TUS SDK 010 Demonstration

Agenda

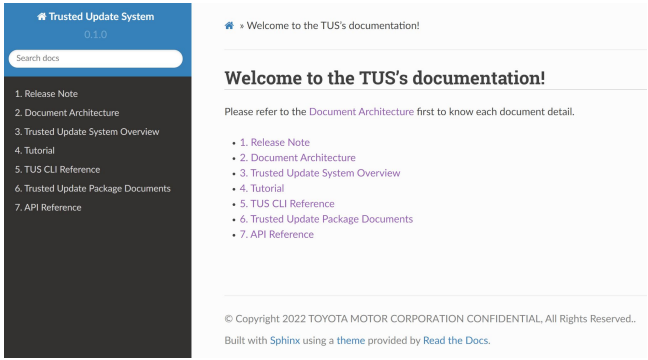
1. Abstraction of the TUS SDK 010
2. TUS structure and behavioral model
3. Demonstration

Abstraction of the TUS SDK 010

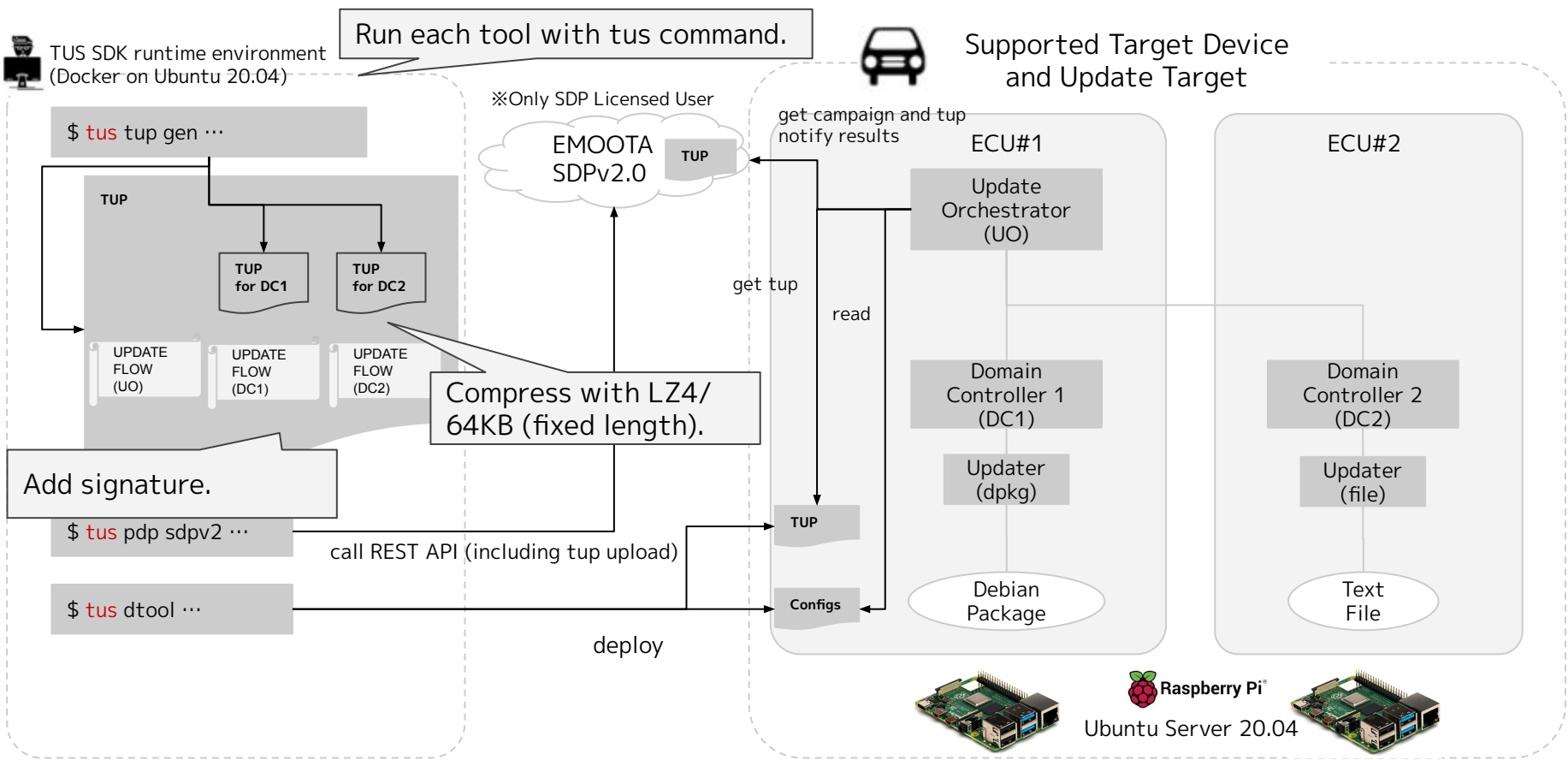
Abstraction of the TUS SDK 010

TUS SDK 010 provides the following items.

1. Source code for components running on Target Device
   - TUS Runtime (Update Orchestrator(UO), Domain Controller(DC), Updater)
   - TUP Parser Library
2. Tools that can be adapted to SDK user's development stage and process (Host-Tools)
   - tus command (command to control each tool)
   - tup generator (TUP generation tool)
   - deployment tool (tool to deploy Configuration and Package to target)
   - EMOOTA SDPv2 operator (only licensed user)
3. Related Documents
   - Release Notes
   - Tutorial
   - TUS Overview
   - TUP Design and Specification
   - API Document



Supported structure of the TUS SDK 010



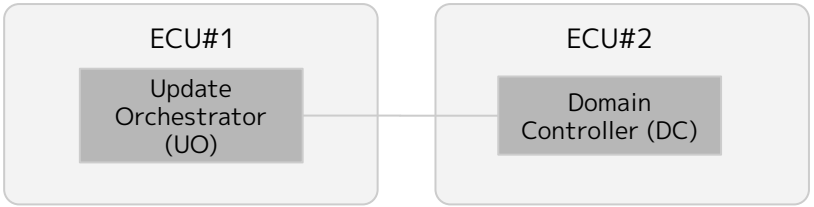
Note. TUP encryption is not supported yet.

TUS structure and behavioral model

TUS structure and behavioral model - Premise

- Relationships among each software module will be explained using static structure diagram and specific sequences such as boot process, Campaign inquiry, UPDATEFLOW execution by UO/DC and state control.

- Structure of UO/DC is assumed to be the following.
  (This is simplified structure of the SDK 010 for explanation.)

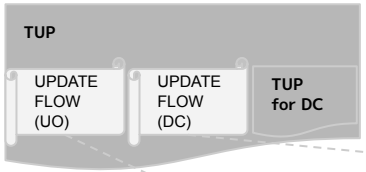


Software Module Structure

TOYOTA InfoTech

Lua Extension | Pure Lua Module | Process

Both UO Main and DC Main are assumed to be installed in the system. (UO/DC Main is a Native implementation and liblua is used for lua runtime environment.)



UPDATEFLOW controls update process with Finite State Machine (FSM). (Detail information is described later.)

Update Orchestrator(UO)

UO Main(main.lua)

providers | domains | tup_parser | gRPC | protobuf

execute

UPDATE FLOW(UO) in TUP

providers | domains | tup_parser | fsm | gRPC | protobuf

expose

httpd

TUP

Domain Controller(DC)

DC Main(lua)

gRPC Server

Notification Queue

fsm | tup_parser | gRPC | protobuf

LuaState switch

UPDATE FLOW(DC) in TUP

Updater(file) | fsm | tup_parser | Updater(dpkg) | gRPC | protobuf

Control Pass
(DC structure, state inquiry, state update, etc.)

Data Pass
(Getting TUP and UpdateScripts)

Software Module Structure

TOYOTA INFOTECH

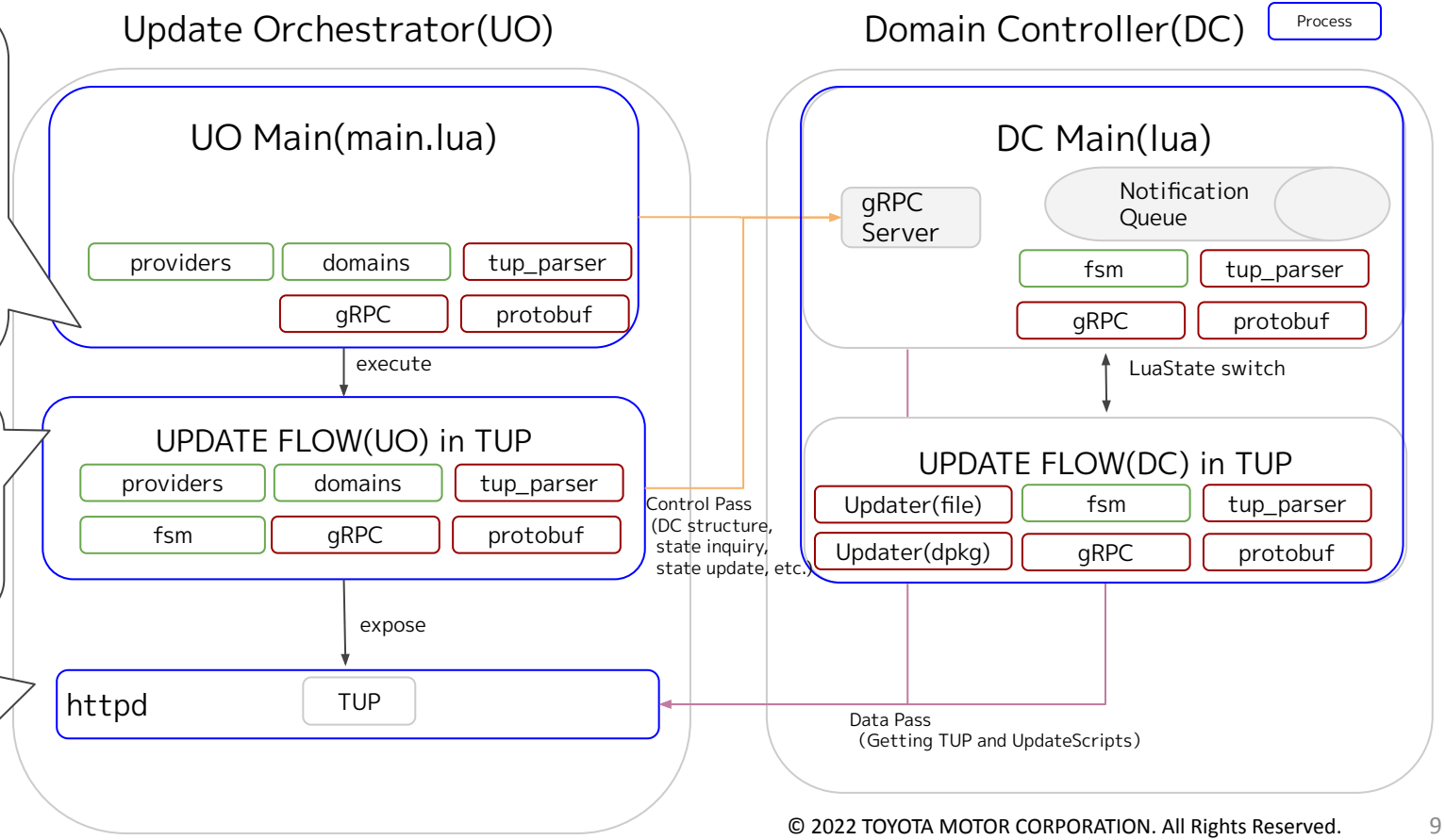
Lua Extension | Pure Lua Module | Process

providers
Functions related to update information such as campaign check and result notification.
domains
Domain management function (getting information such as vehicle-config)
tup_parser
tup parser library
(Domain configuration is described in detail later)

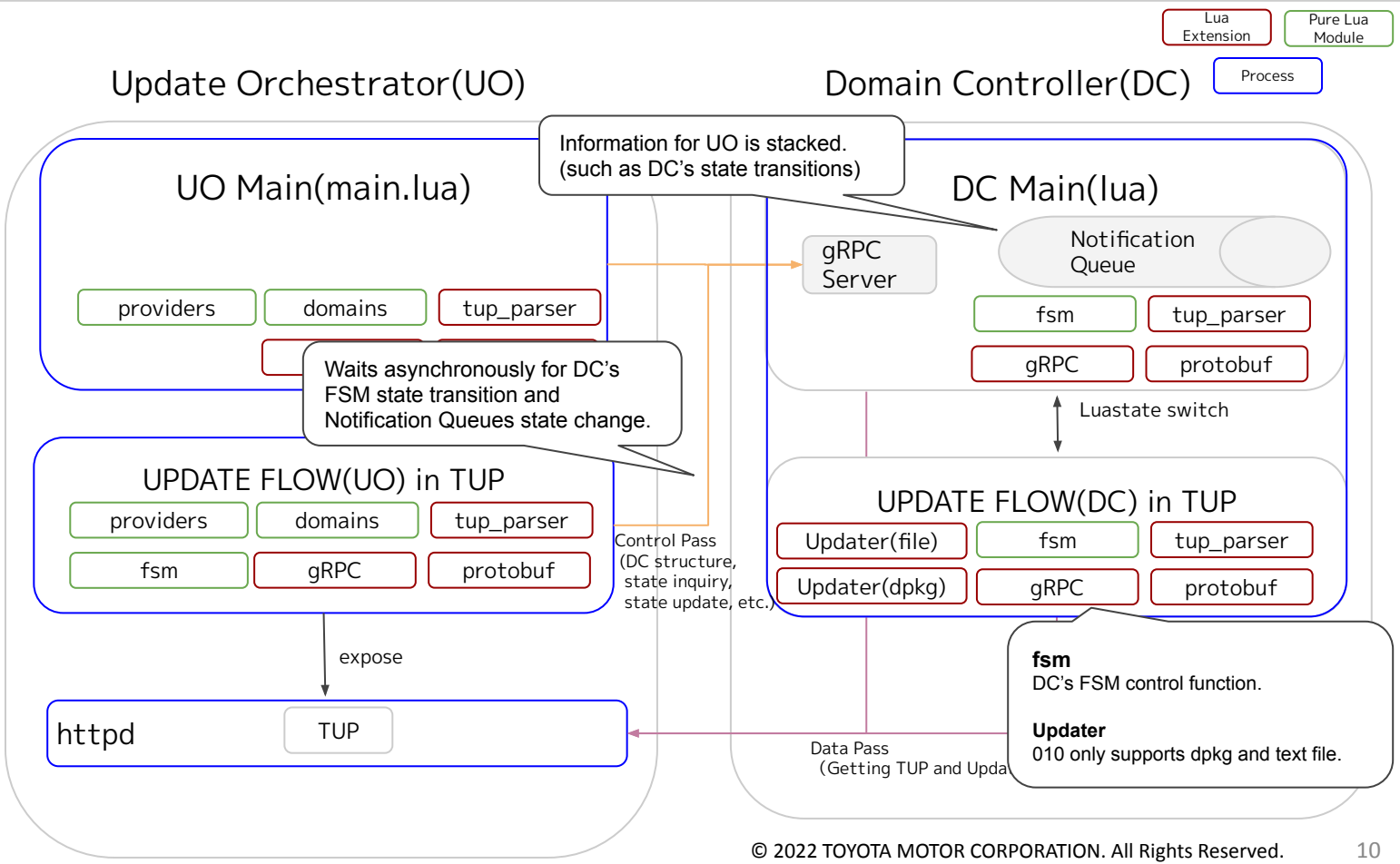
fsm
UO's FSM control function

domains
Domain management function and DC update control such as state transition of DC.

httpd
TUP can be obtained by DC over HTTP.
（010 uses the system's httpd and TUP is written on the file system.）



Software Module Structure



Configuration file on Target Device

Update Orchestrator

File path ※1	Description
config	Root of the configuration files. httpd configuration and provider_path and domain_path of files below are included.
<provider_path>/config	provider(TUP server) specific configuration file. （010 only supports for EMOOTA SDPv2 and Local File System.） (e.g.) In case of Local File System, file path of the TUP is included in this.
<domain_path>/config	File that has each domain name and path to each domain's configuration file.
<each_domain_config_path>	File that has each domain's information. (e.g.) gRPC Server connection information

※1. Please refer to the SDK Documents for the file path details of the SDK 010.
※UO's configuration files can be deployed with`tus dtool…`.

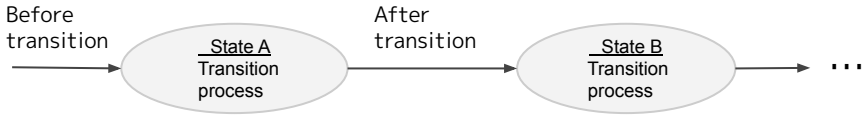
Domain Controller

File path ※2	Description
version/<target_name>	File that has version of UpdateTarget and UpdateTarget Group ※SDK010 only supports for file format.

※2. Relative path from executable binary (domain_controller)

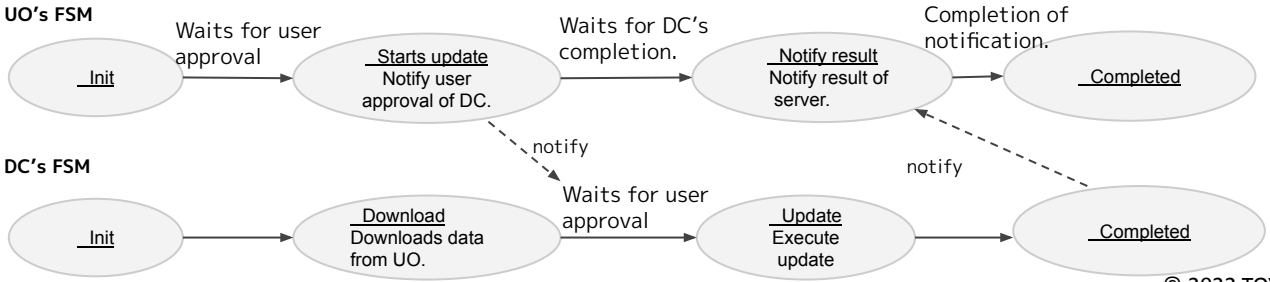
Abstraction of UPDATE FLOW

- TUP contains UPDATE FLOW for UO and DC.
  - TUP Generator generates UPDATE FLOW from metadata(json) automatically.
- Finite State Machine is adopted as programming model.
  - Update process can be written using state, pre-transition process, transitioning process, post-transition process and rollback process (Called in reverse order of addition order)



  - FSM state is updated by external event such as DC's FSM state transition and User operations.
- FSM state and transition conditions need to be determined depending on information such as combination of Update Targets (though it is fixed in the SDK 010).

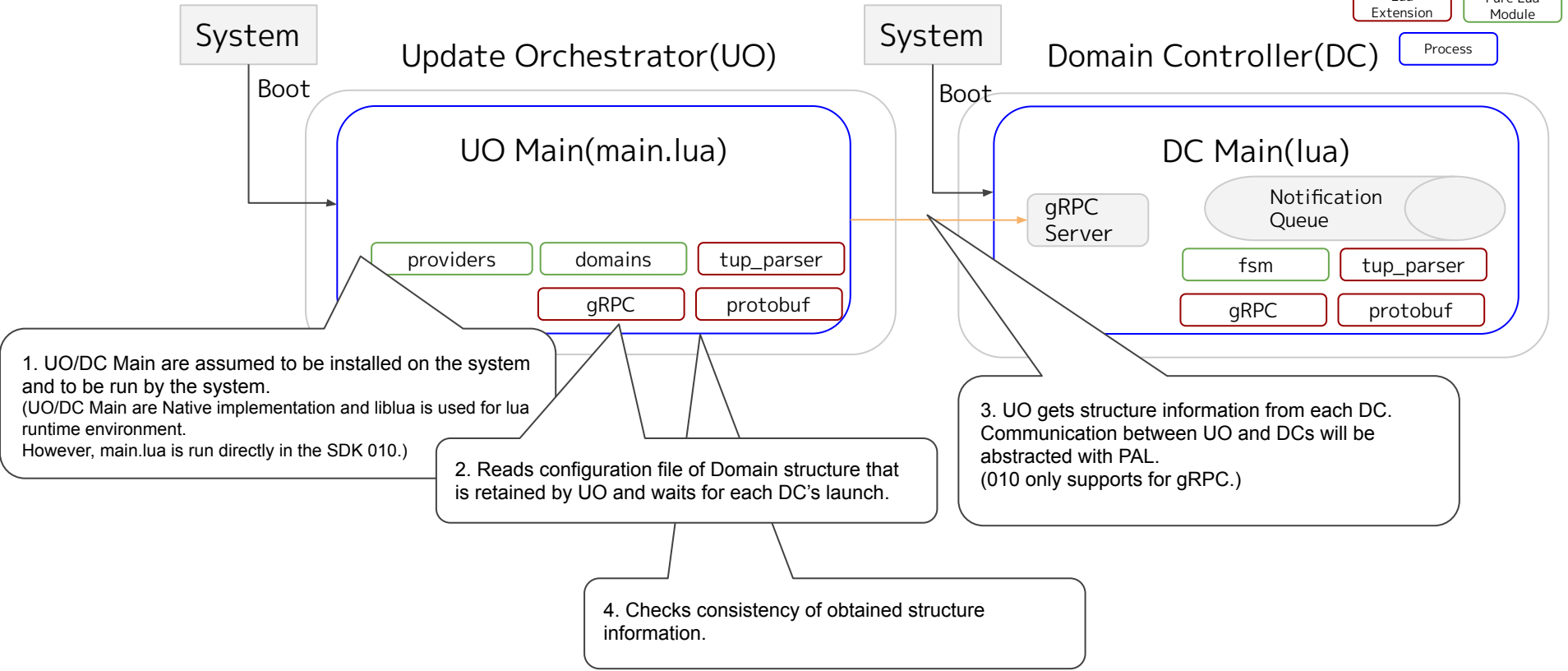
Example



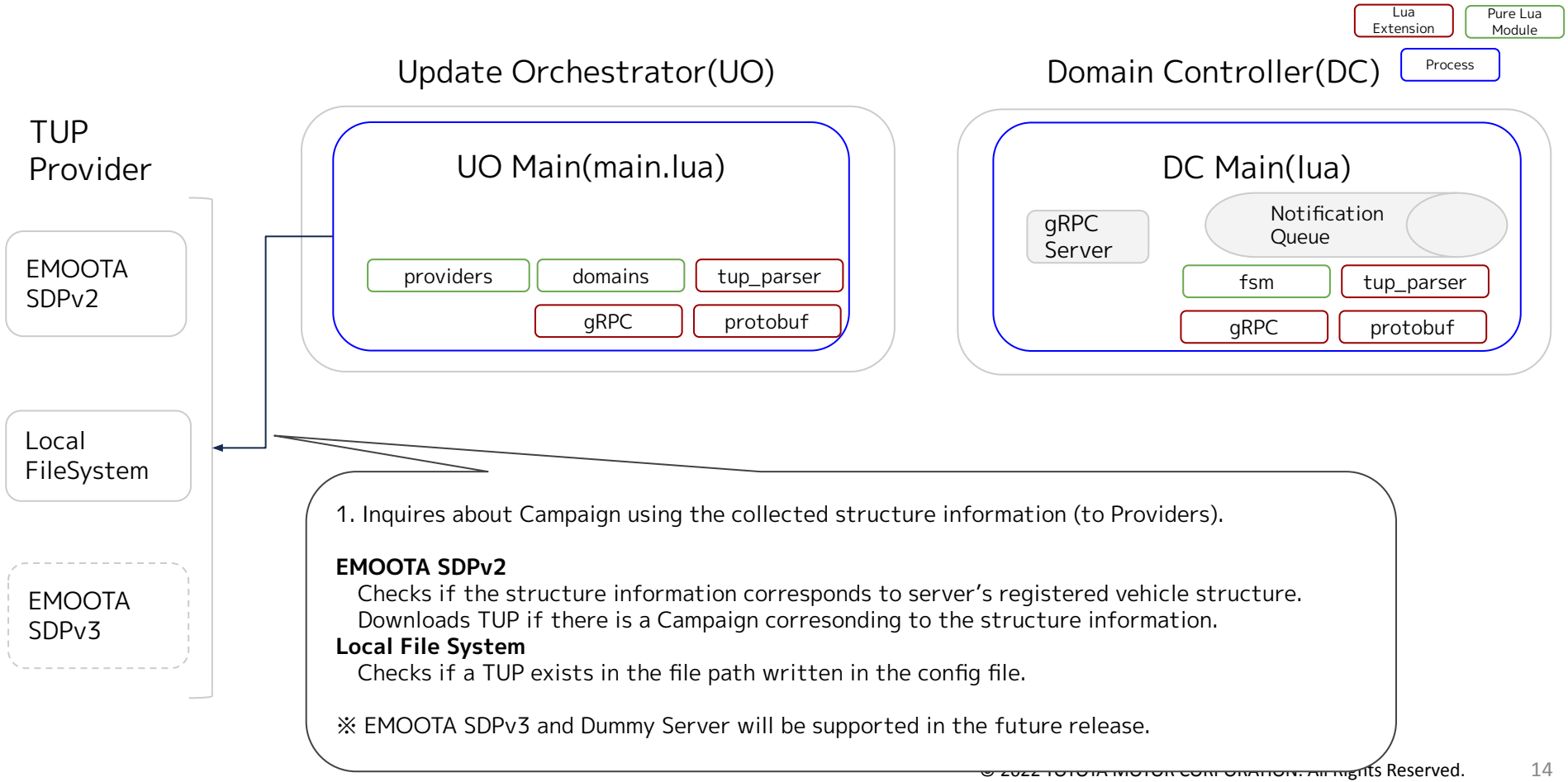
UpdateScript of DC side's FSM (only excerpts related parts)

```
dc.fsm:add_transition({
   from = nil, -- from initial
   to = "downloaded",
   action = download,
})
dc.fsm:add_transition({
   from = "downloaded",
   to = "updated",
   cond = check_dc_updated,
   action = update,
})
```

Boot process



Campaign inquiry



Run UPDATE FLOW for UO

TOYOTA INFOTECH

Lua Extension | Pure Lua Module | Process

1. Download **only necessary part** to run UPDATE FLOW for UO.
(010 downloads the whole TUP.)
2. Processes such as validation will be done.

Provider

EMOOTA SDPv2

Local FileSystem

EMOOTA SDPv3

date Orchestrator(UO)

UO Main(main.lua)

TUP

providers | domains | tup_parser | gRPC | protobuf

execute

UPDATE FLOW(UO) in TUP

providers | domains | tup_parser | fsm | gRPC | protobuf

Domain Controller(DC)

DC Main(lua)

gRPC Server

Notification Queue

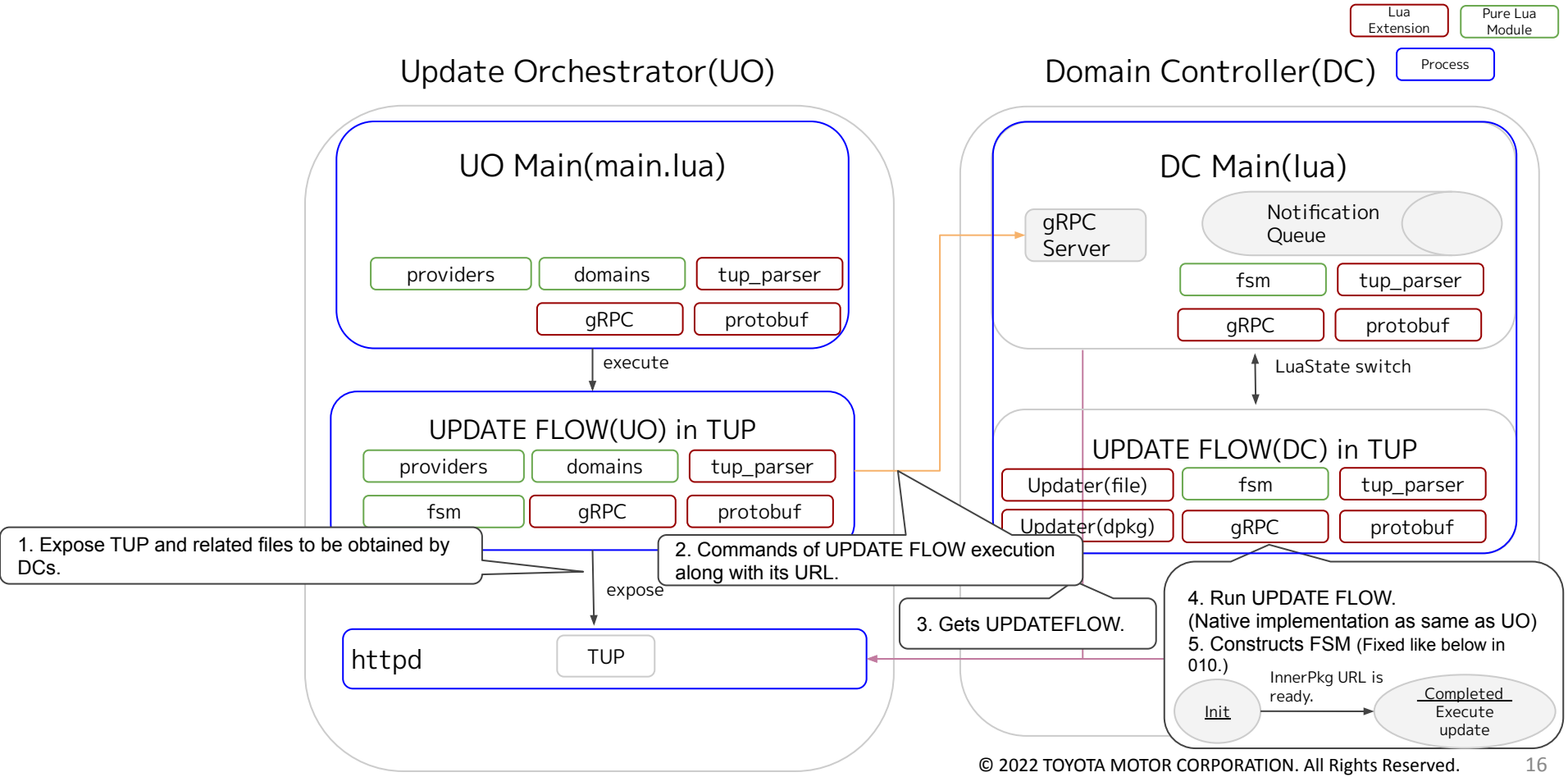
fsm | tup_parser | gRPC | protobuf

3. Run as other process for the reliability.
(Native implementation having Lua runtime environment via libua)

4. Starts worker thread to get DC states asynchronously.
5. Reads UPDATE FLOW and constructs FSM.
(FSM is fixed in the SDK 010.)

Init → Updating → (All the DCs completed) → Completed

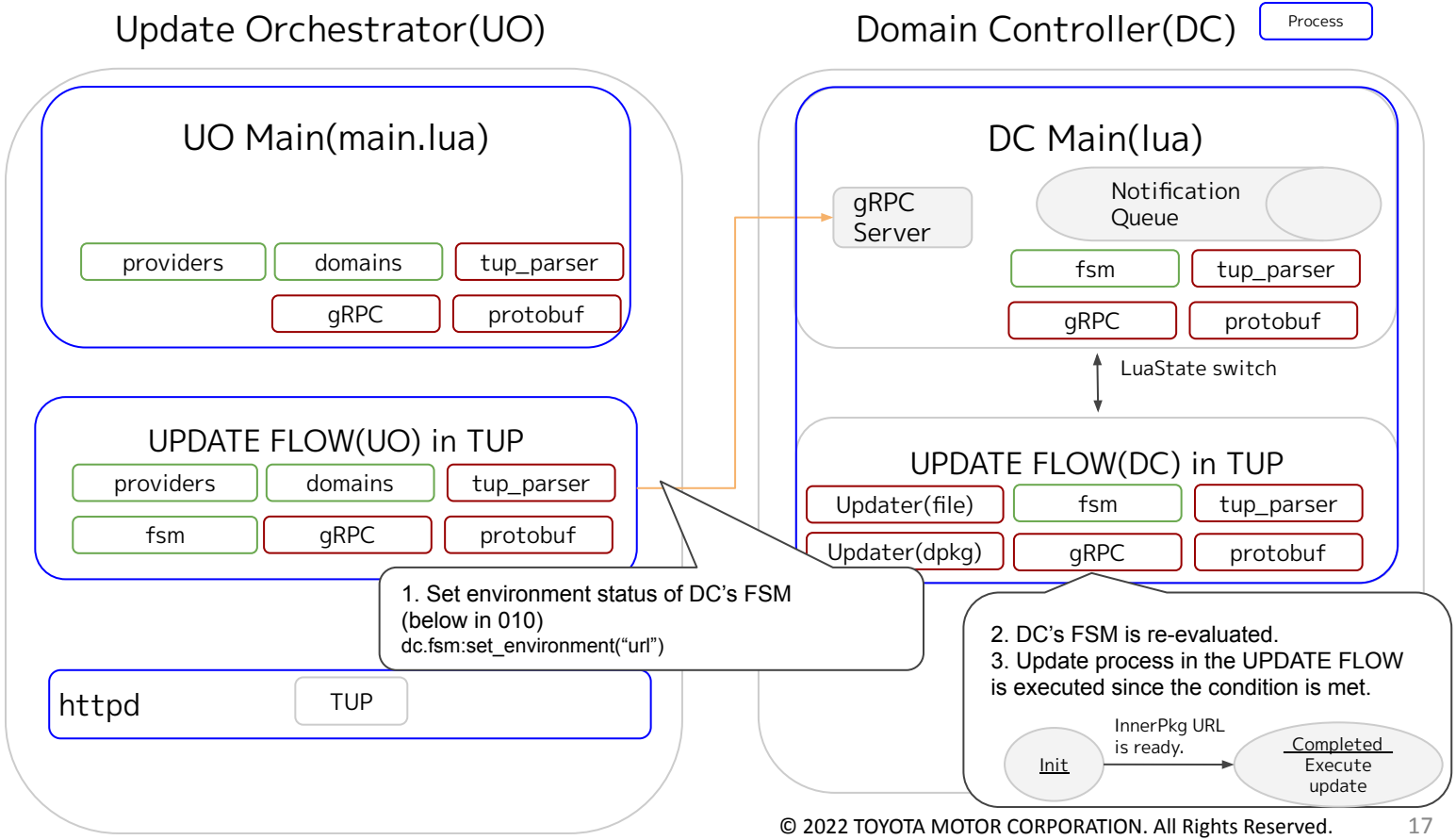
Run UPDATE FLOW for DC



DC’s environment status update from UO

TOYOTA INFOTECH

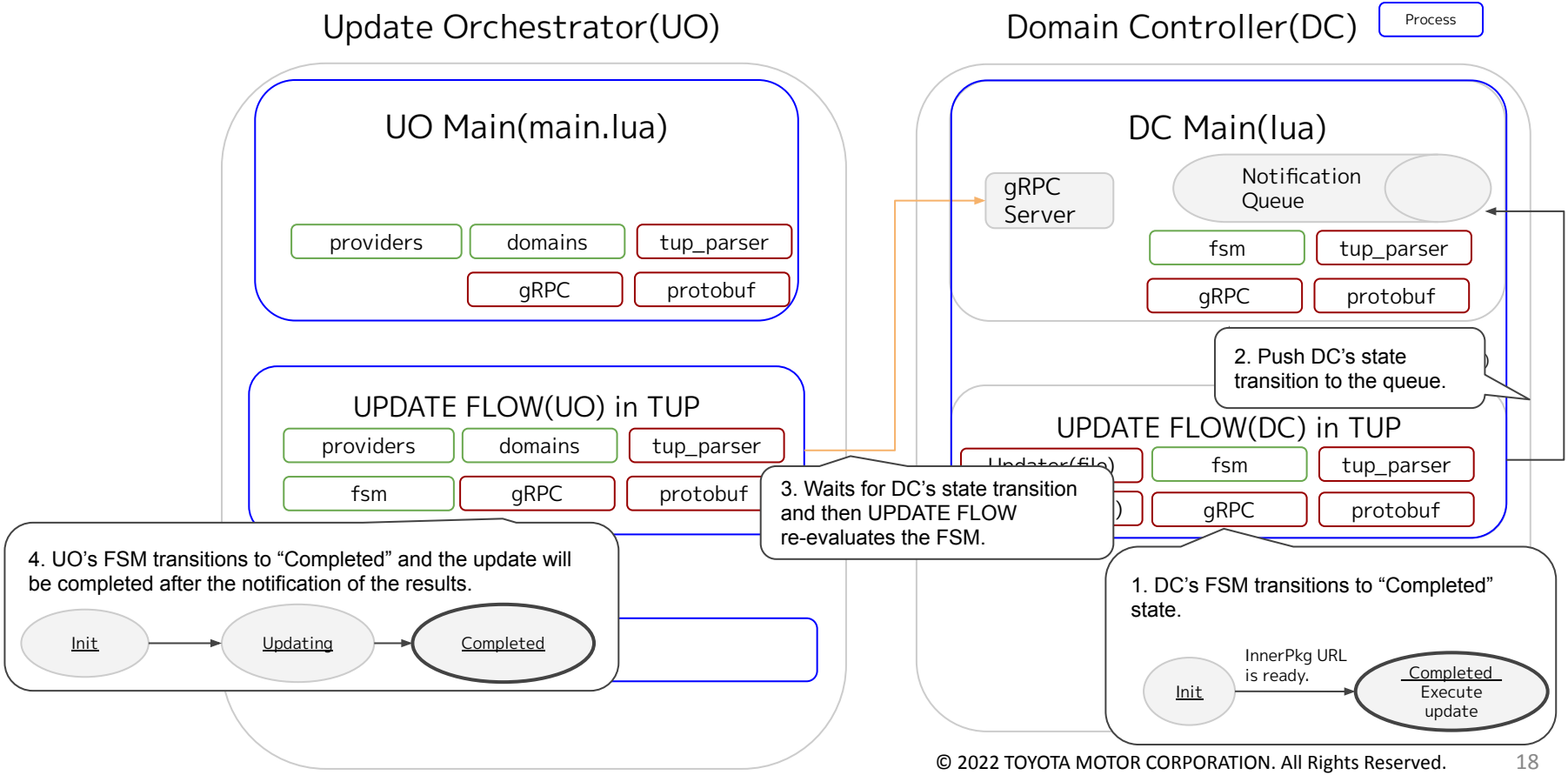
Lua Extension | Pure Lua Module | Process



Notification of state transition from DC to UO

TOYOTA INFOTECH

Lua Extension | Pure Lua Module | Process

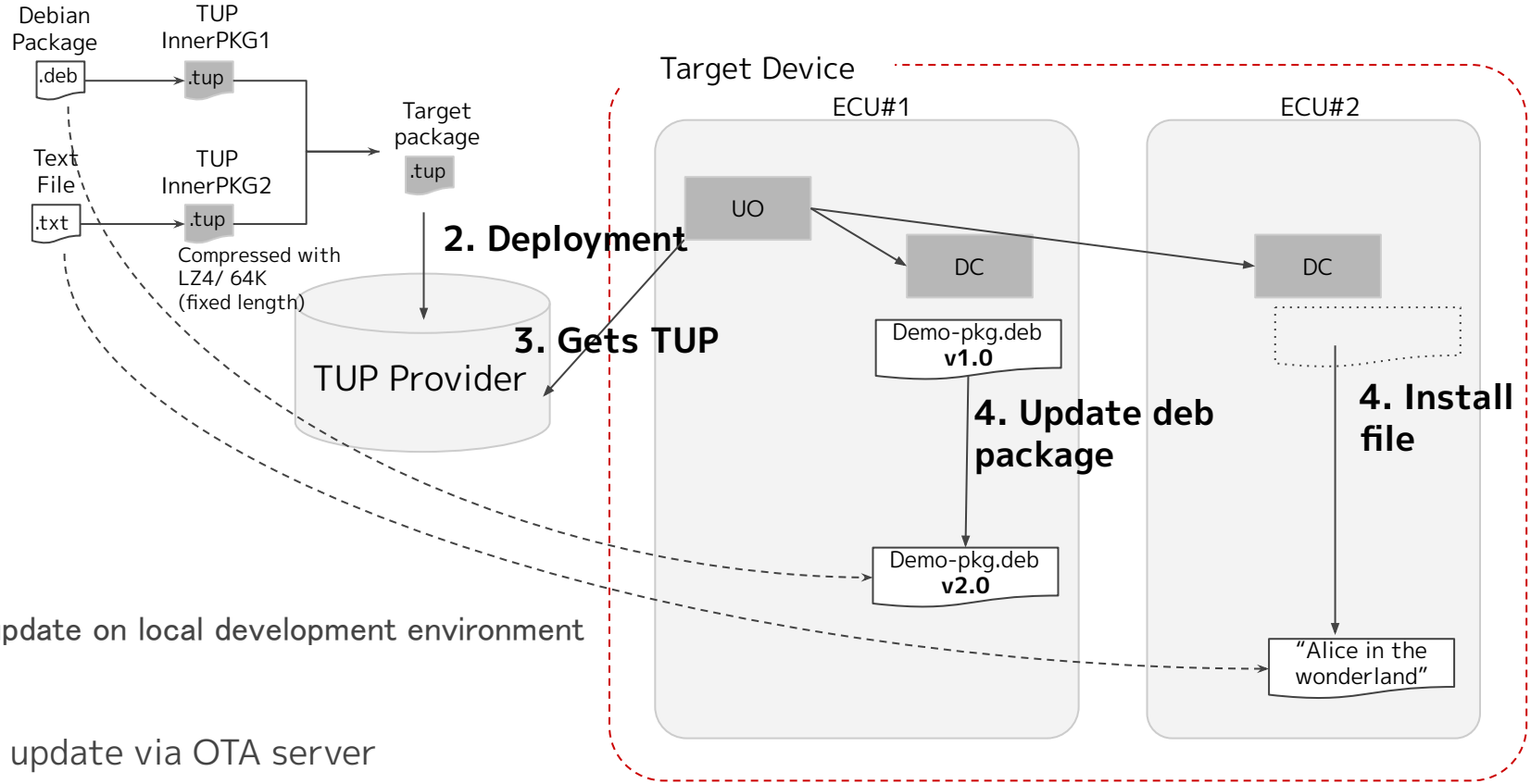


Demonstration

Demo abstract explanation of the demonstration

1. TUP Generation

※ Refer to P.5 for the details of the target structure.



Demo #1

Software update on local development environment

Demo #2

Software update via OTA server

Demo#1 Software update on local development environment

TOYOTA INFOTECH

Scenario of software update using the Target Device's Local FileSystem.

TUS SDK runtime environment
(tus command)

1. TUP Generation

.tup

2. Prepare configs and deployment

configs

3. TUP deployment to Local FS

4. Run processes (UO and DCs) on the Target Device.

※ This is done manually to synchronize explanation and process progress in this demonstration. (They will be run by the system in products.)

Target Device

ECU#1

Update

Local FS

Debian package

ECU#2

Install

Text File

Demo#1 Software update via OTA server

※EMOOTA SDP License is required.