backend Design

Backend API

Example :

- Push, Pull
- Call, DataPoint
- Close
- IsConnected
- Create Server/Client

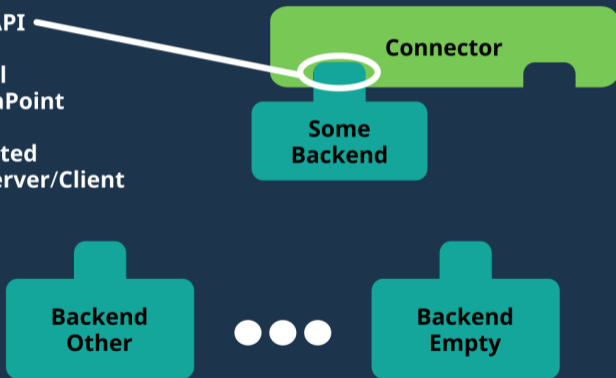


Double Backend Design

Backend API

Example :

- Push, Pull
- Call, DataPoint
- Close
- IsConnected
- Create Server/Client

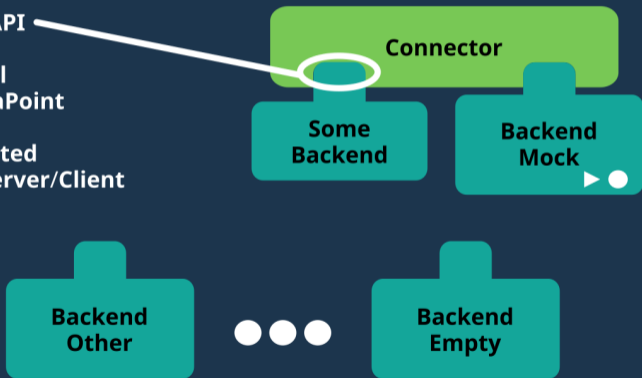


Double Backend Design

Backend API

Example :

- Push, Pull
- Call, DataPoint
- Close
- IsConnected
- Create Server/Client

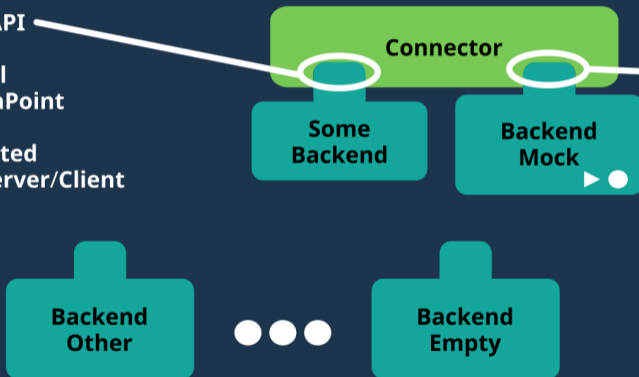


Double Backend Design

Backend API

Example :

- Push, Pull
- Call, DataPoint
- Close
- IsConnected
- Create Server/Client



Backend API +

Common Mock API :

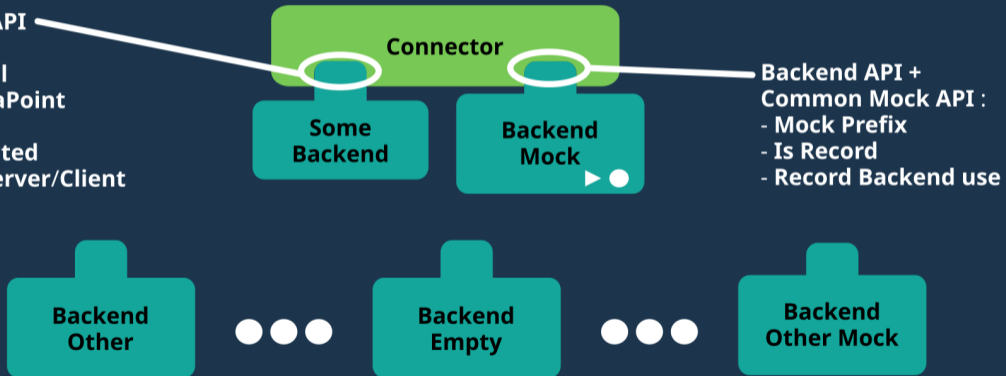
- Mock Prefix
- Is Record
- Record Backend use

Double Backend Design

Backend API

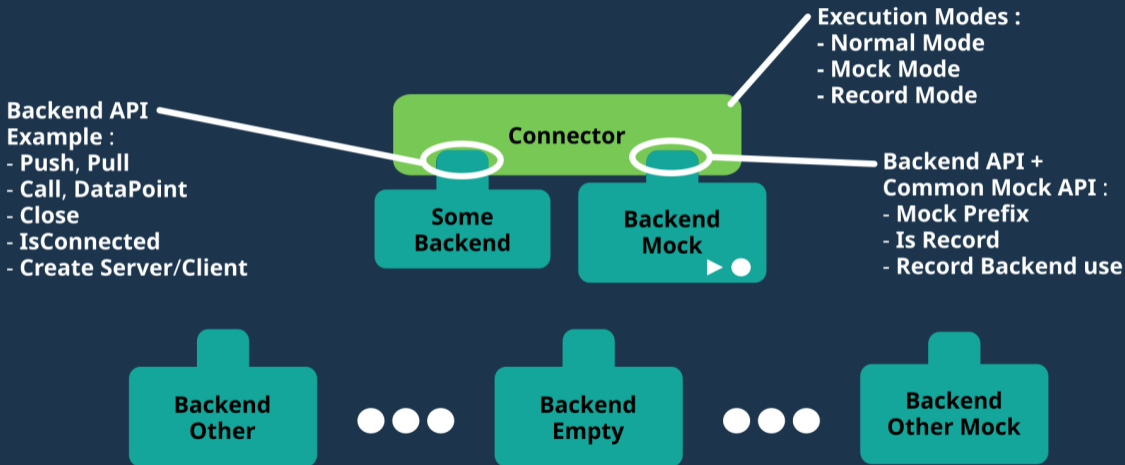
Example :

- Push, Pull
- Call, DataPoint
- Close
- IsConnected
- Create Server/Client

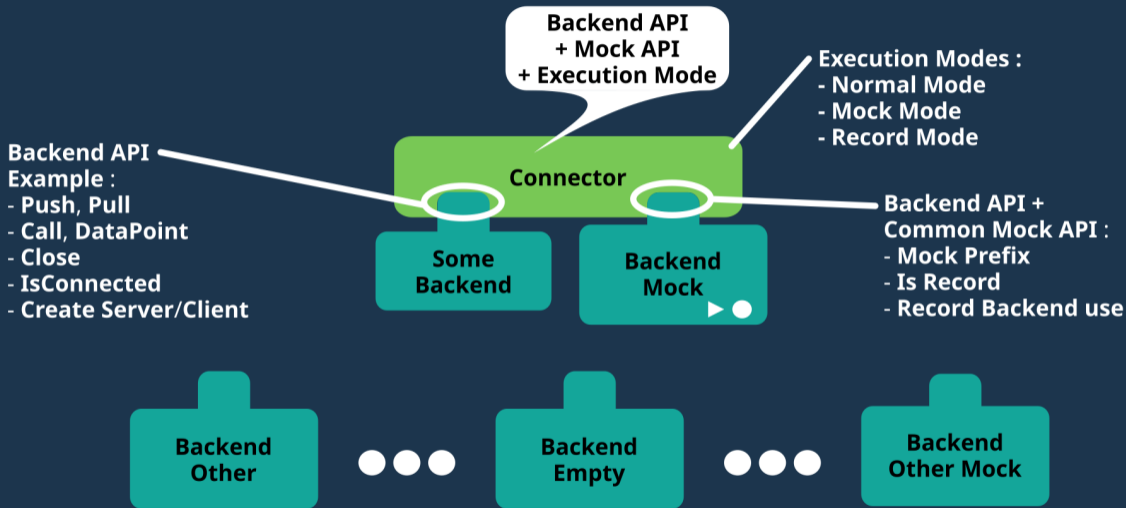


- Backend API +
Common Mock API :
- Mock Prefix
 - Is Record
 - Record Backend use

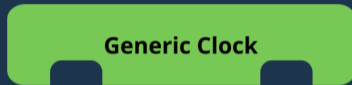
Double Backend Design



Double Backend Design



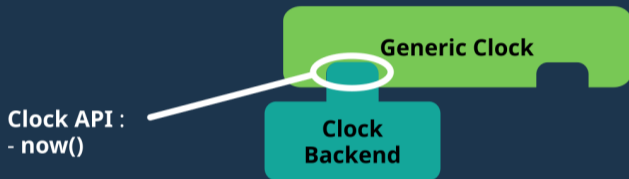
Double Backend Clock



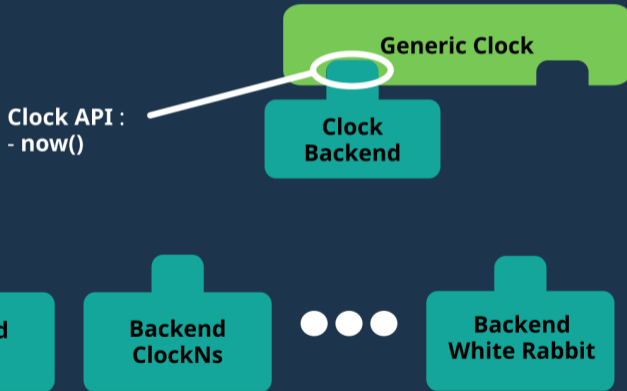
Double Backend Clock



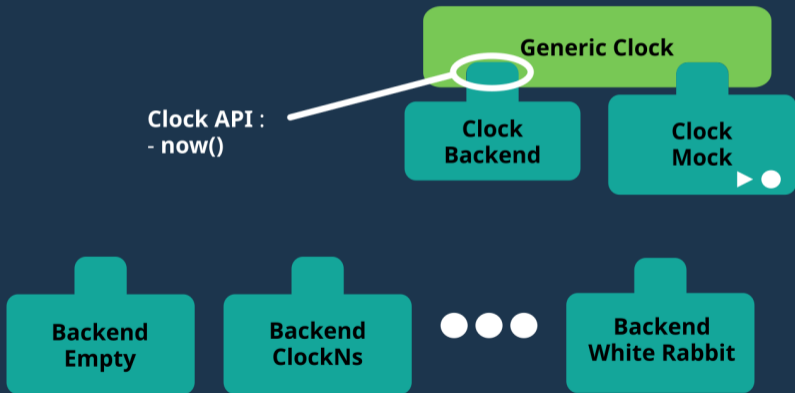
Double Backend Clock



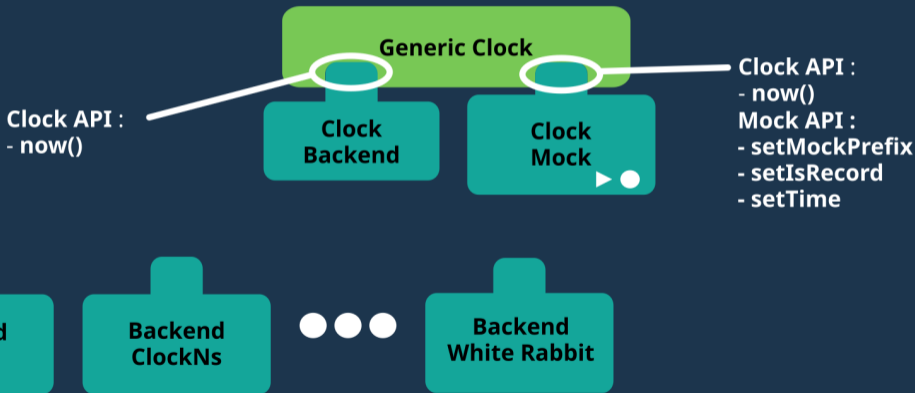
Double Backend Clock



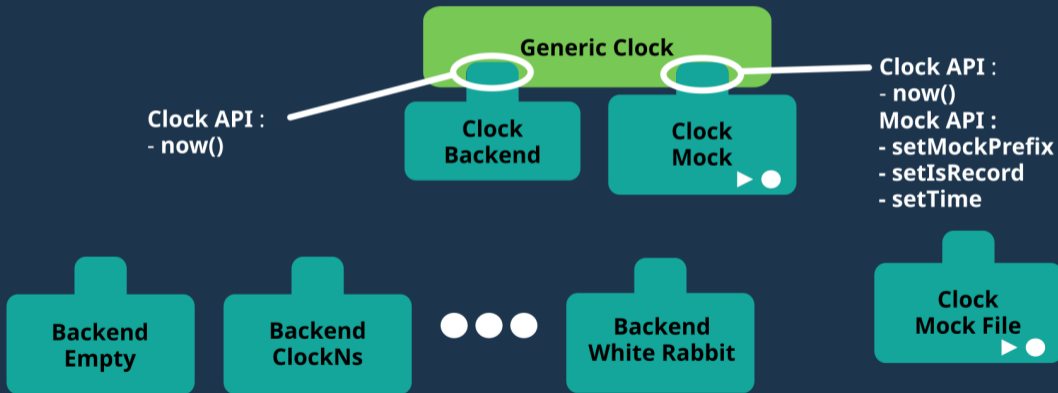
Double Backend Clock



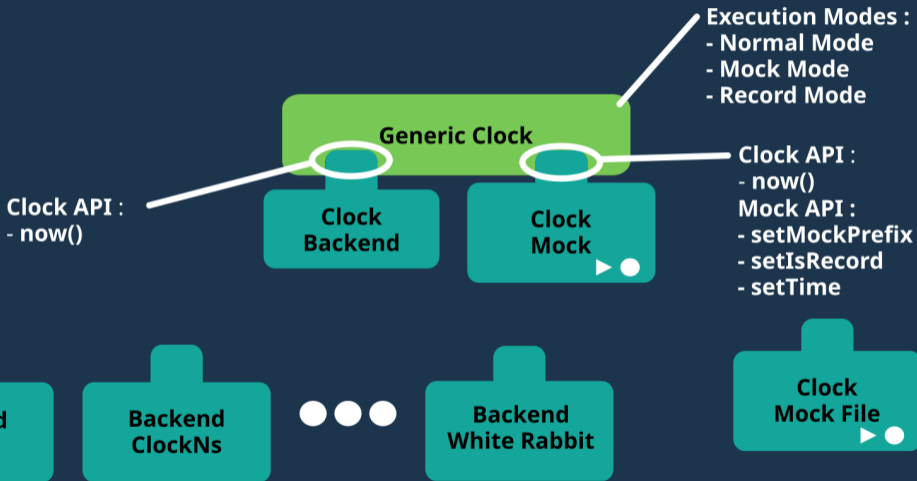
Double Backend Clock



Double Backend Clock



Double Backend Clock



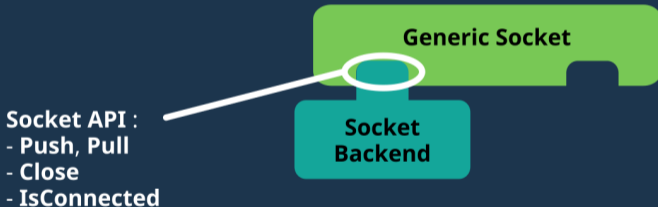


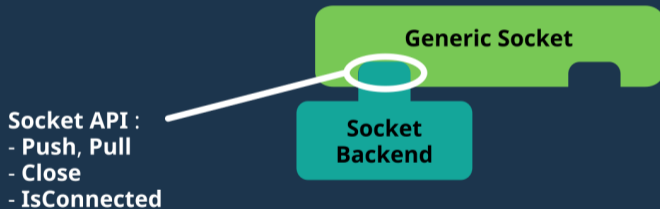
Generic Socket

Double Backend Socket



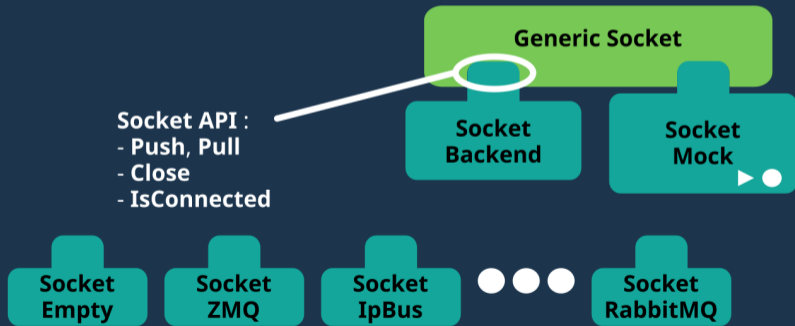
Double Backend Socket





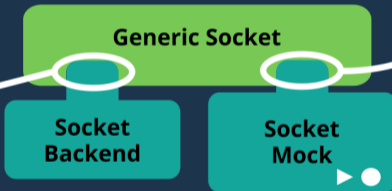
Double Backend Socket

Socket API :
- Push, Pull
- Close
- IsConnected



Double Backend Socket

Socket API :
- Push, Pull
- Close
- IsConnected



Socket API +
Mock API :
- setMockPrefix
- setIsRecord
- setTime

Socket
Empty

Socket
ZMQ

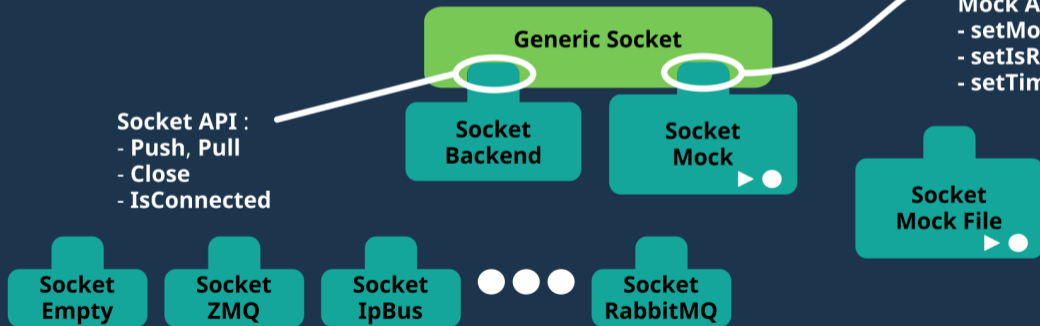
Socket
IpBus

Socket
RabbitMQ

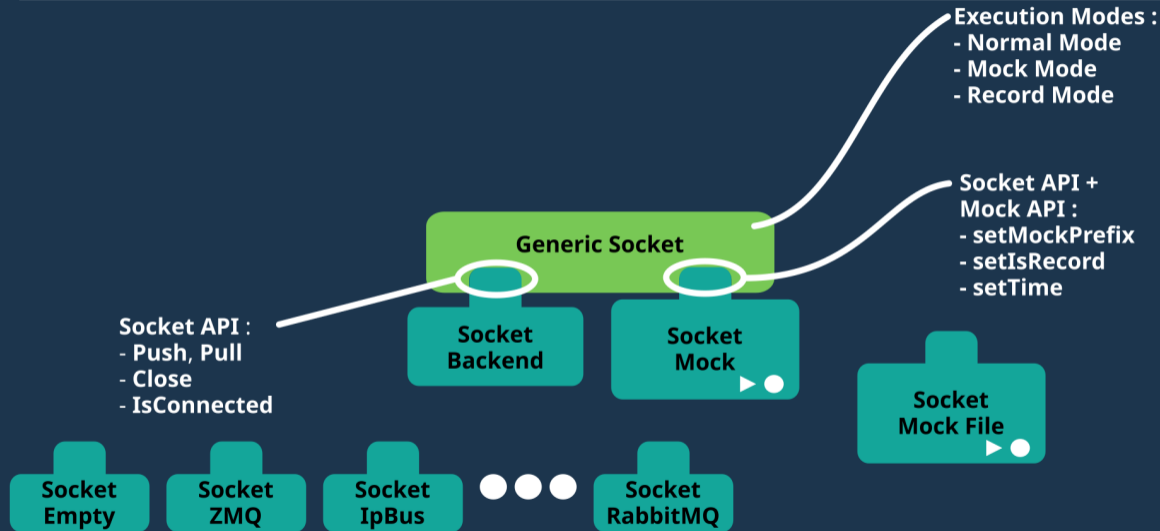
Double Backend Socket

Socket API :
- Push, Pull
- Close
- IsConnected

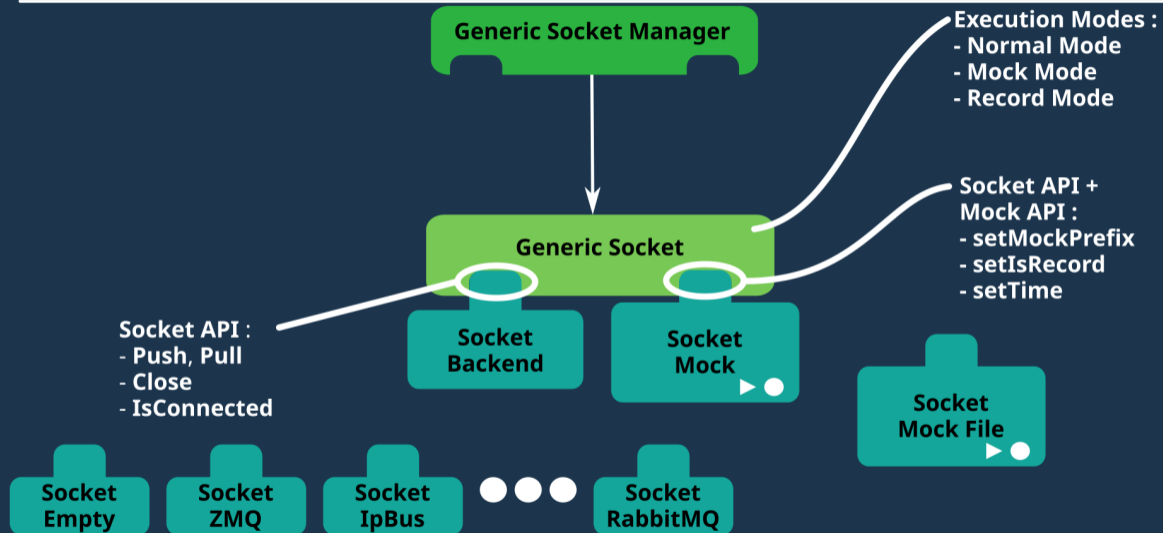
Socket API +
Mock API :
- setMockPrefix
- setIsRecord
- setTime



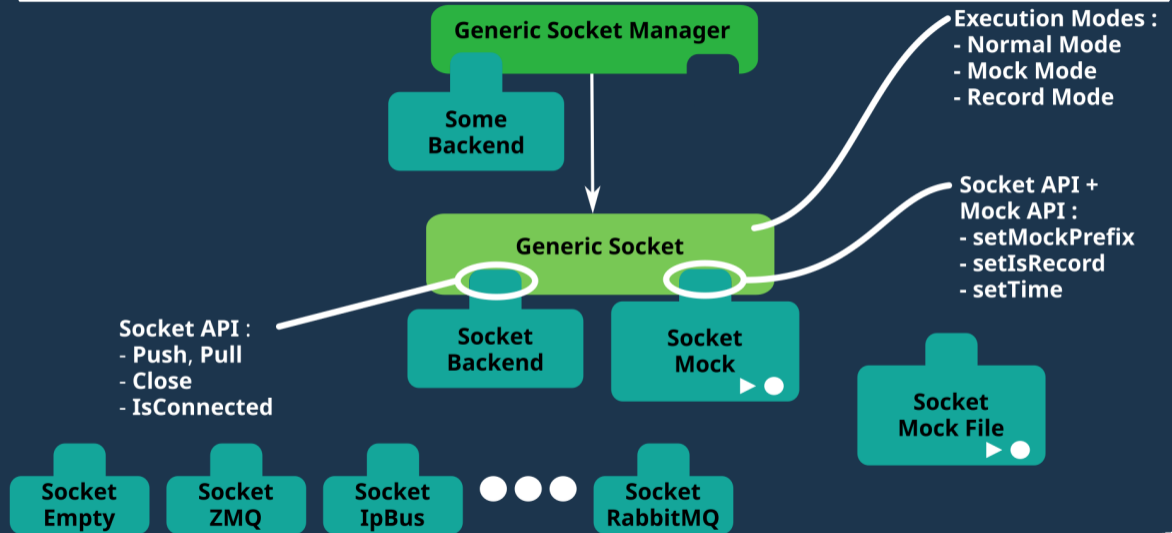
Double Backend Socket



Double Backend Socket



Double Backend Socket



Execution Modes :

- Normal Mode
- Mock Mode
- Record Mode

Socket API + Mock API :

- `setMockPrefix`
- `setIsRecord`
- `setTime`

Socket API :

- `Push`, `Pull`
- `Close`
- `IsConnected`

Double Backend Socket

Backend API :
- Create Server/Client

Generic Socket Manager

Some Backend

Execution Modes :
- Normal Mode
- Mock Mode
- Record Mode

Generic Socket

Socket API + Mock API :
- setMockPrefix
- setIsRecord
- setTime

Socket API :
- Push, Pull
- Close
- IsConnected

Socket Backend

Socket Mock

Socket Mock File

Socket Empty

Socket ZMQ

Socket IpBus

Socket RabbitMQ



Double Backend Socket

Backend API :
- Create Server/Client

Generic Socket Manager

Execution Modes :

- Normal Mode
- Mock Mode
- Record Mode

Backend Empty

Backend ZMQ

Some Backend

Backend IpBus

Backend RabbitMQ

Generic Socket

Socket API + Mock API :

- setMockPrefix
- setIsRecord
- setTime

Socket API :
- Push, Pull
- Close
- IsConnected

Socket Backend

Socket Mock

Socket Mock File

Socket Empty

Socket ZMQ

Socket IpBus

Socket RabbitMQ

Double Backend Socket

Backend API :
- Create Server/Client

Backend
Empty

Backend
ZMQ

Some
Backend

Backend
Mock

Backend
IpBus

Backend
RabbitMQ

Socket
Backend

Socket
Mock

Socket
Mock
File

Socket
Empty

Socket
ZMQ

Socket
IpBus

Socket
RabbitMQ

Generic Socket Manager

Generic Socket

Execution Modes :

- Normal Mode
- Mock Mode
- Record Mode

**Socket API +
Mock API :**

- setMockPrefix
- setIsRecord
- setTime

Socket API :
- Push, Pull
- Close
- IsConnected

Double Backend Socket

Backend API :
- Create Server/Client

Backend
Empty

Backend
ZMQ

Some
Backend

Backend
Mock

Backend
IpBus

Backend
RabbitMQ

Socket
Backend

Socket
Mock

Socket
Mock
File

Socket
Empty

Socket
ZMQ

Socket
IpBus

Socket
RabbitMQ

Generic Socket

Generic Socket Manager

Execution Modes :

- Normal Mode
- Mock Mode
- Record Mode

**Socket API +
Mock API :**

- setMockPrefix
- setIsRecord
- setTime

Socket API :
- Push, Pull
- Close
- IsConnected



Double Backend Socket

Backend API :
- Create Server/Client

Backend Empty

Backend ZMQ

Some Backend

Backend Mock

Backend IpBus

Backend RabbitMQ

Generic Socket

Socket Backend

Socket Mock

Socket Mock File

Socket Empty

Socket ZMQ

Socket IpBus

Socket RabbitMQ

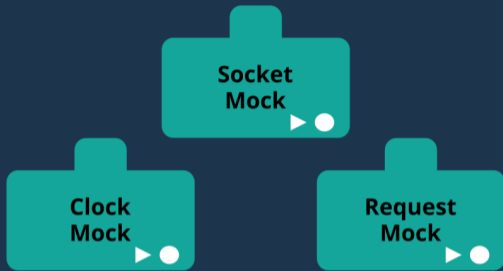
Execution Modes :

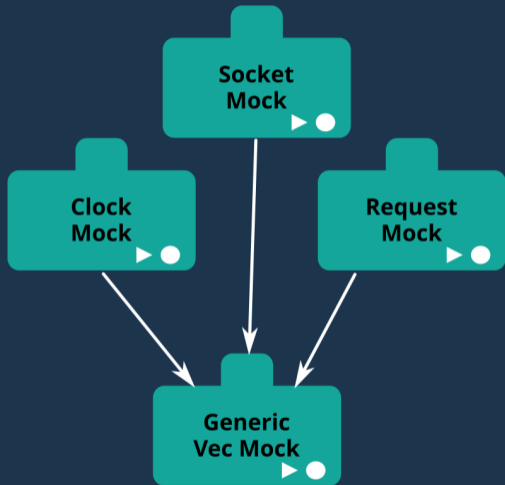
- Normal Mode
- Mock Mode
- Record Mode

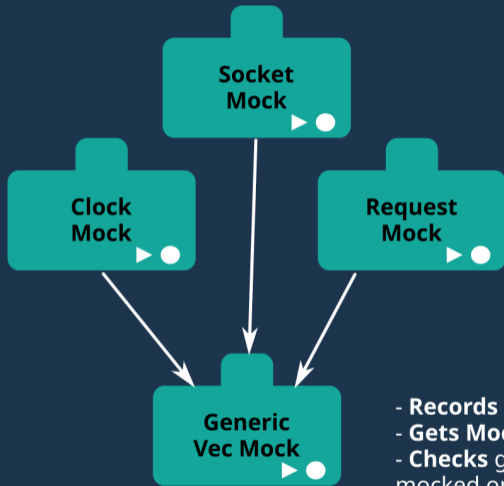
Socket API + Mock API :

- setMockPrefix
- setIsRecord
- setTime

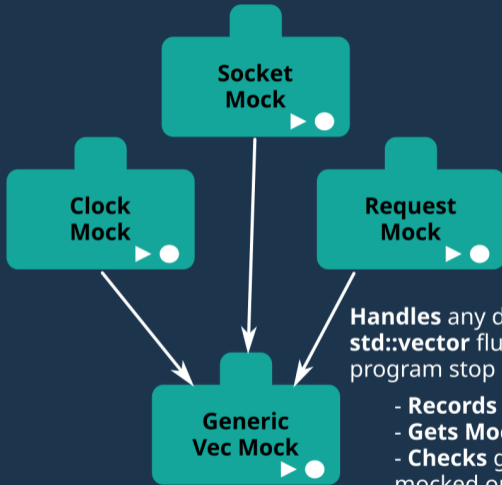
Socket API :
- Push, Pull
- Close
- IsConnected







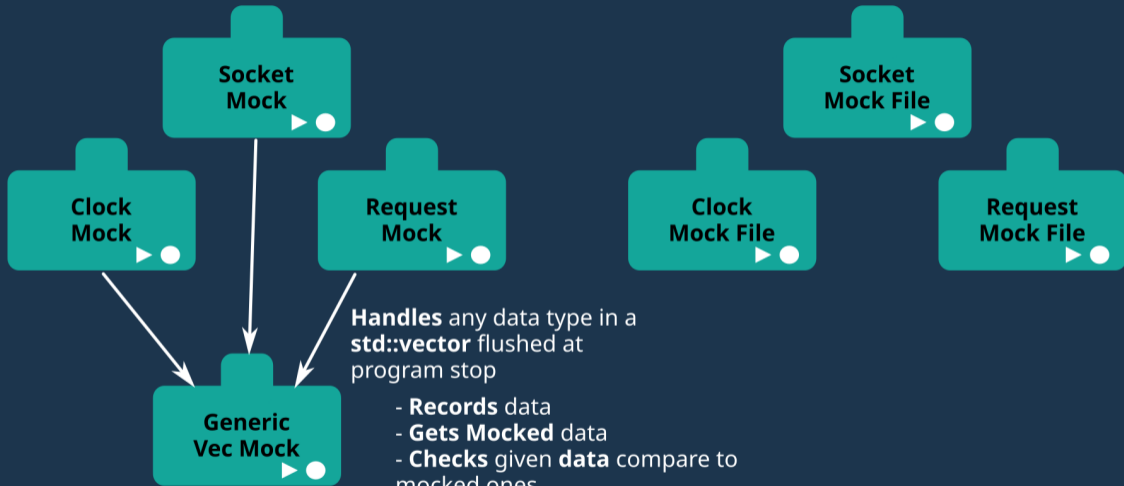
- **Records** data
- **Gets Mocked** data
- **Checks** given **data** compare to mocked ones
- **Human readable feedback** if data are different



Handles any data type in a `std::vector` flushed at program stop

- **Records** data
- **Gets Mocked** data
- **Checks** given **data** compare to mocked ones
- **Human readable feedback** if data are different

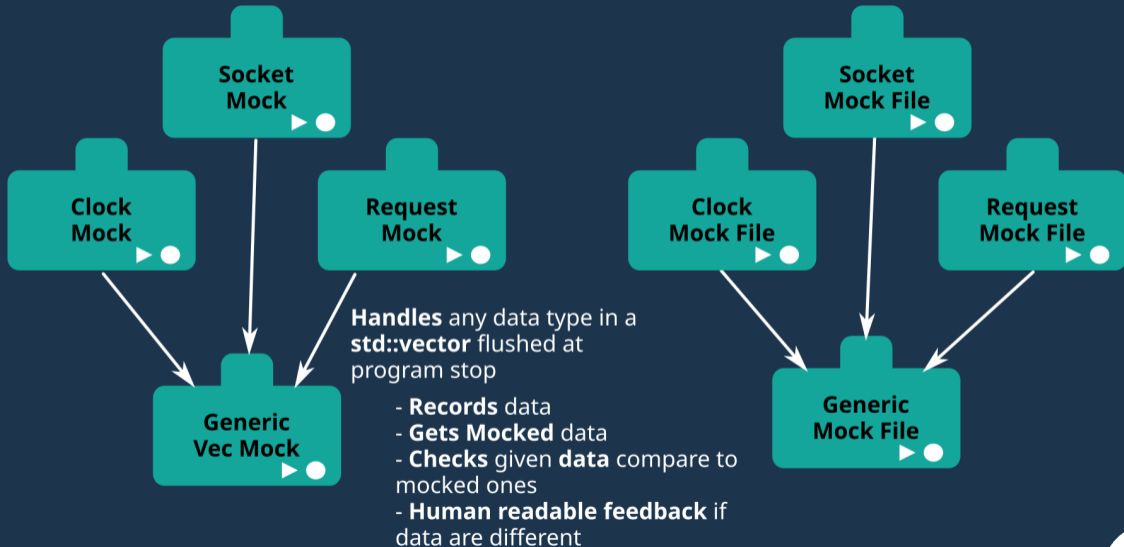
Generic Mock



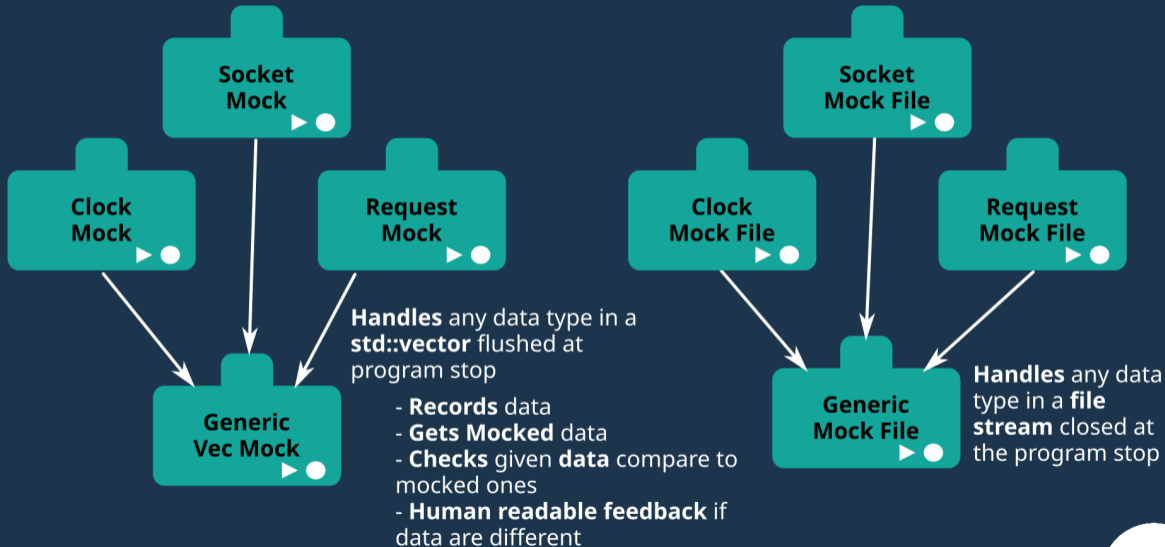
Handles any data type in a `std::vector` flushed at program stop

- **Records** data
- **Gets Mocked** data
- **Checks** given **data** compare to mocked ones
- **Human readable feedback** if data are different

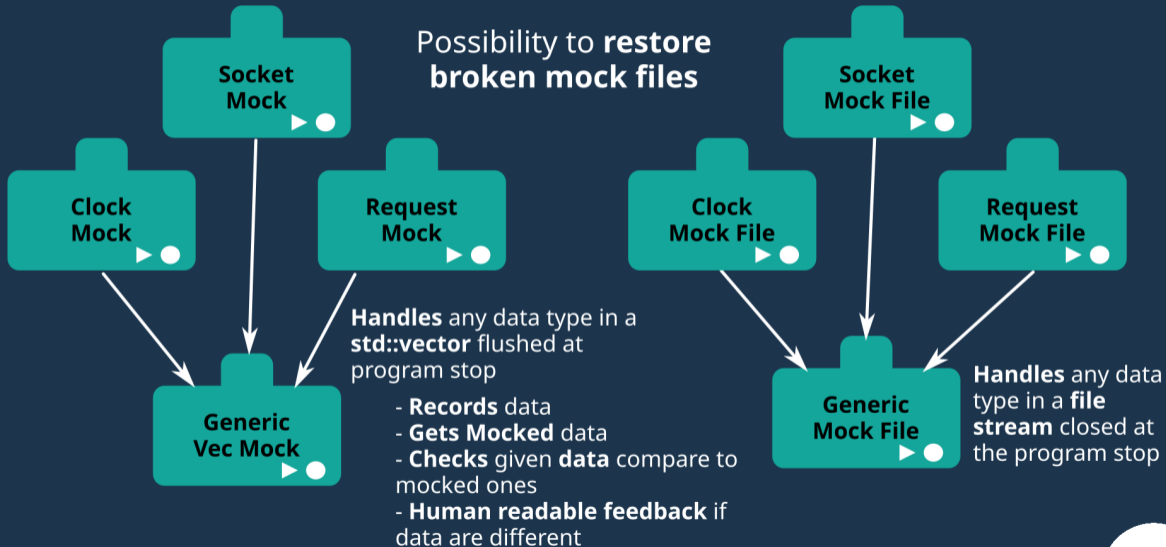
Generic Mock



Generic Mock



Generic Mock



Why use **mocks** ?

- ▶ **Run tests on laptop**, small **CI worker**
- ▶ **No latency**, **no sleeps**, **no timeouts**

Why use **mocks** ?

- ▶ **Run tests on laptop**, small **CI worker**
- ▶ **No latency**, **no sleeps**, **no timeouts**

Mock your **connection API**, not your connected servers :

- ▶ Implement **1 back-end**, mock **any** number of **server using it**
- ▶ Still running in **1 process** on your laptop

Why use **mocks** ?

- ▶ **Run tests on laptop**, small **CI worker**
- ▶ **No latency, no sleeps, no timeouts**

Mock your **connection API**, not your connected servers :

- ▶ Implement **1 back-end**, mock **any** number of **server using it**
- ▶ Still running in **1 process** on your laptop

Use a **mock back-end** and a **real back-end at the same time** :

- ▶ **Create mock files automatically** with **record mode**
- ▶ **Develop** your application with the **mock back-end**, **choose your real back-end later!**

Why use **mocks** ?

- ▶ **Run tests on laptop**, small **CI worker**
- ▶ **No latency, no sleeps, no timeouts**

Mock your **connection API**, not your connected servers :

- ▶ Implement **1 back-end**, mock **any** number of **server using it**
- ▶ Still running in **1 process** on your laptop

Use a **mock back-end** and a **real back-end at the same time** :

- ▶ **Create mock files automatically** with **record mode**
- ▶ **Develop** your application with the **mock back-end**, **choose your real back-end later!**

What is **available now** (C++, Rust, and Python soon) :

- ▶ C++ Project : https://gitlab.in2p3.fr/CTA-LAPP/PHOENIX_LIBS2/serialize-io
- ▶ C++ Project : https://gitlab.in2p3.fr/CTA-LAPP/PHOENIX_LIBS2/network
- ▶ C++ Project : https://gitlab.in2p3.fr/CTA-LAPP/PHOENIX_LIBS2/clock