A practical example of the proposed UI-API

Rasmus Fogh

MXCuBE / ISPyB joint meeting
Trieste, September 2018

GΦL

Credits

- Contents by the MXCuBE developers group
- Main contribution by the MXCuBE3 project
  - who already had a similar API, we could build on

GΦL

Contents

- **Organisation**
- Example: SampleChanger

GΦL

The two sides

User Interface Qt4 / Web

Beamline control (Hardware Objects)

GΦL

Strictly separated

User Interface Qt4 / Web

Air Gap

Message passing only

Beamline control (Hardware Objects)

Principles

User Interface Qt4 / Web

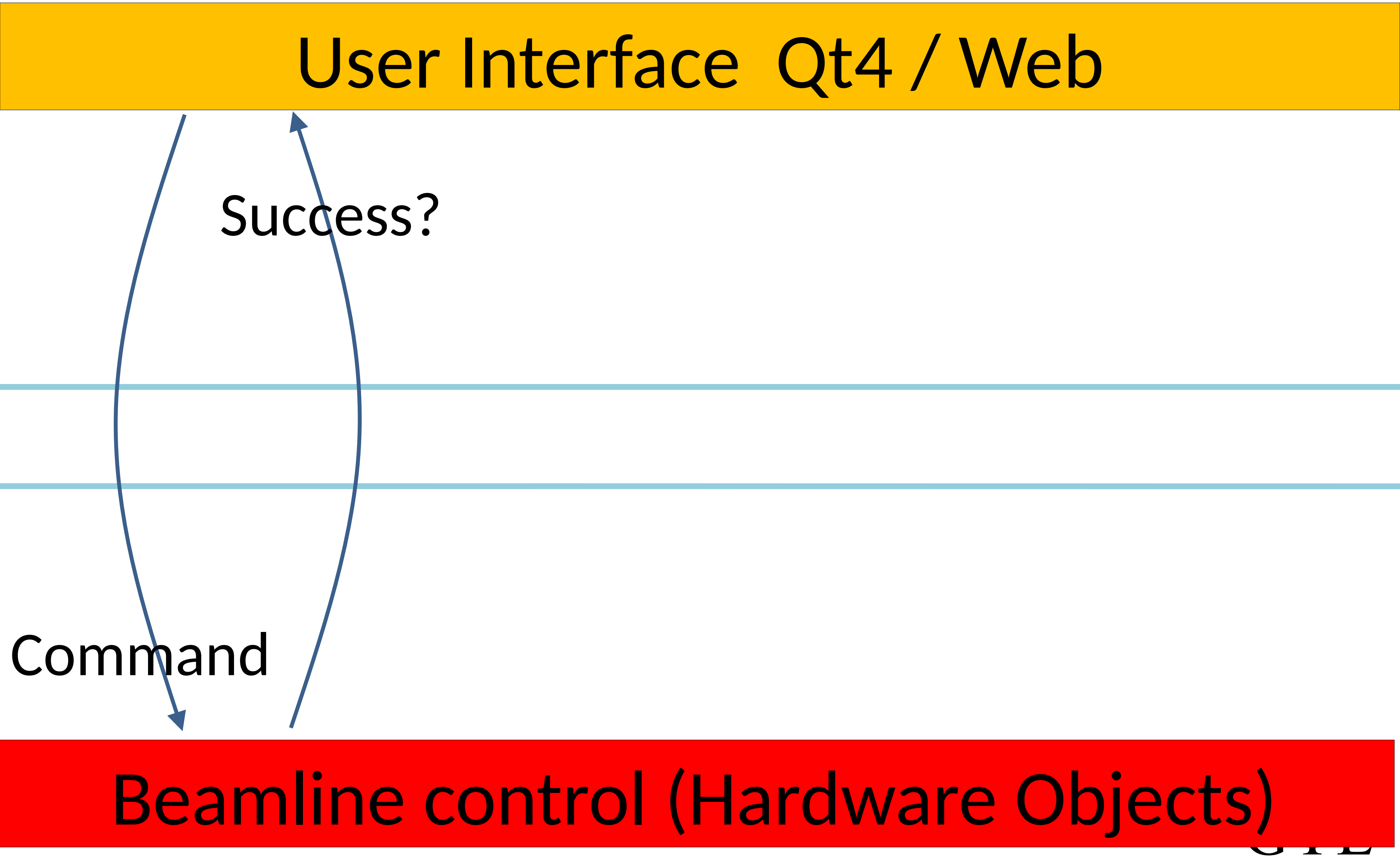
- All actions start on the UI side
- Only data relevant to UI
  - Hardware object data handled elsewhere
  - No sharing of data or objects

Air Gap

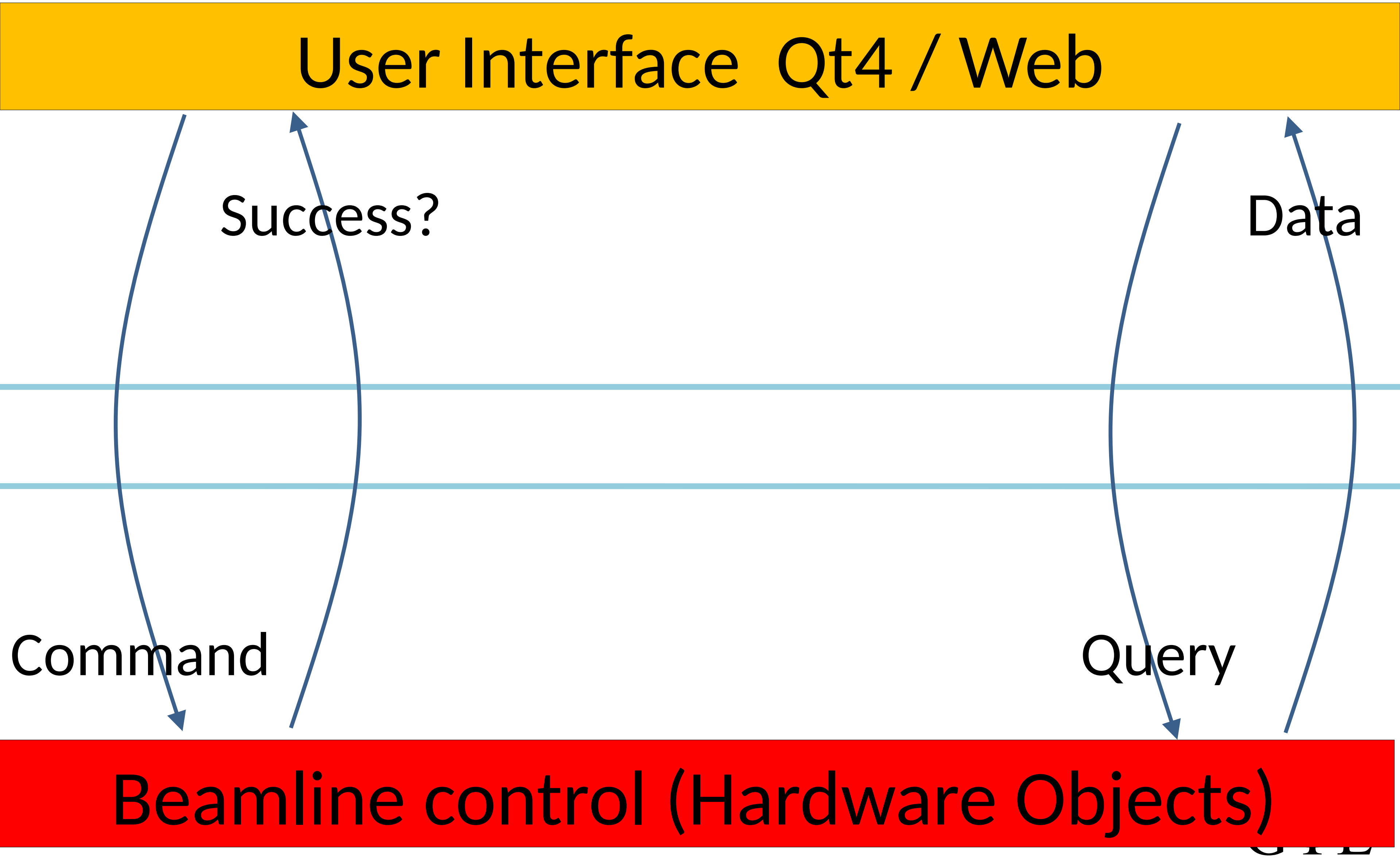
- All UI communication through this interface
- All system state kept on the beamline side
- Mainly high-level commands

Beamline control (Hardware Objects)

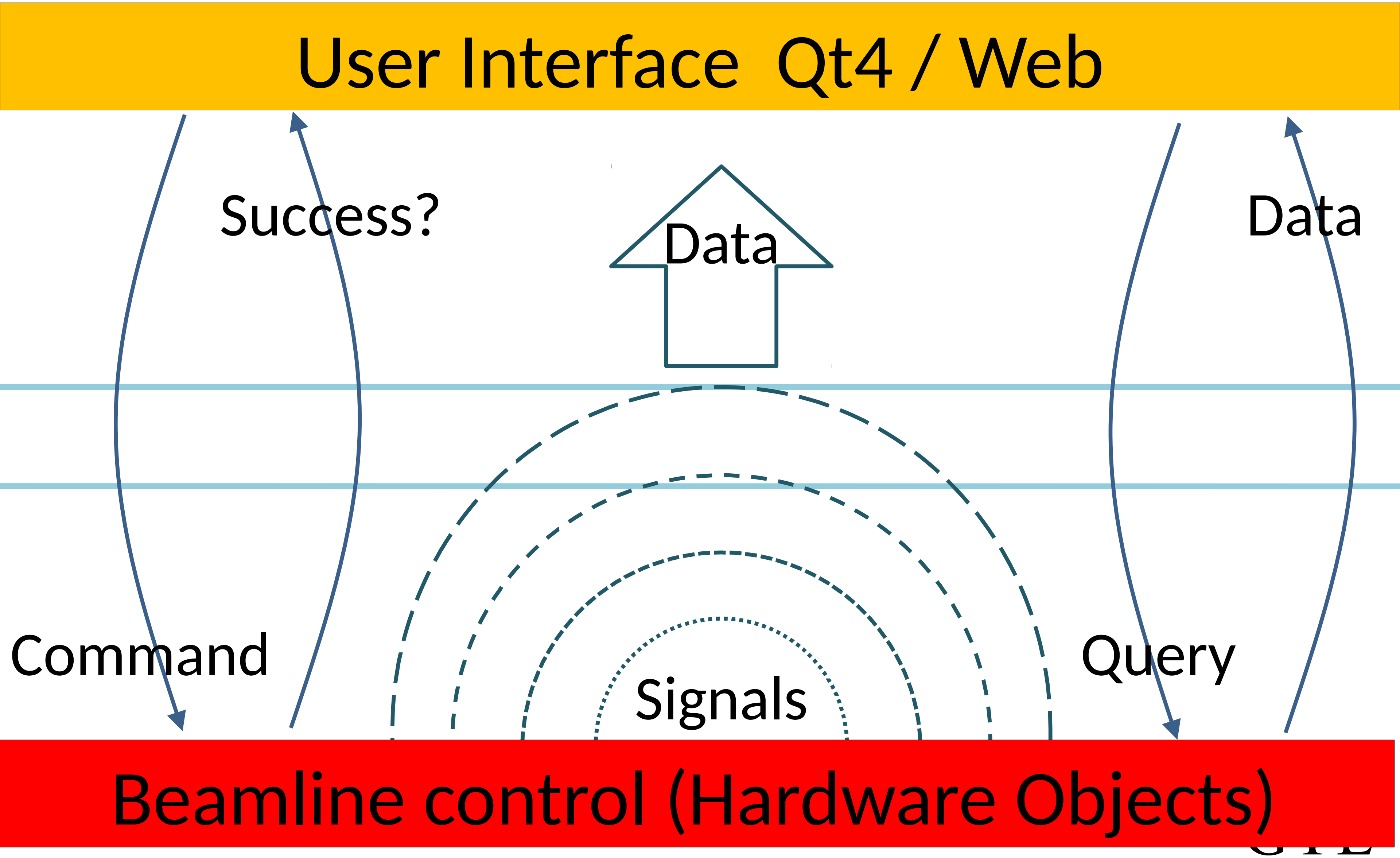
Overview


User Interface Qt4 / Web
Success?
Command
Beamline control (Hardware Objects)

Overview


User Interface Qt4 / Web
Success?
Data
Command
Query
Beamline control (Hardware Objects)

Overview


User Interface Qt4 / Web
Success?
Data
Data
Command
Signals
Query
Beamline control (Hardware Objects)

Contents

- Organisation
- **Example: SampleChanger**
  - *Example is COMPLETE*
  - *Slightly cleaned-up from current draft*

GΦL

Data Structures

```
LocationStr: "cell:basket:sample"


SampleChangerState: # enumeration


class SampleNode:
  id: str
  name: str
  location: LocationStr
  selected: bool
  loadable: bool
  children: List[SampleNode]

class ProcedureData:
  # Command name, parameters, …
  # Defined elsewhere
```

GΦL

Commands

```
select_location(location:LocationStr) -> bool:


scan_location(location:LocationStr) -> bool:
   # Check location for contents


mount_sample(location:LocationStr) -> bool:


unmount_current_sample(to_location:
              LocationStr=None) -> bool:
```

Queries

```
get_state() -> SampleChangerState :

get_current_sample() -> LocationStr :

get_sample_list() -> List[SampleNode]:
   # get list of actual samples

get_sc_contents() -> SampleNode:
   # get hierarchy of samples and containers

get_full_state() -> Dict:
   # All of the above
   # plus get_available_commands and a message
```

GΦL

Signals

```
stateChanged (old_state:SampleChangerState,
         new_state:SampleChangerState)
   # SC state changed

loadedSampleChanged (newSample:SampleNode)
   # New sample loaded

contentsUpdated (sample:SampleNode)
   # Sample queue updated

scError: (error_code:str, msg:str)
   # SC error

cmdStateChanged (commandNames:List[str], msg:str)
   # List of available commands changed
```

GΦL

Additional functions

- Additional functions can be registered
  - Extensions
  - Complex procedures
  - Site-specific functions

```
get_available_commands()
        -> OrderedDict[str, ProcedureData]:
  # List of available commands
  # ‘ProcedureData’ defined elsewhere

exec_command(name:str, **kwargs) -> bool:
  # Execute command with keyword arguments
```

GΦL

END

GΦL