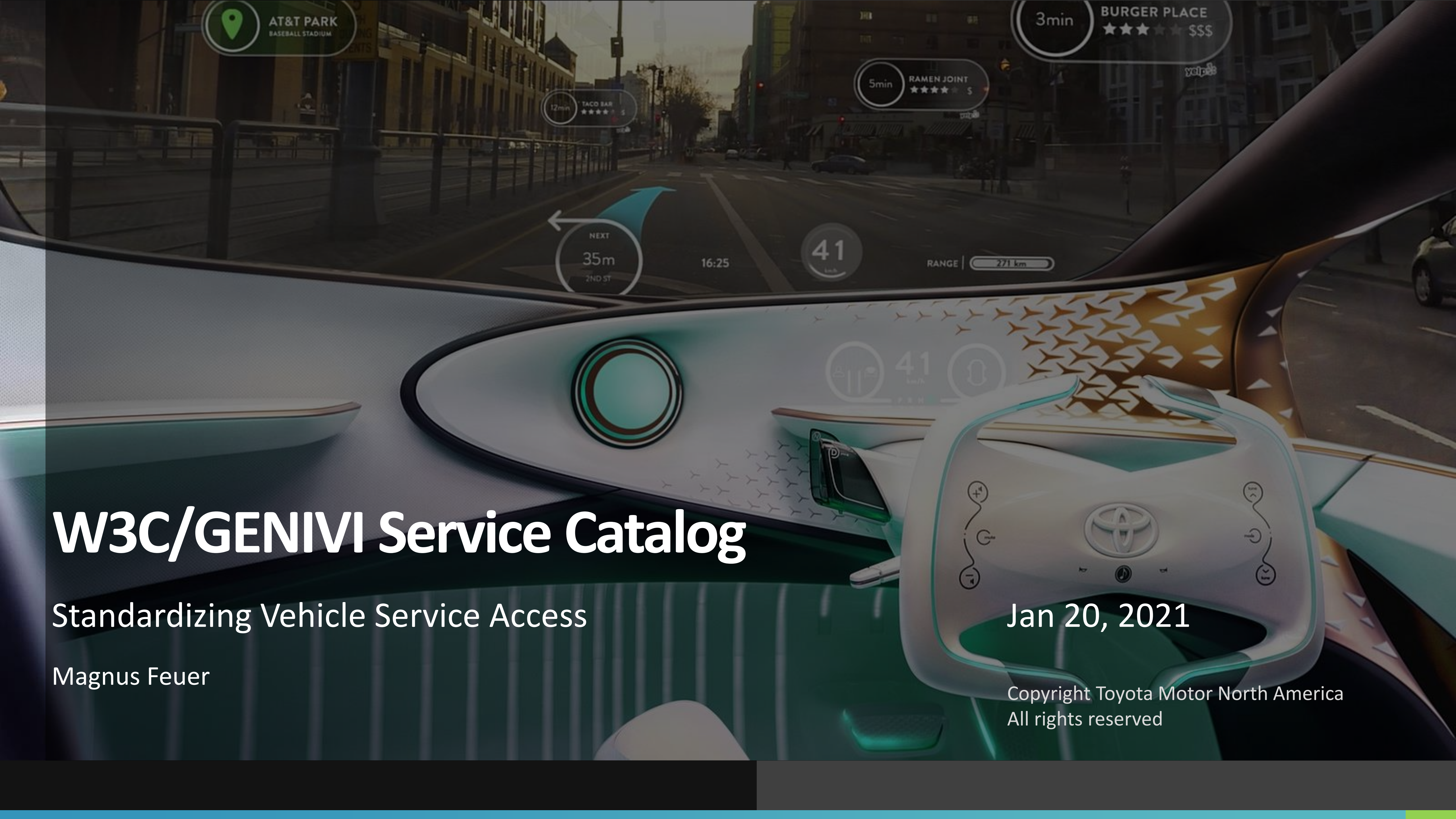


W3C/GENIVI Service Catalog

Standardizing Vehicle Service Access

Jan 20, 2021

Magnus Feuer

Copyright Toyota Motor North America
All rights reserved

●PUBLIC 公開

Background: W3C & GENIVI Standardization

Vehicle Signal Specification (VSS) established as industry standard

- Specifies 1500+ signals for HVAC, drivetrain, IVI, and other ECUs
- Used by several major OEMs, Tier-1s, and others
- W3C Automotive VISS ratified as standard to transmit signals between ECUs and OTA
- GENIVI maintains the signal catalog and tooling as open source

VSS is being extended with Remote Procedure Calls

- Create an open source service catalog and tools to control vehicle functions
- Create validators, code generation tools, and reference implementations
- Can be used for vehicle-internal communication and remote access

Copyright 2021 - Toyota Motor North America. All rights reserved

●PUBLIC 公開

Objectives

1. Specify a GENIVI catalog of standardized services to support in-vehicle and offboard service-oriented architectures

2. Specify a W3C protocol for vehicle access

3. Promote industry adoption of standardized services and protocols

Copyright 2021 - Toyota Motor North America. All rights reserved

●PUBLIC 公開

Market drivers to standardize services

- **OEM drivers**
  - Use standardized APIs to decouple solutions from vendor-specific technologies
  - Push for standard-compliance in RFIs & RFQs to ease side-by-side bid comparison
  - Use open source, standardized tools, and joint industry effort to create a higher starting point, allowing programs to focus resources on brand-differentiating experiences
- **Tier 1 & 2 drivers**
  - Implement standardized API to minimize program customization and maintenance, migrating toward off-the-shelf offers to OEMs
  - Portal/Host value-added services from third parties
- **Non-automotive drivers**
  - Manage mixed-asset fleets with minimum of system integration and maintenance
  - Widen and accelerate market for new 3rd party automotive services

Copyright 2021 - Toyota Motor North America. All rights reserved

Why Yet Another Standard?

- **Language and protocol agnostic**
  We need to try out different languages, protocols, and philosophies before we commit to something we want to standardize

- **Scale across 100s of interoperating services**
  Name spacing, interface imports, and API vs. Implementation version management are all needed in large-scale deployment

- **Lightweight**
  CLI oriented. Five minutes to running tutorial. Small, componentized codebase

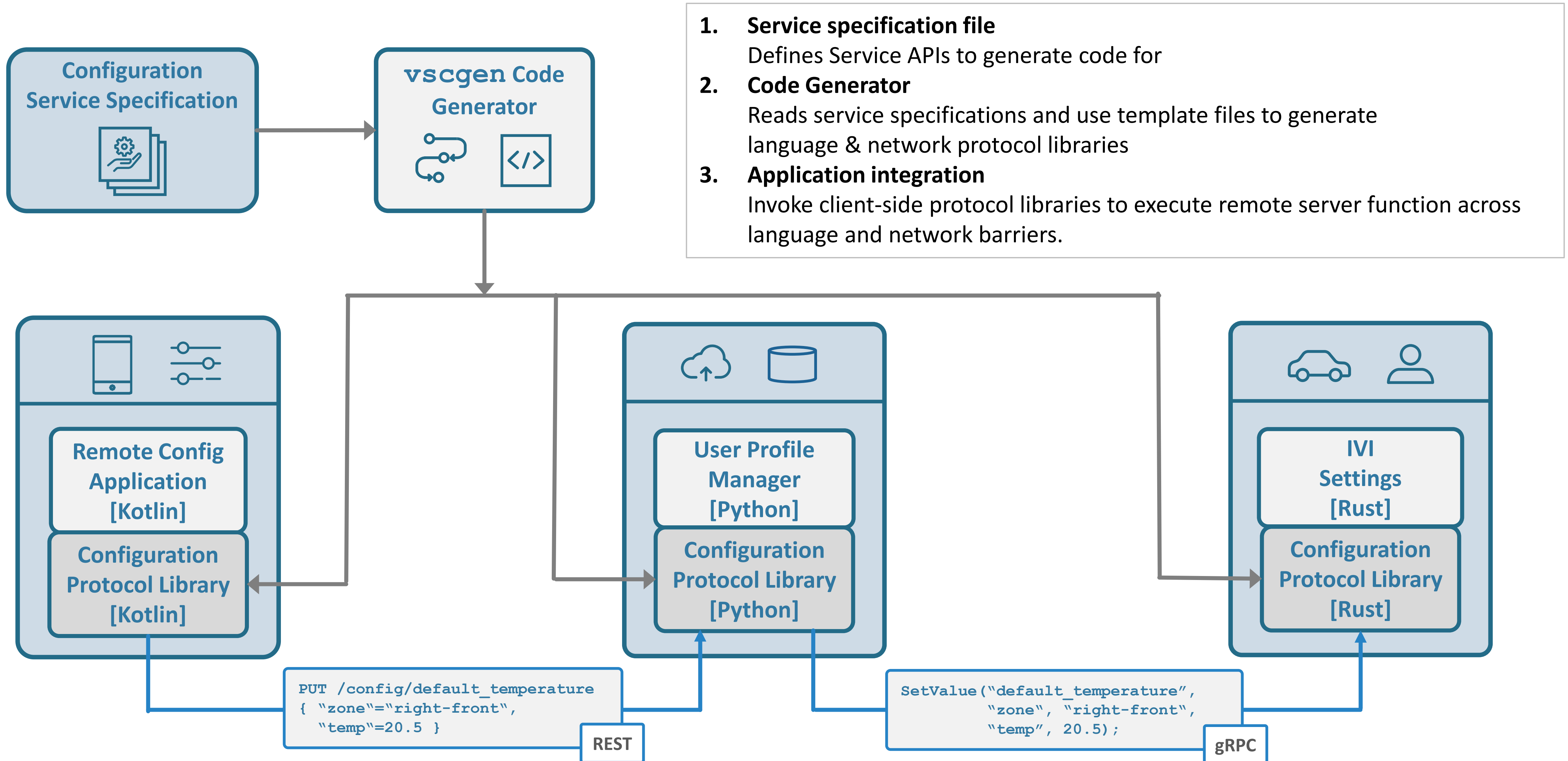
- **Cross-IDL portability**
  We need to be able to import (and export) existing IDL formats into a generic, easy-to-parse syntax while maintaining semantic equivalence

Copyright 2021 - Toyota Motor North America. All rights reserved

Usage examples

●PUBLIC 公開

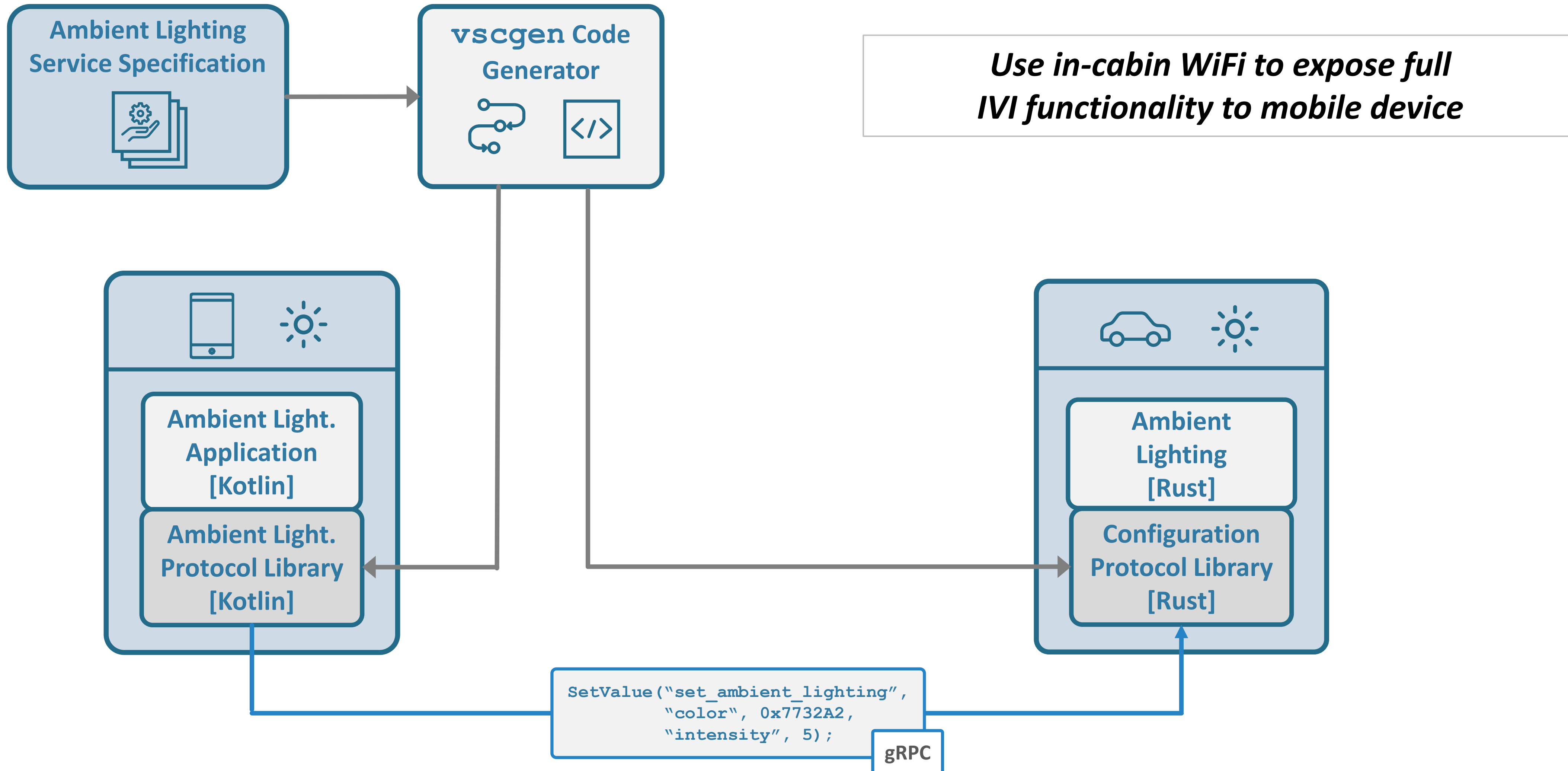
Remote Configuration Service Example



Copyright 2021 - Toyota Motor North America. All rights reserved

BYOD Ambient Lighting Service Example

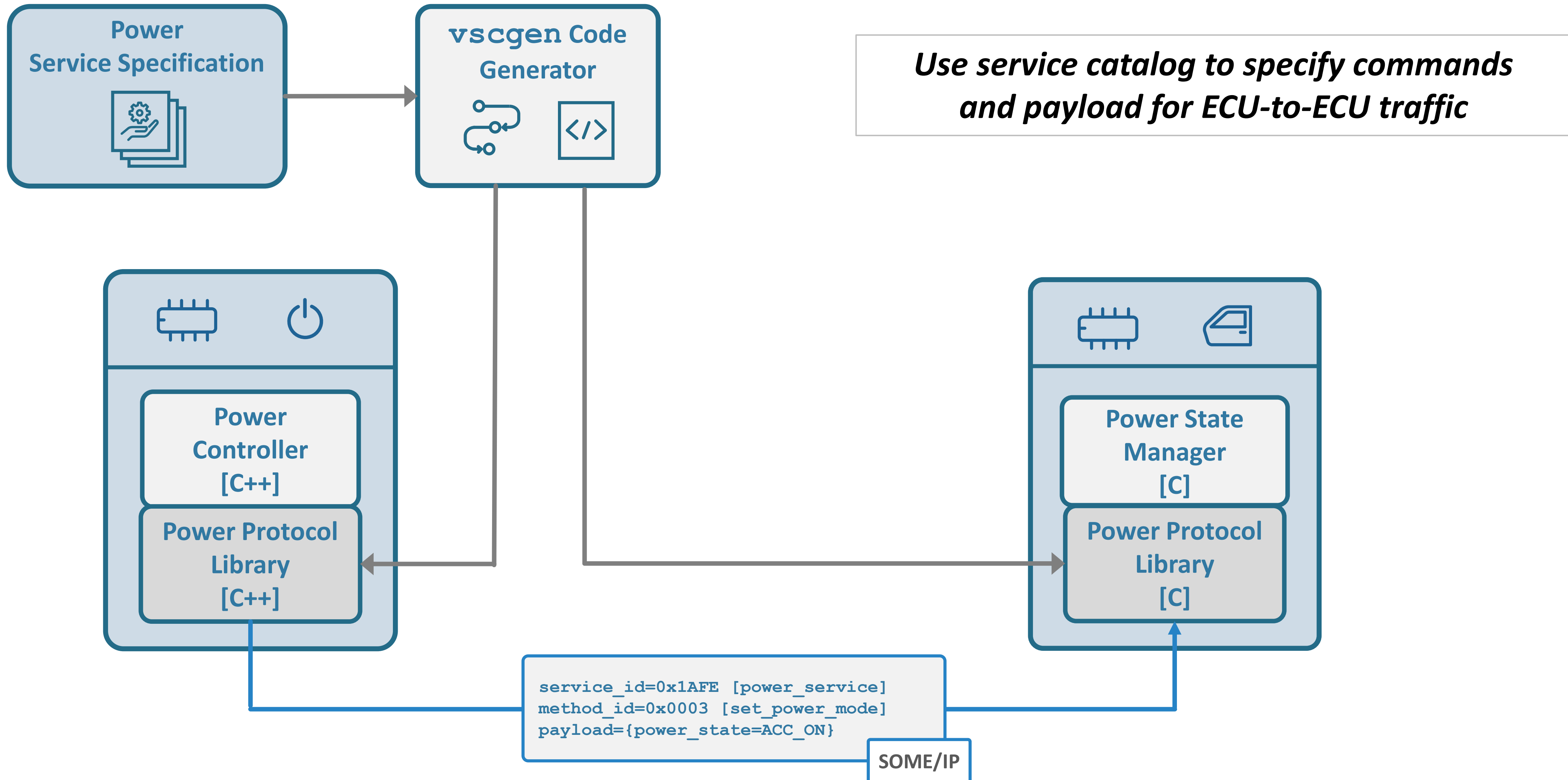
Use in-cabin WiFi to expose full IVI functionality to mobile device



Copyright 2021 - Toyota Motor North America. All rights reserved

●PUBLIC 公開

Onboard Service Example

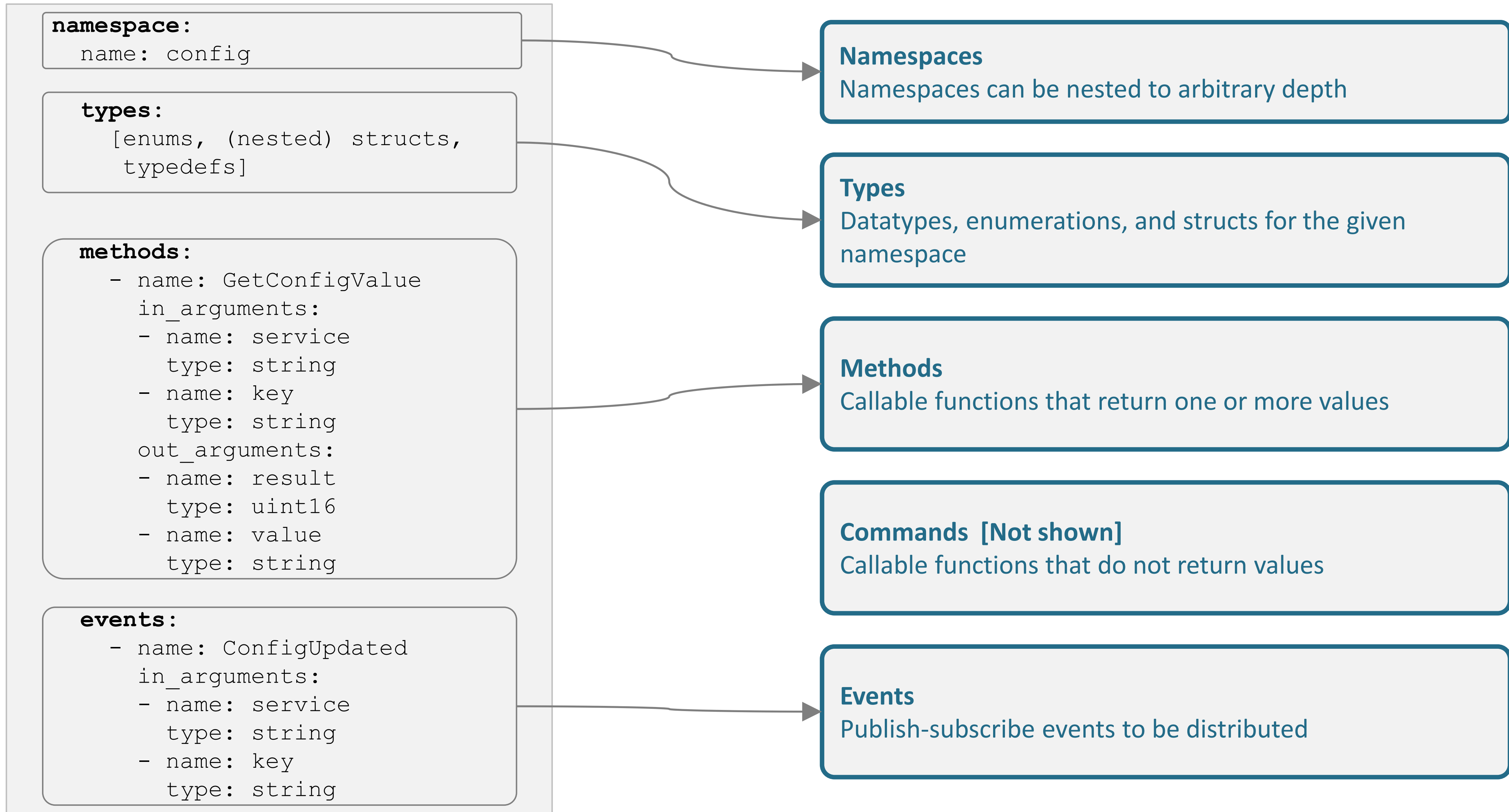


Use service catalog to specify commands and payload for ECU-to-ECU traffic

Copyright 2021 - Toyota Motor North America. All rights reserved

Service Catalog Format

Service Specification File Structure



Copyright 2021 - Toyota Motor North America. All rights reserved

Service Specification → Datatypes

```yaml
namespace:
  name: config
  types:
    - name: error_code
     options:
     - name: ok
       value: 0
     - name: not_found
        value: 1

    methods:
    - name: GetConfigValue
      in_arguments:
      - name: service
        type: string
      - name: key
        type: string
      out_arguments:
      - name: value
        type: string

    events:
    - name: ConfigUpdated
      in_arguments:
      - name: service
        type: string
      - name: key
        type: string
```

```python
class config():

  #
  # Datatypes
  #
  class config_ns():
    class error_code(Enum):
      ok = 0
      not_found = 1

  #
  # Server-side stub code
  #
  class config_server():
    def GetConfigValue(self, service, key):
      return self._impl.GetConfigValue(service, key)

    def ConfigValueUpdated(self, sevice, key):
      self._dbus.emit("ConfigValueUpdated", service, key)
  #
  # Client-side stub code
  #
  class config_client():
    def GetConfigValue(self, service, key):
      return self._dbus.GetConfigValue(service, key)

    def ConfigValueUpdated(self, sevice, key):
      self._impl.process_signal("ConfigValueUpdated", service, key
```

Copyright 2021 - Toyota Motor North America. All rights reserved

Service Specification → Methods

```
namespace:
  name: config
  types:
    - name: error_code
     options:
     - name: ok
       value: 0
     - name: not_found
        value: 1

    methods:
    - name: GetConfigValue
      in_arguments:
      - name: service
        type: string
      - name: key
        type: string
      out_arguments:
      - name: value
        type: string

    events:
    - name: ConfigUpdated
      in_arguments:
      - name: service
        type: string
      - name: key
        type: string
```

```
class config():

  #
  # Datatypes
  #
  class config_ns():
    class error_code(Enum):
      ok = 0
      not_found = 1

  #
  # Server-side stub code
  #
  class config_server():
    def GetConfigValue(self, service, key):
      return self._impl.GetConfigValue(service, key)

    def ConfigValueUpdated(self, sevice, key):
      self._dbus.emit("ConfigValueUpdated", service, key)
  #
  # Client-side stub code
  #
  class config_client():
    def GetConfigValue(self, service, key):
      return self._dbus.GetConfigValue(service, key)

    def ConfigValueUpdated(self, sevice, key):
      self._impl.process_signal("ConfigValueUpdated", service, key
```

Copyright 2021 - Toyota Motor North America. All rights reserved

Service Specification → Events (pub/sub)

```yaml
namespace:
  name: config
  types:
    - name: error_code
     options:
     - name: ok
       value: 0
     - name: not_found
        value: 1

    methods:
    - name: GetConfigValue
      in_arguments:
      - name: service
        type: string
      - name: key
        type: string
      out_arguments:
      - name: value
        type: string

    events:
    - name: ConfigUpdated
      in_arguments:
      - name: service
        type: string
      - name: key
        type: string
```

```python
class config():

  #
  # Datatypes
  #
  class config_ns():
    class error_code(Enum):
      ok = 0
      not_found = 1

  #
  # Server-side stub code
  #
  class config_server():
    def GetConfigValue(self, service, key):
      return self._impl.GetConfigValue(service, key)

    def ConfigValueUpdated(self, sevice, key):
      self._dbus.emit("ConfigValueUpdated", service, key)
  #
  # Client-side stub code
  #
  class config_client():
    def GetConfigValue(self, service, key):
      return self._dbus.GetConfigValue(service, key)

    def ConfigValueUpdated(self, sevice, key):
      self._impl.process_signal("ConfigValueUpdated", service, key
```

Copyright 2021 - Toyota Motor North America. All rights reserved

Deployment file structure

```
namespaces:
  - name: config
    dbus_interface: org.genivi.config
    methods:
    - name: GetConfigValue
      dbus_name: get-config-value
```

- Extends service specification file with language & protocol-specific information that must be known at build time
- Values used by code generator template
- Does not replace runtime configuration files

Examples:

- Name → Method ID mapping in SOME/IP
- Protocol conversions (little-endian, etc)
- Method and argument renaming to comply with language syntax (`user-id` → `user_id`)

Copyright 2021 - Toyota Motor North America. All rights reserved

●PUBLIC 公開

Importing global definitions and interfaces

```
# global-errors.yml
namespace:
  name: global_error
   - types:
     name: result
     type: enumeration
     options:
     - name: ok
       value: 0
```

```
 namespace:
  name: configuration
    import:
    - file_name: global-errors.yml
    - file_name: diagnostic_interface.yml

    methods:
    - name: GetConfigValue
      out_argument: global_error.result
```

- Imports commands, methods, events, and datatypes
- Attached to the local namespace
- Generated code contains all imports
- Allows services to import globally defined interfaces that have to be implemented (life cycle management, diagnostics, etc)

Copyright 2021 - Toyota Motor North America. All rights reserved

Template files

```
## DBUS introspection XML file generation

<interface name='$iface.dbus_interface'>

#for $cmd in $iface.get('commands', [])
  <method name='$cmd.name'>

  #for $inarg in $cmd.get(in_arguments, [])
    <arg
      type='$dbus_support.
            convert_vsc_type_to_dbus($inarg)'
      name='$inarg.name'
      direction='in'
    />
  #end for

  </method>
#end for
```

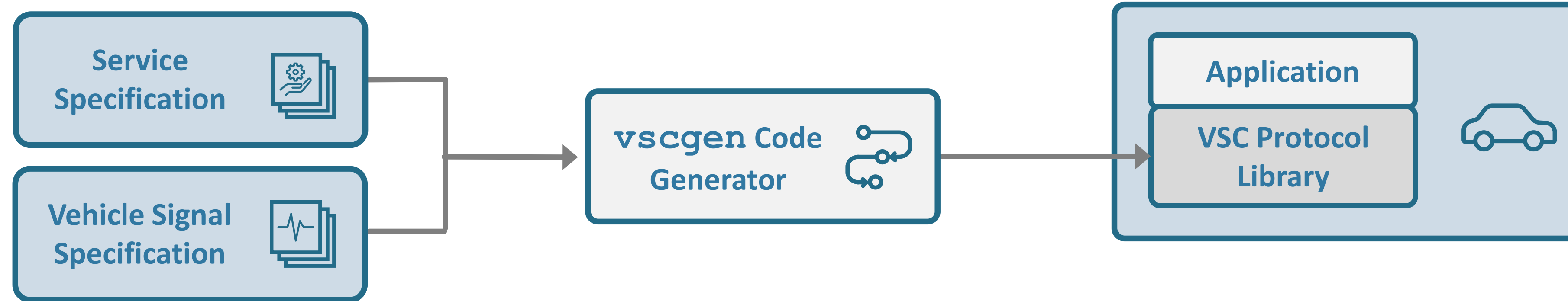
- Uses Cheetah Python template library
- Each template generates code for specific language protocol stack combination
- Replaces tokens in template file with elements from service file parse tree
- Template for Rust/DBUS and Python/DBUS supported

Copyright 2021 - Toyota Motor North America. All rights reserved

Vehicle Signal Spec Integration

Three options to discuss

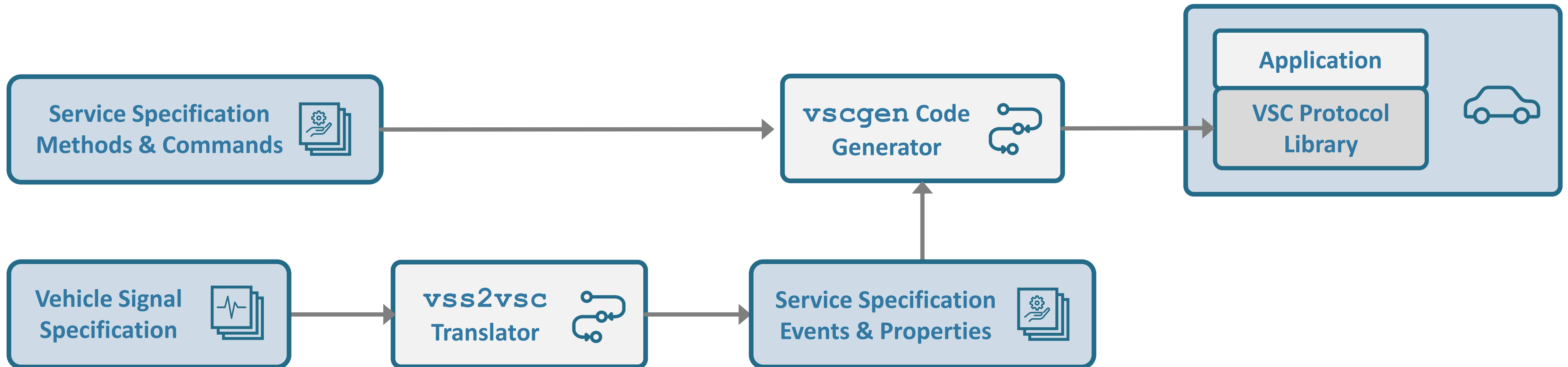
VSS Integration Option 1 – Single-tool generation



- **vscgen tool ingests both signal and service specs**
  A single library is generated to handle both signals and RPCs
- **Signals are exposed as properties and/or events in VSC**
- **No direct relation to VISS**

Copyright 2021 - Toyota Motor North America. All rights reserved

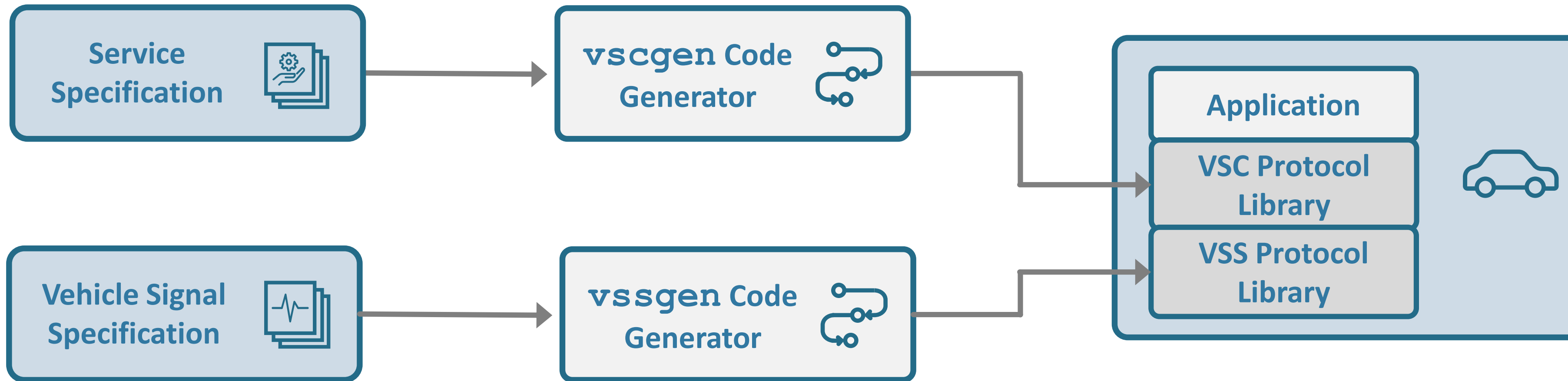
VSS Integration Option 2 – Signal Spec → Service Spec translation



- **Special tool to convert VSS spec to VSC format**
- **Signals are translated properties and/or events in VSC**
- **No direct relation to VISS**

Copyright 2021 - Toyota Motor North America. All rights reserved

VSS Integration Option 3 – Maintain separation



- **VSC and VSS/VISS are not aware of each other**
- **Application is responsible for integrating libraries and generated code**

Copyright 2021 - Toyota Motor North America. All rights reserved

Next steps

- **Agree on name**
- **Open Source tooling**
  - MPLv2 / CC-BY-SA 4.0 licensing in progress
  - To be hosted by GENIVI
- **Agree on if/how we want to integrate Vehicle Signal Specification**
- **Create initial set of services**
  - Service proposals needed

Copyright 2021 - Toyota Motor North America. All rights reserved

Thank you

Magnus Feuer
magnus.feuer@toyota.com
+1-949-294 7871