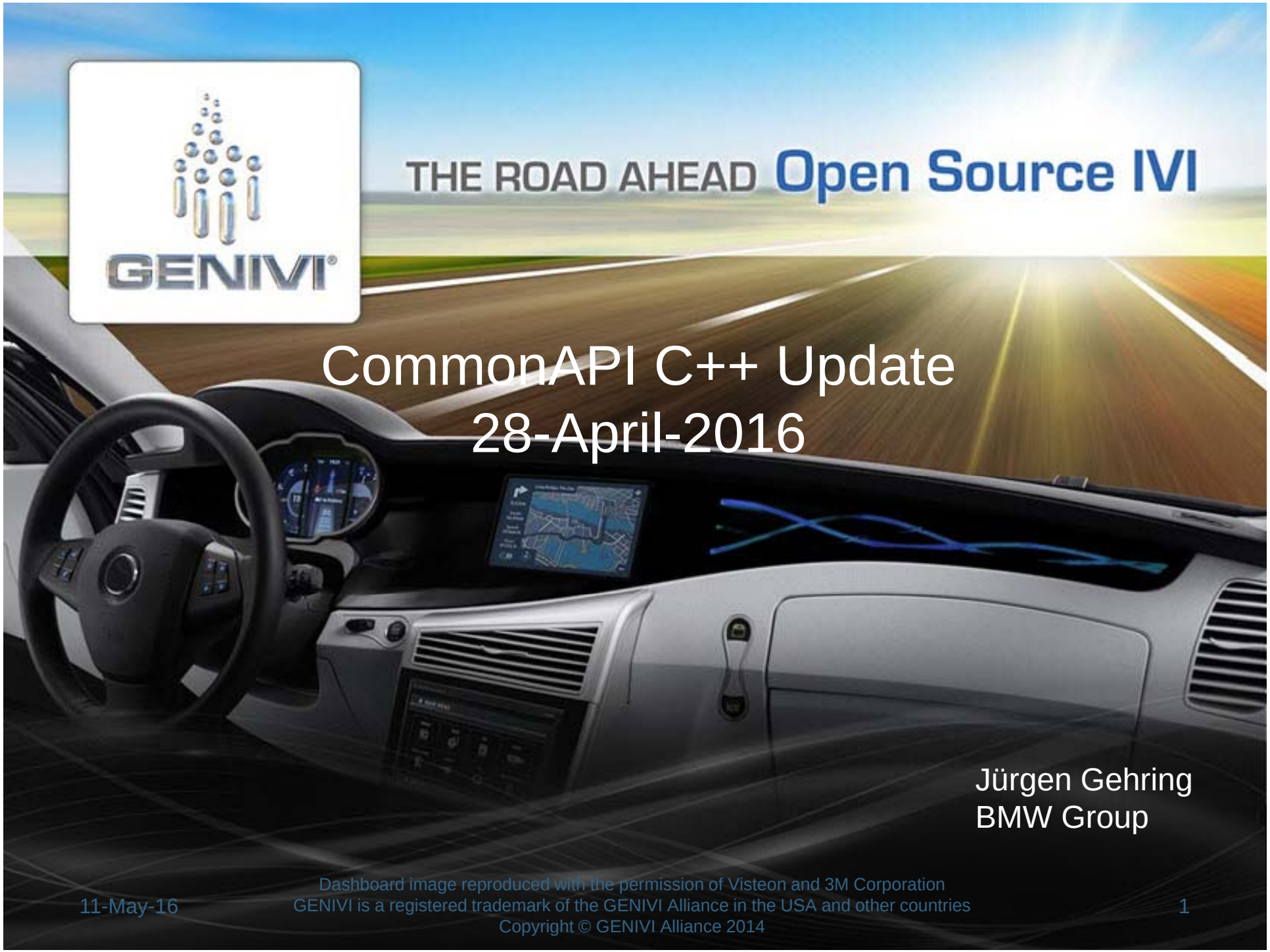


CommonAPI C++ Update
28-April-2016

Jürgen Gehring
BMW Group

Content

- CommonAPI C++ SOME/IP
  - SOME/IP specification
  - vsomeip
  - SOME/IP binding
- CommonAPI C++ Platform Integration
  - Overview
  - Deployment
  - Special Topics

Overview SOME/IP Specification

SOME/IP
On-Wire Format

- Header
- Datatypes
- Serialization

SOME/IP
Protocol

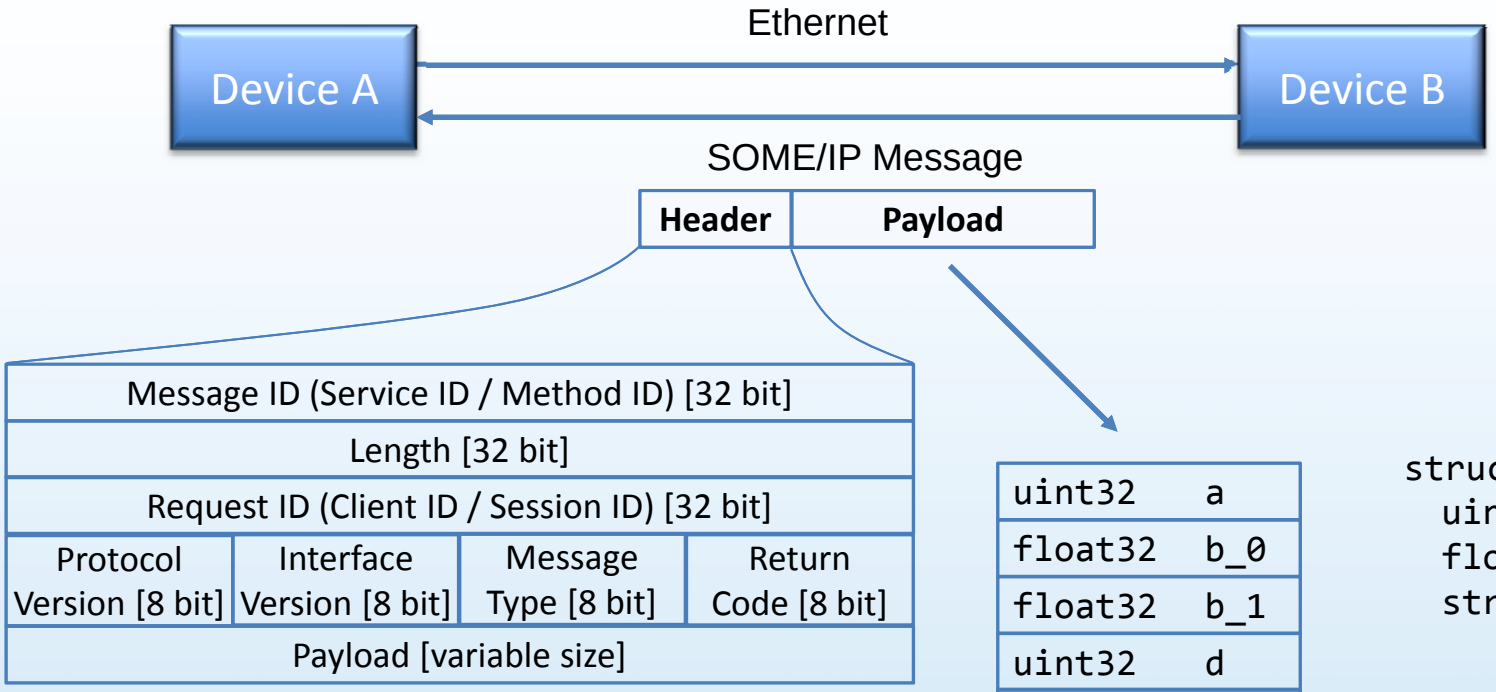
- UDP / TCP
- Request /Response
- Publish / Subscribe
- Fields
- Error Handling

SOME/IP
Service Discovery

- Message Format
- Endpoints
- SD Messages
- Startup / Shutdown

http://some-ip.com/

SOME/IP On-Wire Format

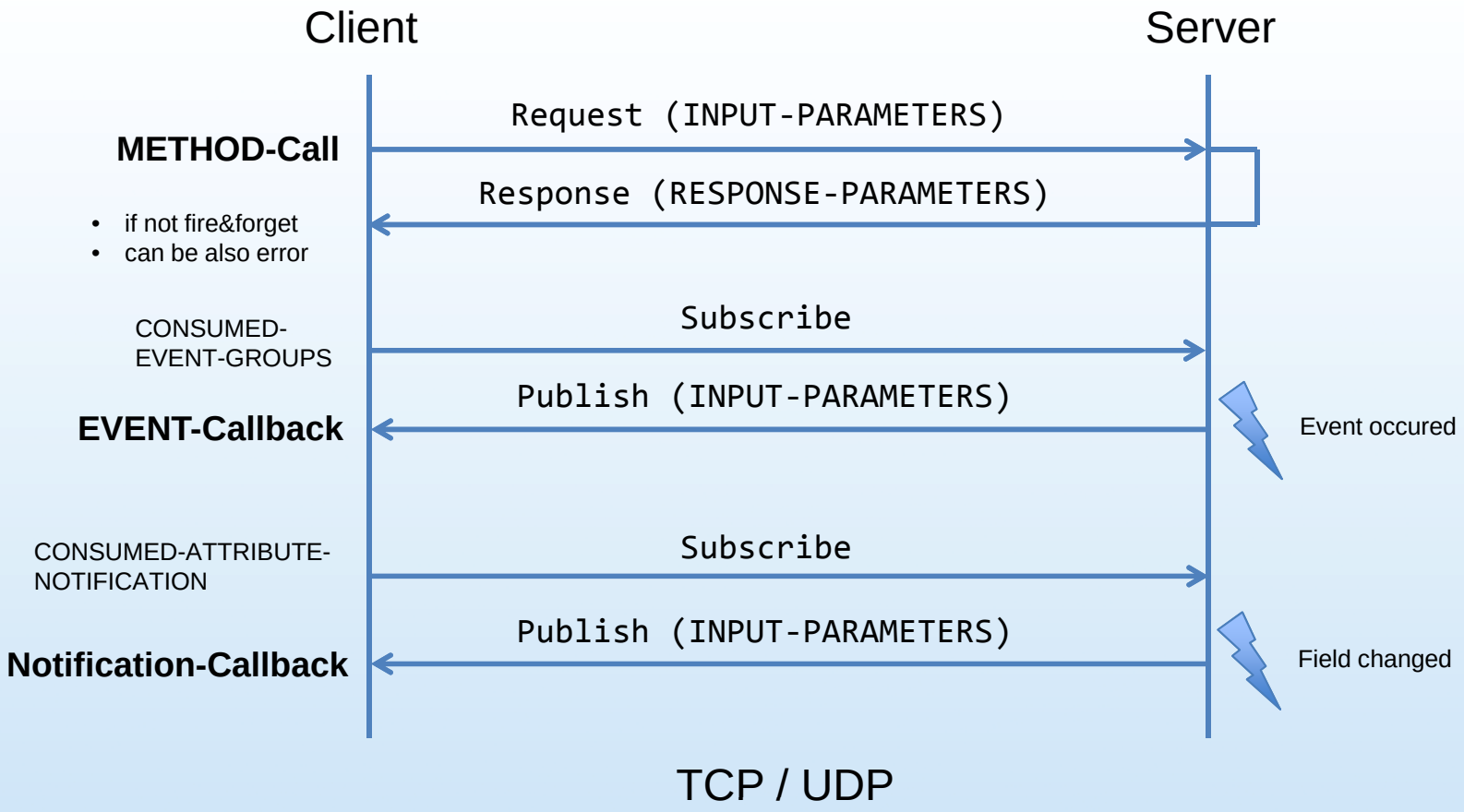


simple types, strings, arrays, enumeration, bitfield, union, (maps)

SOME/IP IDs

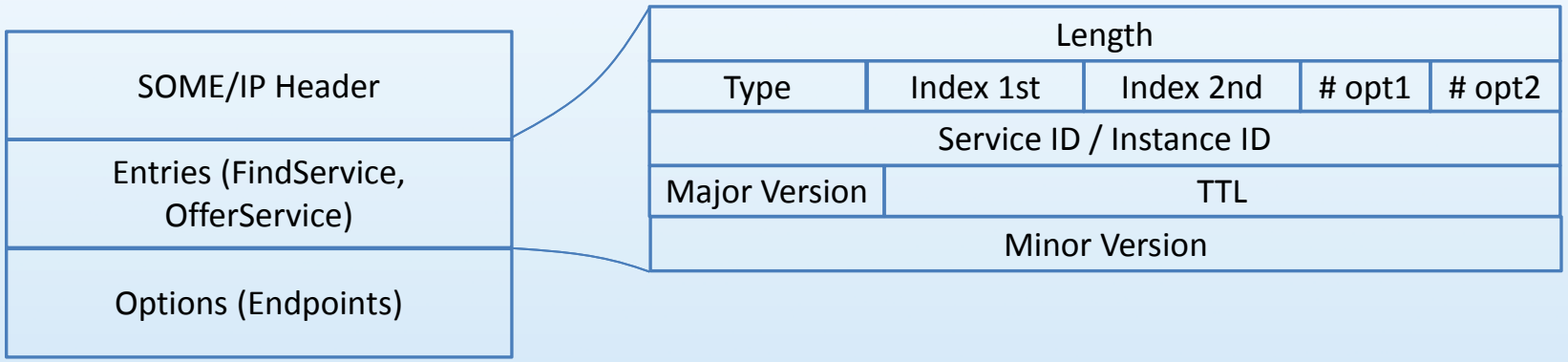
Service ID	unique identifier for each service
Method ID	0-32767 methods, 32768-65535 events
Length	length of payload in byte
Client ID	unique identifier for the calling client inside the ECU
Session ID	identifier for session handling
Protocol Version	0x01
Interface Version	major version of the service interface
Message Type	REQUEST, NOTIFICATION, RESPONSE, ERROR, ...
Return Code	E_OK, E_NOT_OK, E_UNKNOWN_SERVICE, ...

SOME/IP Protocol



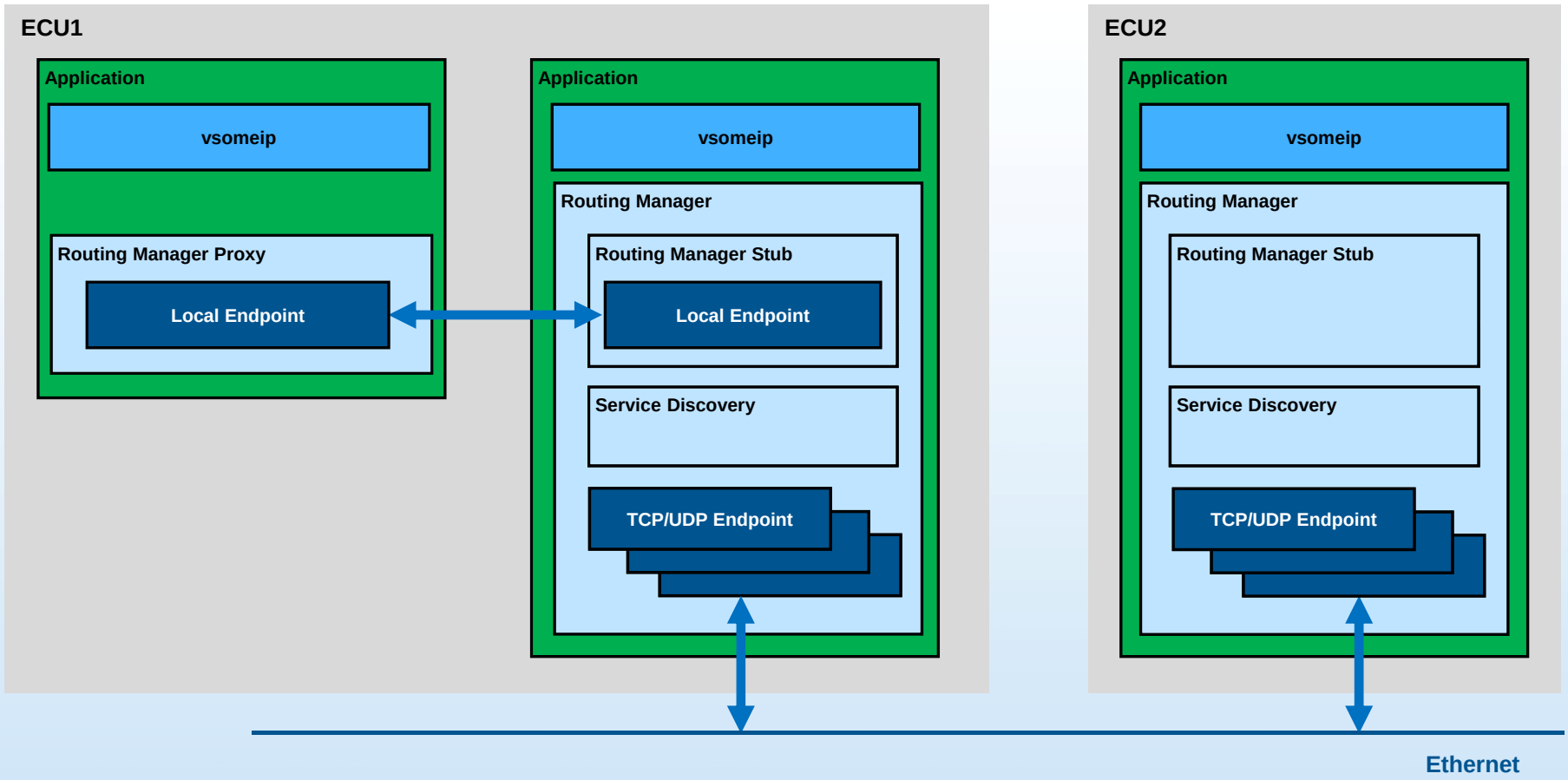
SOME/IP Service Discovery

- SD messages:
  - SERVICE ID 0xFFFF
  - METHOD ID 0x8100
  - CLIENT ID 0x0



- IPv4 Address
- 0x06: TCP, 0x11: UDP)
- Port

vsomeip Architecture



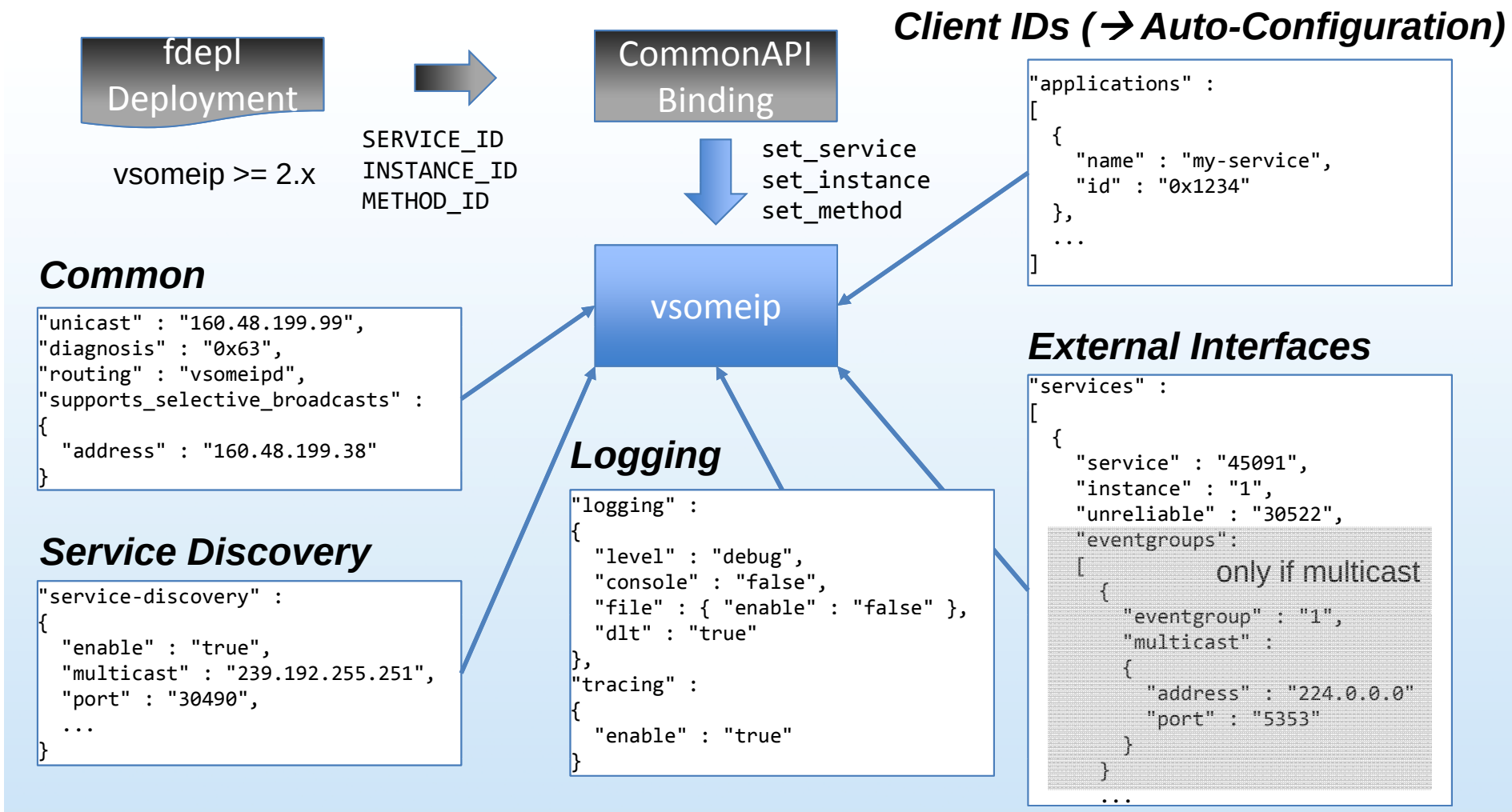
vsomeip Code Example

Example "send REQUEST":

(see request-sample.cpp in examples folder in vsomeip repository)

```
app = vsomeip::runtime::get()->create_application();
request = vsomeip::runtime::get()->create_request(false);
...
app->register_state_handler(...);
app->register_message_handler(...);
...
payload = vsomeip::runtime::get()->create_payload();
...
request->set_payload(payload);
...
request->set_service(SAMPLE_SERVICE_ID);
request->set_instance(SAMPLE_INSTANCE_ID);
request->set_method(SAMPLE_METHOD_ID);
...
app->send(request, true);
```

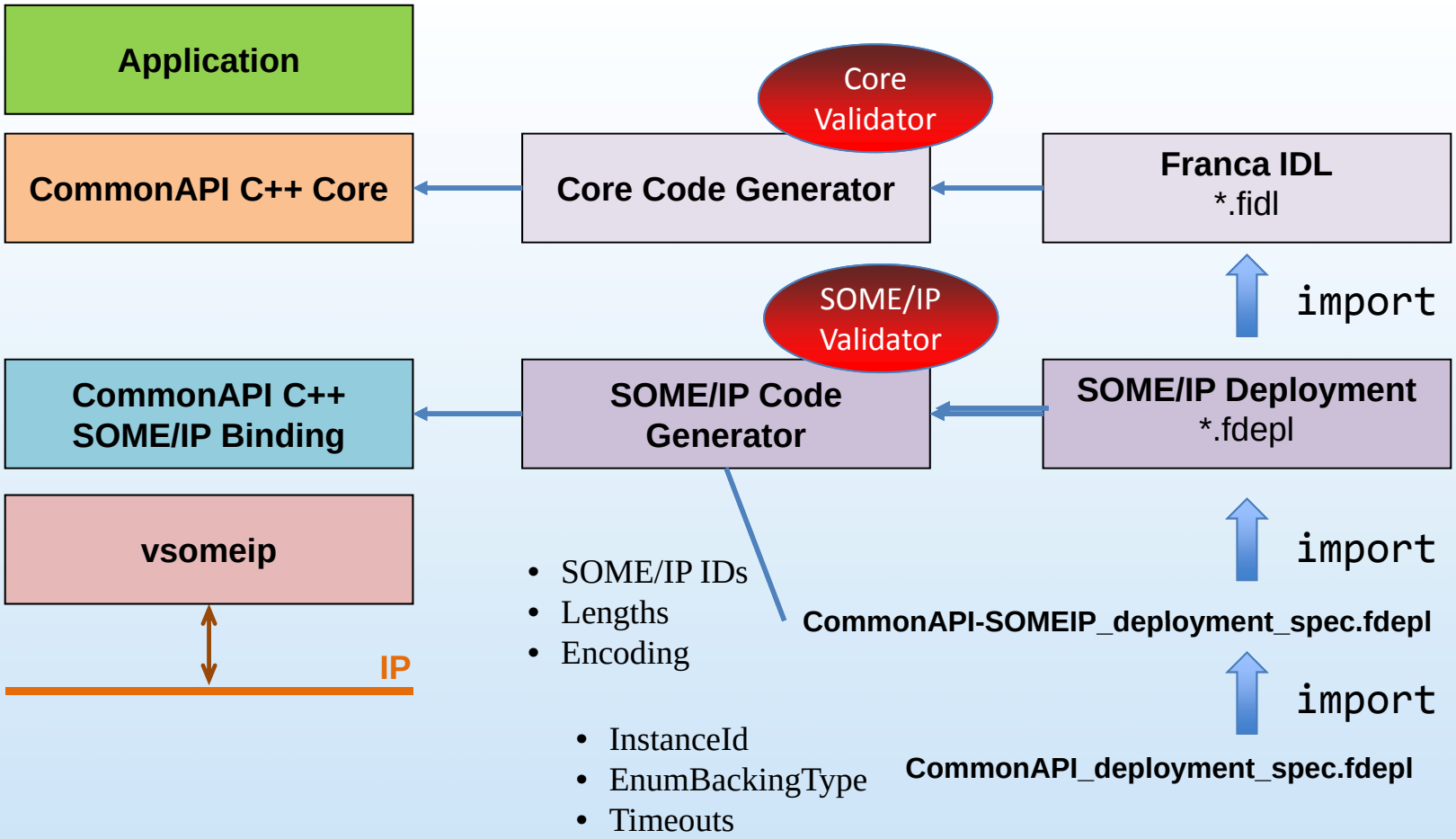
vsomeip Configuration



vsomeip Features Summary

- Full implementation of the SOME/IP specification without serialization
- No dependencies to CommonAPI (only BOOST is used)
- Serialization is done by the CommonAPI SOME/IP binding
- Service Discovery included
- Complete communication backend for external and internal messages
- Device-internal communication by unix domain sockets (point-to-point)
- Auto-configuration of ClientIDs (vsomeip >= 2.x)
- API extension for SOME/IP identifier settings (vsomeip >= 2.x)
- Internal communication completely without any configuration possible
- Configuration can be split into several files in /etc/vsomeip
- IPv6 support

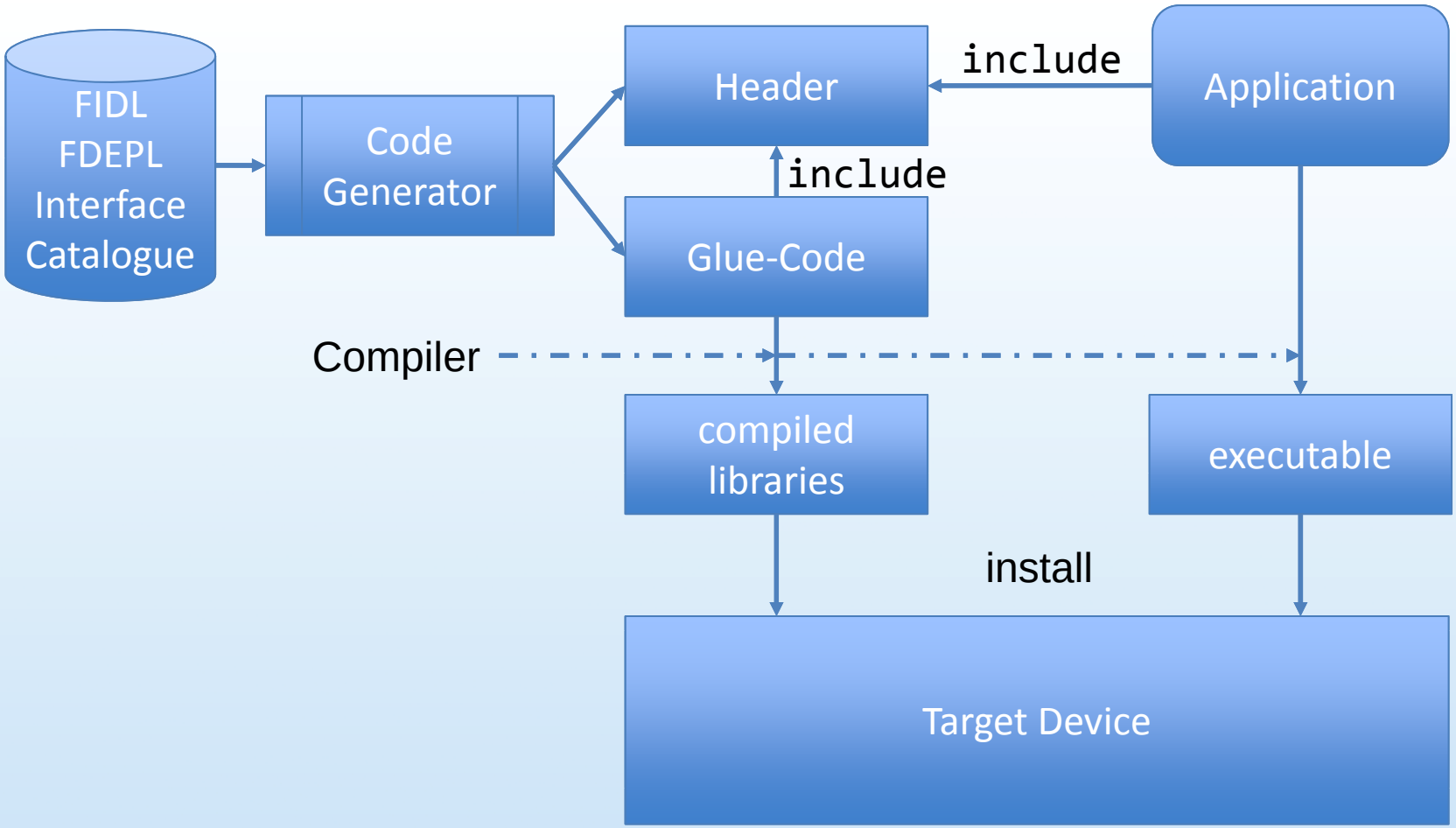
CommonAPI C++ SOME/IP



CommonAPI C++ SOME/IP Restrictions

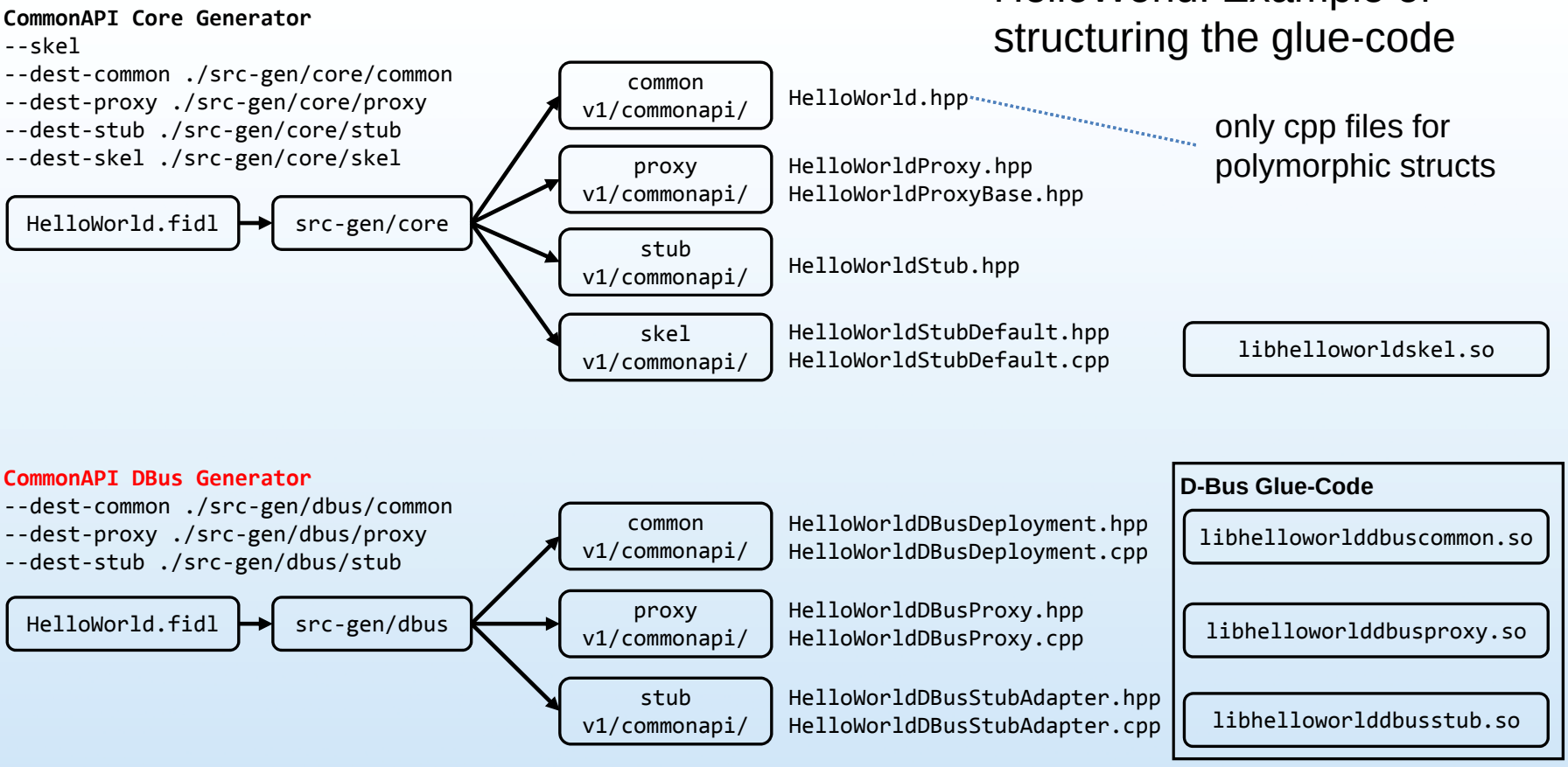
- Code generator works only with deployment (IDs must be specified):
  - Method / Event IDs must be unique (also for extended interfaces)
  - Service ID / Instance ID combination must be unique → internal service ID range
  - Client ID must be unique
- Message length restrictions:
  - external message length is configured per service in json file
  - effective message length is maximum of service message lengths per port
  - default values for external message length is define in SOME/IP specification (tcp 4095 bytes, udp 1416 bytes)
  - internal message length is adapted automatically
- Franca features as *managed*, *polymorphic*, *selective* are only supported by CommonAPI applications.
- If two applications are client for the same selective (external) broadcast, they must use different ports.
- The on-wire realization for maps is an array of struct with key and value element.

CommonAPI C++ Integration Overview

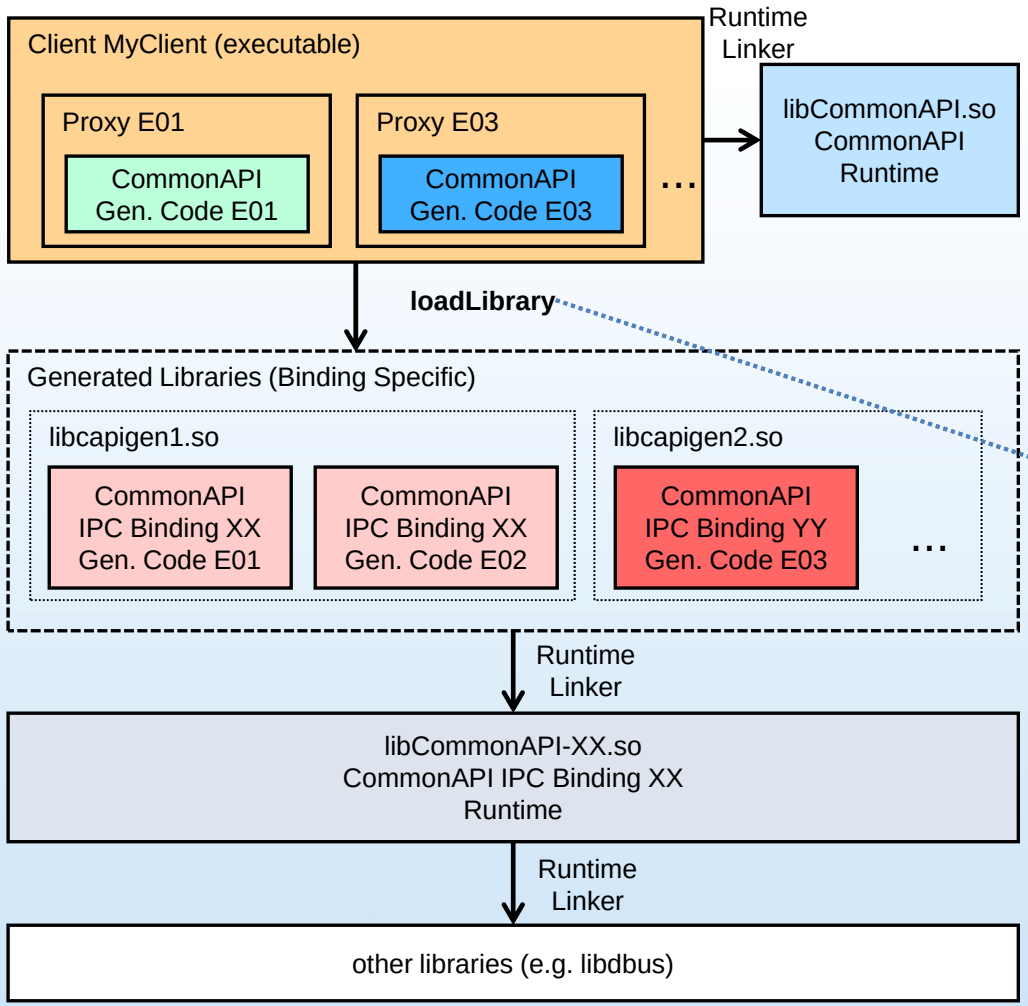


CommonAPI C++ Structuring Glue-Code (Example)

HelloWorld: Example of structuring the glue-code



CommonAPI C++ Load Libraries



Possible solutions:

- loading via runtime linker (library must be linked with `-no-as-needed` flag)
- load via `commonapi.ini` configuration file (standard way)
- (compile all sources in one executable or use static libraries)

Internal Runtime CommonAPI APIs Examples

include/CommonAPI/Runtime.hpp

```
COMMONAPI_EXPORT static
std::shared_ptr<Runtime> get();
```

HelloWorldProxy.hpp

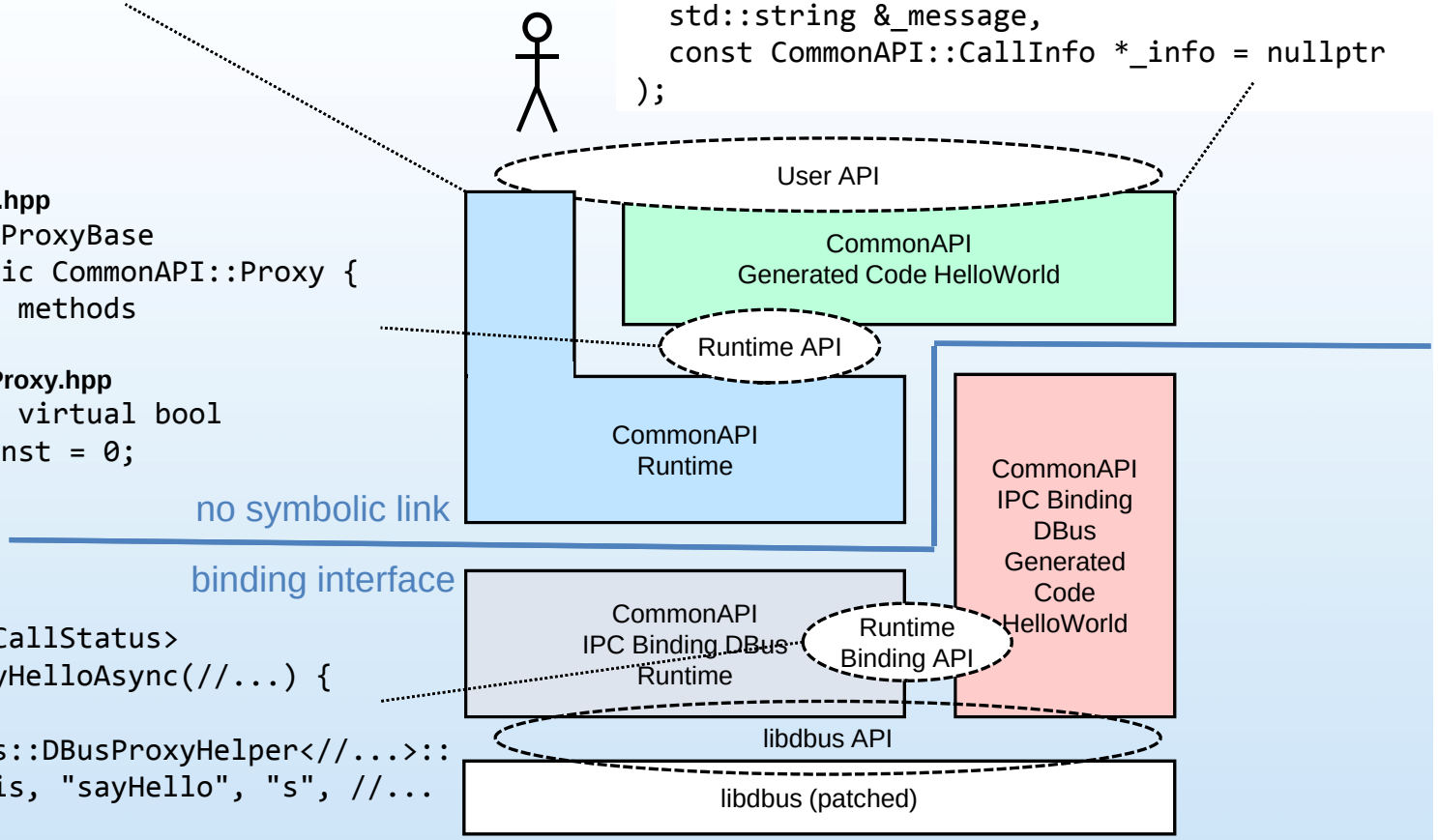
```
virtual void sayHello (
  const std::string &_name,
  CommonAPI::CallStatus &_internalCallStatus,
  std::string &_message,
  const CommonAPI::CallInfo *_info = nullptr
);
```

HelloWorldProxyBase.hpp

```
class HelloWorldProxyBase
  : virtual public CommonAPI::Proxy {
... // generated methods
};
```

include/CommonAPI/Proxy.hpp

```
COMMONAPI_EXPORT virtual bool
isAvailable() const = 0;
```



HelloWorldDBusProxy.cpp

```
std::future<CommonAPI::CallStatus>
HelloWorldDBusProxy::sayHelloAsync(//...) {
  //...
  return CommonAPI::DBus::DBusProxyHelper<//...>::
    callMethodAsync(*this, "sayHello", "s", //...
   );
 }
```

Internal Runtime-Binding APIs Examples

Runtime → Binding Runtime

Example:
Curiously recurring template pattern
include/CommonAPI/Outputstream.hpp

```
template<class Derived_> class OutputStream {
public:
    template<class Deployment_>
    OutputStream &writeValue(
      const bool &_value,
      const Deployment_ *_depl = nullptr) {
        return get()->writeValue(_value, _depl);
      }
    ...
```

include/CommonAPI/DBus/DBusOutputstream.hpp

```
class DBusOutputStream:
  public OutputStream<DBusOutputStream> {
public:
   COMMONAPI_EXPORT OutputStream &writeValue(
     const bool &_value,
     const EmptyDeployment *_depl) {
        (void)_depl;
        uint32_t tmp = (_value ? 1 : 0);
        return _writeValue(tmp);
     }
     ...
```

Runtime → Binding Generated Code

Example:

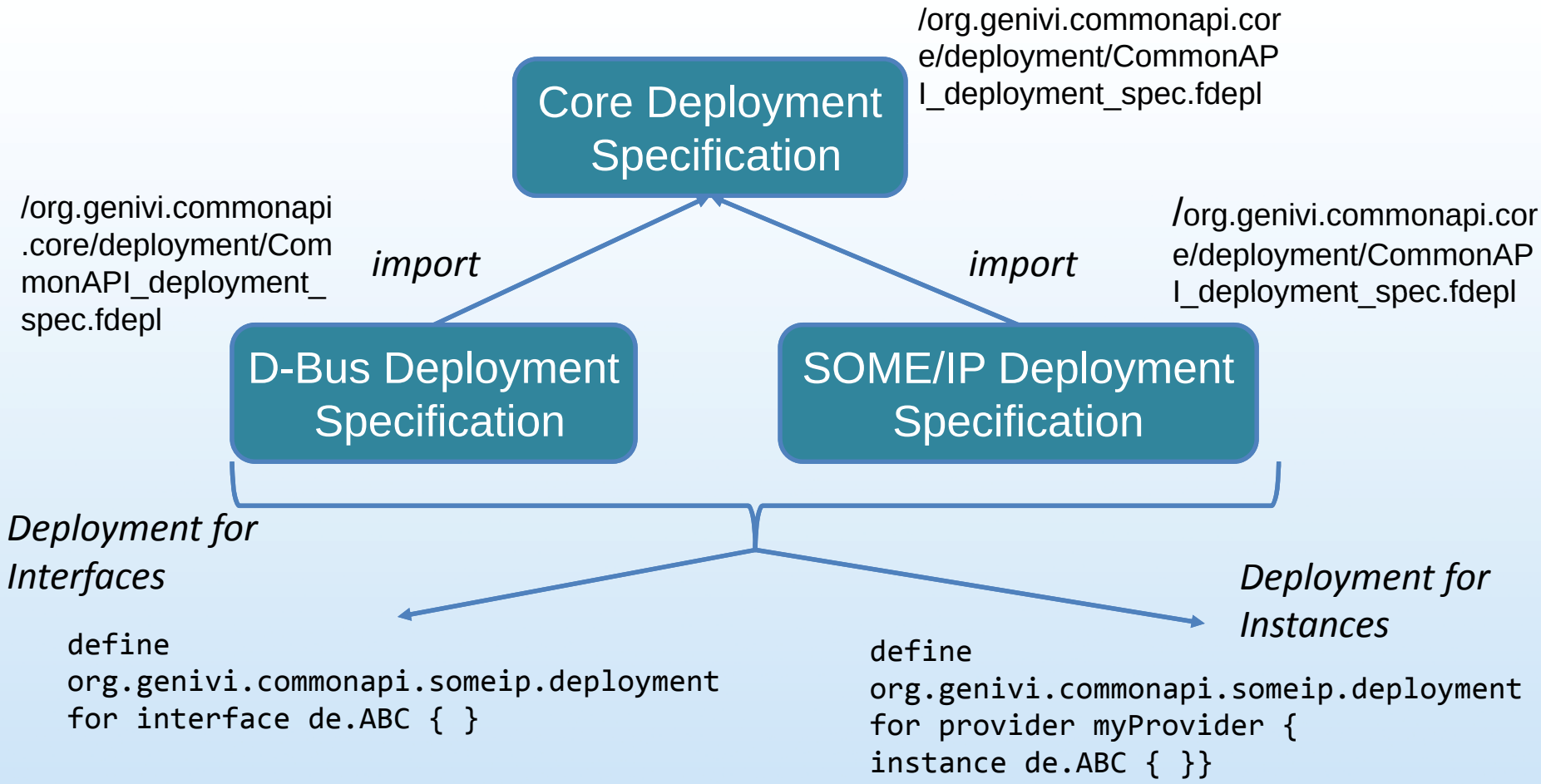
```
class HelloWorldProxy \\...
 private:
 std::shared_ptr<HelloWorldProxyBase> delegate_;
//...
template <//...>
std::future<CommonAPI::CallStatus>
  HelloWorldProxy<//...>::sayHelloAsync(
    const std::string &_name,
    SayHelloAsyncCallback _callback,
    const CommonAPI::CallInfo *_info) {
    return delegate_->sayHelloAsync(
      _name, _callback, _info);
}
```

this delegate is set to the derived binding specific proxy pointer when the glue-code is loaded.

CommonAPI Versions
Integration of CommonAPI

- There is one CommonAPI version number for code generators and runtimes for all bindings (e.g. 3.1.5).
- "Point-fixes" are marked with a fourth version number (e.g. 3.1.5.1) and can be updated independently.
- The vsomeip version is completely independent like the D-Bus version.
- CommonAPI version only concerns application source code: it must not be changed in case of minor or patch updates.
- Best practice: Deliver CommonAPI code without glue-code.
- If binaries are delivered, the CommonAPI version must not be changed → always must recompile (Header). Exeption: There is a pointfix only in cpp files of the runtime.

CommonAPI Deployment



Deployment for Instances

SOME/IP

```
define org.genivi.commonapi.someip.deployment for provider MyServiceSomeIP {
    instance commonapi.HelloWorld {
        InstanceId = "testSomeIP"
        SomeIpInstanceID = 22136
        // + IP Address + Port = Endpoint
    }
}
```

SomeIPServiceID is part of the interface deployment

Name of CommonAPI Instance

D-Bus

```
define org.genivi.commonapi.dbus.deployment for provider MyServiceDBus {
    instance commonapi.HelloWorld {
        InstanceId = "testDBus"
        DBusServiceName = "xyz.MyServiceName"
        DBusObjectPath = "/abc/def"
        DBusInterfaceName = "xyz.BobsHelloWorld"
    }
}
```

Generated Code for instances

Example:
Generated code for the address translator in HelloWorldDBusProxy.cpp.

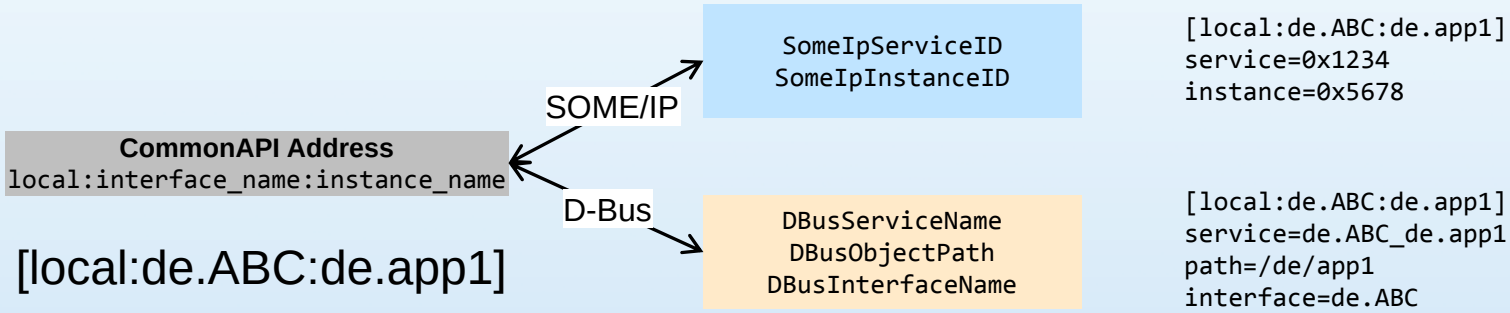
```
void initializeHelloWorldDBusProxy() {
    CommonAPI::DBus::DBusAddressTranslator::get()->insert(
        "local:commonapi.HelloWorld:testDBus",
        "xyz.MyServiceName",
        "/abc/def",
        "xyz.BobsHelloWorld");
     //...
}

INITIALIZER(registerHelloWorldDBusProxy) {
    CommonAPI::DBus::Factory::get()->registerInterface(initializeHelloWorldDBusProxy);
}
```

The binding specific CommonAPI address translator is initialized by deployment settings.

CommonAPI Address Translator

1. **CommonAPI configuration file: c**onfiguration of address translator by `commonapi-dbus.ini` or `commonapi-someip.ini` (use local file or set `COMMONAPI_SOMEIP_CONFIG` or `COMMONAPI_DBUS_CONFIG`).
2. **Deployment:** Generated code from provider deployment files with definded Instance IDs → no further configuration necessary.
3. **CommonAPI address translator API in code:** Address.hpp (setInterface, setInstance, ...).
4. **Automatic conversion:** see below (only D-Bus).



Thank You For Your Attention!