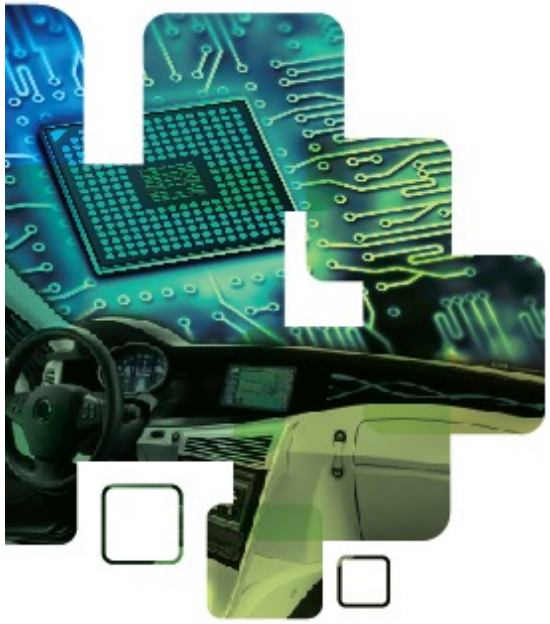


CommonAPI C++ Update

21-Oct-15

Jürgen Gehring
BMW Group

Content

- CommonAPI C++ 3
  - Introduction
  - New Features / API Changes
  - Roadmap
- yamaica
  - Overview and Roadmap

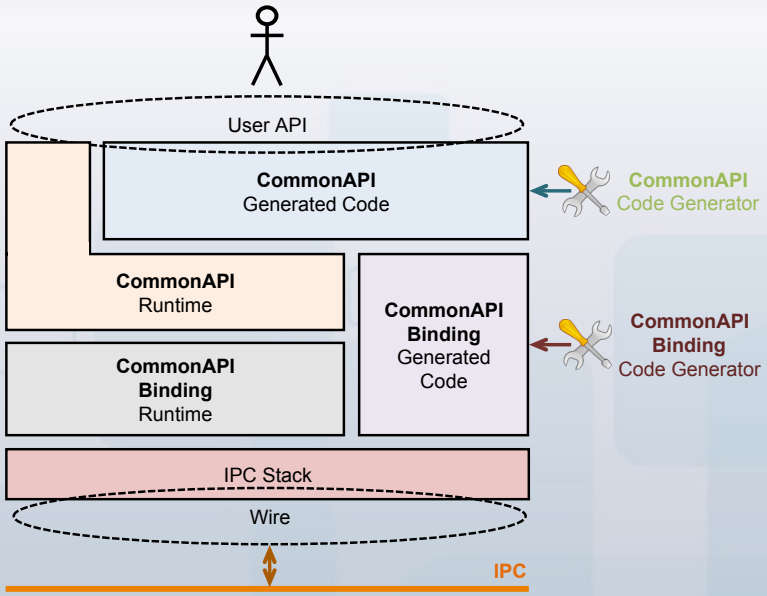
CommonAPI C++ Basic Features

- High-level, thread-safe C++ API for IPC
- Adaption of application code to specific IPC by CommonAPI bindings
- Actual support for D-Bus (libdbus) and SOME/IP
- Franca IDL base code generator (actual Franca 0.9.1)
- High performant implementation using C++ templates

CommonAPI C++ Documentation / Links

- **Wiki:** https://genivi-ss.atlassian.net/wiki/display/COMMONAPICPP/CommonAPI-cpp
- **Specification / UserGuides:** http://docs.projects.genivi.org/ipc.common-api-tools/ (see binding specific tools projects for binding specific documentation)
- **Source code:** http://git.projects.genivi.org/ (note that the SOME/IP stack for CommonAPI C++ SOME/IP is vsomeip).
- **Executable Code Generators:** http://docs.projects.genivi.org/yamaica-update-site/CommonAPI/generator/

CommonAPI C++ 3 and yamaica

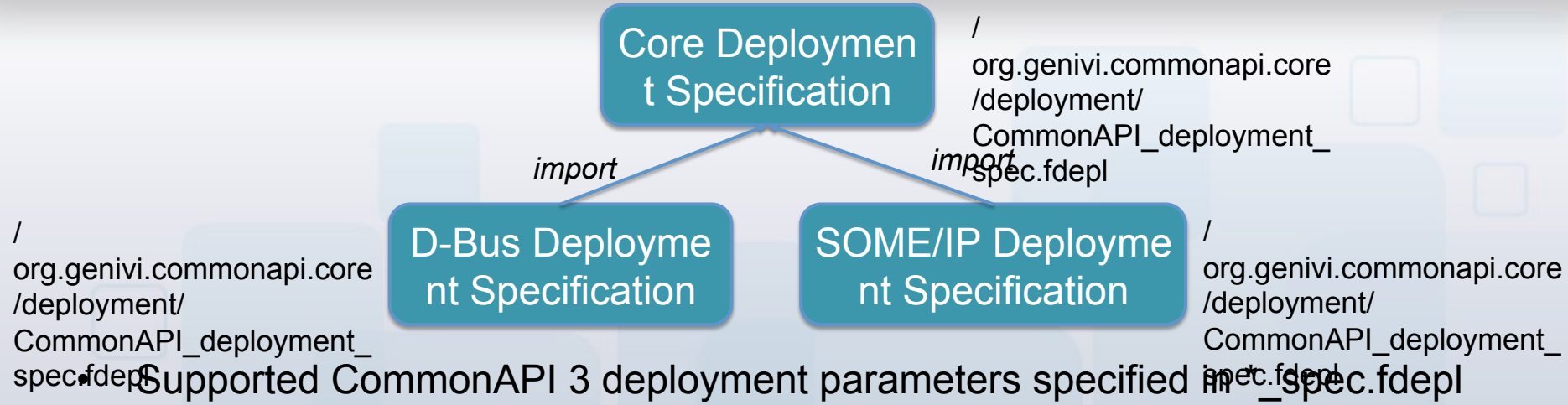


Tool for editing and transforming interface descriptions based on Eclipse Modeling Tools, Xtext and Xtend.

CommonAPI C++ 3 New Features

- New concept for generic integration of deployment parameters
- CommonAPI C++ logging (supports integration with DLT)
- Enhanced code generator features
- New runtime loading concept
- Multi-version support
- Asynchronous stubs
- New configuration parameters (e.g. timeouts)
- SOME/IP binding

CommonAPI C++ 3 Deployment Parameters



- Supported CommonAPI 3 deployment parameters specified in *_spec.fdepl
  - Deployment parameters cover serialization parameters (e.g. string encoding) and deployment parameters like instance names
  - For SOME/IP a deployment specification is mandatory (the SOME/IP code generator can only be started with a fdepl-file

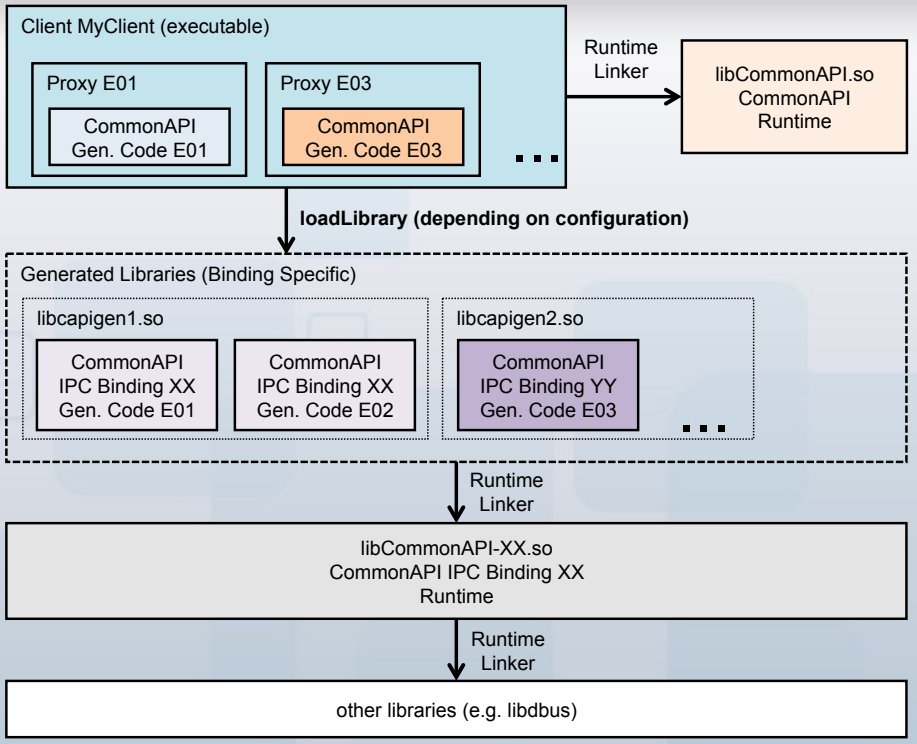
CommonAPI C++ 3 Core Deployment

```
specification org.genivi.commonapi.core.deployment {
    for interfaces {
        DefaultEnumBackingType : {UInt8, UInt16, UInt32, UInt64, Int8, Int16, Int32, Int64} (default: UInt32);
    }
    for providers {
        ClientInstanceReferences : Instance[] (optional);
    }
    for instances {
        Domain :                    String (default: "local"); // the domain part of the CommonAPI address.
        InstanceId :                String;                    // the instance id of the CommonAPI address.
        DefaultMethodTimeout :      Integer (default:0);
        PreregisteredProperties : String [] (optional);
    }
    for methods {
        Timeout : Integer (default: 0);
    }
    for enumerations {
        EnumBackingType : {UInt8, UInt16, UInt32, UInt64, Int8, Int16, Int32, Int64} (optional);
    }
}
```

CommonAPI C++ 3 SOME/IP Deployment

```
specification org.genivi.commonapi.someip.deployment extends org.genivi.commonapi.core.deployment
for instances {
        SomeIpInstanceID:           Integer                 ;
        SomeIpUnicastAddress:       String                  (default: "");
        SomeIpReliableUnicastPort:  Integer                 (default: 0);
        SomeIpUnreliableUnicastPort: Integer                (default: 0);
... // Other parameters here

        for interfaces {
                SomeIpServiceID:            Integer                 ;
                SomeIpEventGroups:          Integer[]               (optional);
        }

        for attributes {
                SomeIpGetterID:             Integer                 (optional);
                SomeIpGetterReliable:       Boolean                 (default: false);
                SomeIpGetterPriority:       Integer                 (optional);
... // Other parameters here

        for strings {
                SomeIpStringLength:         Integer                 (default: 0);
                SomeIpStringEncoding:       {utf8, utf16le, utf16be}    (default: utf8);
... // Other parameters here
```

CommonAPI C++ 3 Loading Runtime



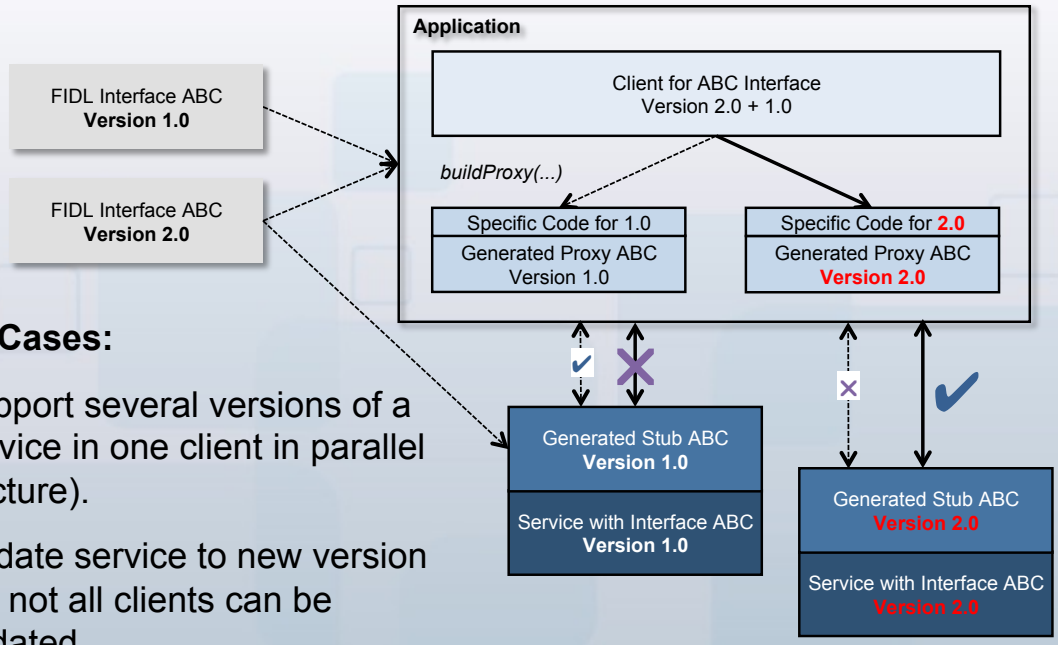
- The application implements against the CommonAPI C++ runtime API and the CommonAPI generated API.
- The executable application links against the CommonAPI runtime library and the generated CommonAPI C++.
- The binding dependent generated glue-code (D-Bus or SOME/IP) can be built in one or several glue-code libraries
- The CommonAPI configuration or the application itself defines which glue-code library is loaded for a certain instance of an interface.
- In order to build the glue-code libraries it is necessary to link against the binding specific runtime library.

CommonAPI C++ 3 Loading Runtime

Standard CommonAPI loading code:

```
/* DLT context ID, only necessary in case of DLT logging */
CommonAPI::Runtime::setProperty("LogContext", "ABCD");

/* Optional: Load this library if there is no other library configured */
CommonAPI::Runtime::setProperty("LibraryBase", "MyLibrary");

/* Get generic CommonAPI runtime object */
std::shared_ptr<CommonAPI::Runtime> runtime = CommonAPI::Runtime::get();

/* The domain is always local */
std::string domain = "local";

/* Instance name */
std::string instance = "MyInstance";

/* Optional: Connection ID (= internal thread if no external mainloop is used, replaces old factory) */
std::string id = "app01";

/* Get proxy; no factory is needed; necessary is only instance name */
std::shared_ptr<MyProxy<>> myProxy = runtime->buildProxy<MyProxy>(domain, instance, id);
```

CommonAPI C++ 3 Parallel Versions



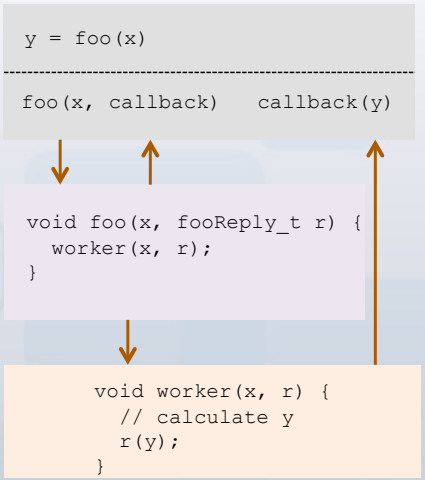
Use Cases:

- Support several versions of a service in one client in parallel (picture).
- Update service to new version but not all clients can be updated.

Parallel Versions:

- The version is part of the namespace; therefore different versions of the same service can exist in parallel.
- By default the code generator generates code in subdirectories with the name of the version.
- Different versions must be provided by different instances of the same service.

CommonAPI C++ 3 Asynchronous Stubs



Client

Service

- It is not necessary anymore to calculate the return values in the stub implementation synchronously.
- Use Cases:
- Applications which work as intermediate layer that only passes messages.
- Swap out the calculation of big computationally intensive tasks to worker threads.

CommonAPI C++ 3 CallInfo

```
namespace CommonAPI {

struct COMMONAPI_EXPORT CallInfo {
        CallInfo()
                : timeout_(DEFAULT_SEND_TIMEOUT_MS), sender_(0) {
        }
        CallInfo(Timeout_t _timeout)
                : timeout_(_timeout), sender_(0) {
        }
        CallInfo(Timeout_t _timeout, Sender_t _sender)
                : timeout_(_timeout), sender_(_sender) {
        }

        Timeout_t timeout_;
        Sender_t sender_;
};

} // namespace CommonAPI
```

- It is possible to pass an optional CallInfo argument to all function calls (also setter/getter of attributes).
- At the moment CallInfo contains the timeout parameter (when do I expect that the function returns) and a sender identifier. The timeout can also be defined in the deployment.
- The sender identifier is only used to log the correlation between this ID and IPC specific identifiers (like the D-Bus serial number).

CommonAPI C++ 3 SOME/IP

- SOME/IP: Scalable service-Oriented middlewarE over IP is an automotive/embedded RPC mechanism including the definition of the serialization / wire format.
- The specification defines datatypes, fields, messages, events, subscriptions, service discovery and so on (refer to http://some-ip.com/ for the details).
- CommonAPI SOME/IP provides a full implementation of this specification; the serialization is done by the CommonAPI SOME/IP binding; the implementation of the basic communication and the service discovery by vsomeip (see http://git.projects.genivi.org/?p=vSomeIP.git;a=summary).
- For CommonAPI SOME/IP see the git projects common-api/cpp-someip-runtime.git and common-api/cpp-someip-tools.git at GENIVI.

CommonAPI C++ 3 Roadmap

- Actual version CommonAPI 3.1.3
- Mid of November 2015 CommonAPI 3.1.4: Bugfixing
- January 2016 CommonAPI 3.2.0: Enhanced Service Discovery features and version management.

yamaica

- yamaica is an acronym for **y**et **a**nother **m**odel **a**nd **i**nterface **c**onversion **a**pplication.
- It is a collection of Eclipse plugins which are integrated together in one tool for convenient use and to avoid version conflicts. At GENIVI it is available as Eclipse update-site (http://docs.projects.genivi.org/yamaica-update-site/yamaica/updatesite/).
- The main feature is the EnterpriseArchitect-to-Franca transformator which works in both directions (roundtrip).
- This feature allows it to use all the modeling features of EA to describe interfaces and to export then a Franca IDL description.

yamaica Roadmap

- The actual yamaica version 0.9.1 is very old. It is still based on Franca 0.8 and contains old code generators.
- The new version yamaica 15 will be published at the end of October 2015 (Franca 0.9.1, full roundtrip for the EA transformations, newest code generators integrated).
- yamaica 16 is planned at the end of the year (bugfixing).