CommonAPI C++ Update
28-April-2016

Jürgen Gehring
BMW Group
• 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

- Ethernet

Device A → Device B

SOME/IP Message

**Header**

- Message ID (Service ID / Method ID) [32 bit]
- Length [32 bit]
- Request ID (Client ID / Session ID) [32 bit]

**Payload**

- Protocol Version [8 bit]
- Interface Version [8 bit]
- Message Type [8 bit]
- Return Code [8 bit]
- Payload [variable size]

```
struct x1 {
    uint32 a
    float32 b_0
    float32 b_1
    uint32 d
    float32 e_0
    float32 e_1
    uint8 f
}
```

```
struct x2 {
    uint32 d
    float32 e[2]
    uint8 f
}
```

Simple types, strings, arrays, enumeration, bitfield, union, (maps)
### SOME/IP IDs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service ID</td>
<td>unique identifier for each service</td>
</tr>
<tr>
<td>Method ID</td>
<td>0-32767 methods, 32768-65535 events</td>
</tr>
<tr>
<td>Length</td>
<td>length of payload in byte</td>
</tr>
<tr>
<td>Client ID</td>
<td>unique identifier for the calling client inside the ECU</td>
</tr>
<tr>
<td>Session ID</td>
<td>identifier for session handling</td>
</tr>
<tr>
<td>Protocol Version</td>
<td>0x01</td>
</tr>
<tr>
<td>Interface Version</td>
<td>major version of the service interface</td>
</tr>
<tr>
<td>Message Type</td>
<td>REQUEST, NOTIFICATION, RESPONSE, ERROR, ...</td>
</tr>
<tr>
<td>Return Code</td>
<td>E_OK, E_NOT_OK, E_UNKNOWN_SERVICE, ...</td>
</tr>
</tbody>
</table>
SOME/IP Protocol

Client

METHOD-Call
- if not fire&forget
- can be also error

EVENT-Callback

Notification-Callback

Server

Request (INPUT-PARAMETERS)

Response (RESPONSE-PARAMETERS)

Subscribe

Publish (INPUT-PARAMETERS)

Subscribe

Publish (INPUT-PARAMETERS)

TCP / UDP

Event occurred

Field changed

11-May-16
SOME/IP Service Discovery

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

- IPv4 Address
- 0x06: TCP, 0x11: UDP
- Port
Example "send REQUEST":
(see request-sample.cpp in examples folder in vsomeip repository)

```cpp
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

Client IDs (→ Auto-Configuration)

```
"applications": [
  {
    "name": "my-service",
    "id": "0x1234"
  },
  ...
]
```

Common

```
"unicast": "160.48.199.99",
"diagnosis": "0x63",
"routing": "vsomeip",
"supports_selective_broadcasts": {
  "address": "160.48.199.38"
}
```

Service Discovery

```
"service-discovery": {
  "enable": "true",
  "multicast": "239.192.255.251",
  "port": "30490",
  ...
}
```

Logging

```
"logging": {
  "level": "debug",
  "console": "false",
  "file": { "enable": "false" },
  "dlt": "true"
},
"tracing": {
  "enable": "true"
}
```

External Interfaces

```
"services": {
  {"service": "45091",
   "instance": "1",
   "unreliable": "30522",
   "eventgroups": {
     "enable": true,
     "multicast": {"address": "224.0.0.0",
                   "port": "5353"}
   }},
  ...```
• 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

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

FIDL Interface Catalogue

Code Generator

Header

Glue-Code

Application

include

include

Compiler

compiled libraries

compiled lib
rar

target device

target device

executable

Target Device

include

install
HelloWorld: Example of structuring the glue-code

only cpp files for polymorphic structs

libhelloworldskel.so

CommonAPI Core Generator
--skel
--dest-common ./src-gen/core/common
--dest-proxy ./src-gen/core/proxy
--dest-stub ./src-gen/core/stub
--dest-skel ./src-gen/core/skel

HelloWorld.fidl src-gen/core

common v1/commonapi/
proxy v1/commonapi/
stub v1/commonapi/
skel v1/commonapi/

HelloWorld.hpp
HelloWorldProxy.hpp
HelloWorldProxyBase.hpp
HelloWorldStub.hpp
HelloWorldStubDefault.hpp
HelloWorldStubDefault.cpp

CommonAPI DBus Generator
--dest-common ./src-gen/dbus/common
--dest-proxy ./src-gen/dbus/proxy
--dest-stub ./src-gen/dbus/stub

HelloWorld.fidl src-gen/dbus

common v1/commonapi/
proxy v1/commonapi/
stub v1/commonapi/

HelloWorldDBusDeployment.hpp
HelloWorldDBusDeployment.cpp
HelloWorldDBusProxy.hpp
HelloWorldDBusProxy.cpp
HelloWorldDBusStubAdapter.hpp
HelloWorldDBusStubAdapter.cpp

D-Bus Glue-Code
libhelloworlddbuscommon.so
libhelloworlddbusproxy.so
libhelloworldbusstub.so
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)
include/CommonAPI/Runtime.hpp
COMMONAPI_EXPORT static
std::shared_ptr<Runtime> get();

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

Example:
Curiously recurring template pattern

```cpp
#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

  classDBusOutputStream:
    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);
        }

      ...}
```
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). Exception: There is a pointfix only in cpp files of the runtime.
CommonAPI Deployment

Core Deployment Specification

D-Bus Deployment Specification

SOME/IP Deployment Specification

import 

import 

Deployment for Interfaces

define org.genivi.commonapi.someip.deployment for interface de.ABC { }

Deployment for Instances

define org.genivi.commonapi.someip.deployment for provider myProvider { 
instance de.ABC { } }
Deployment for Instances

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

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

SOME/IP
Name of CommonAPI Instance
SomeIPSomeId is part of the interface deployment

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

```c++
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**: configuration 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 defined 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).

CommonAPI Address
local:interface_name:instance_name

D-Bus
DBusServiceName
DBusObjectPath
DBusInterfaceName

SOME/IP
SomeIpServiceID
SomeIpInstanceID

[local:de.ABC:de.app1]
service=0x1234
instance=0x5678

[local:de.ABC:de.app1]
service=de.ABC.de.app1
path=/de/app1
interface=de.ABC
Thank You For Your Attention!