Product Development

FNV2 Programmer Guide System Service Descriptor

Version 1.1

Version Date: September 6, 2019

UNCONTROLLED COPY IF PRINTED

FORD CONFIDENTIAL

Table of Contents

1 Introduction

Service Manager uses data provided in a descriptor file to create Services, FTCP Cpmmand Hand;ers and ACL databases to

1. Make FTCP command handlers discoverable by FCI
2. Allow SOA clients to communicate to each other.

2 Descriptor content

Descriptor XM schema description.

2.1 Service ID

Service ID must be an FNV wise unique string. Service developers are supposed to add service IDs to a database to make then discoverable by other developers.

2.2 Service Group ID

Service group ID attribute allows to group services to allow consumers to use a single <soa-service-request> element to request FULL access to all services in the group.

Example:

FCI declares "FNV_ECG_FCI" service group, FCI test app requests access to the group.

Service group ID should be an FNV wise unique string.

Services from different components cannot belong to the same group.

2.3 FCI Client

Example

FCI client "id" should match "uniqueServiceId" parameter of FciClient::createInstance() API.

2.4 Access Type

Service Manager currently do not support "RPC" and "DATA" access type. It ignores the attribute and grants "FULL" service access. The feature support is coming soon. Descriptor can declare different access types.

2.5 Consumer "id" attribute vs Consumer "endpoint" element.

"id" attribute and "endpoint" element are mutually exclusive. A descriptor cannot contain both. If "id" attribute explicitly provided, Service Manager generates a topic: ""SERVICES/RESPONSE/**ECU_ID**/**UID**/**CONSUMER_ID**".

where.

ECU_ID is an id of the consumer's ECU

UID is a user id of the consumer's client (process)

CONSUMER_ID is the "id" attribute value

This topic pattern allows to reduce a probability of duplicate topics.

"id" attribute can be leveraged if

1. Consumer topic follows the pattern
2. Consumer was created using the next SOA API. "id" attribute value should match the "consumerId" parameter value.

```
        /**
         * Factory method for creating a SoaConsumer object
         * @param messageManager a shared pointer to a SoaMessageManager instance
         * @param serviceDirectoryManager a shared pointer to a
SoaServiceDirectoryManager instance
         *                                can be a nullptr if the consumer does not
need access to Service Manager Services
         *                                Supported Service Manager APIs:
         *                                  getCommandHandlers
         *                                  getCommandHandlersAsync
         *                                Not supported Service Manager APIS
         *                                  requestServiceStatusCached
         *                                  requestServiceStatusAsync
         *                                  requestServiceStatus
         *                                  subscribeToServiceStatusUpdateAsync
         *                                  subscribeToServiceStatusUpdate
         *                                  unsubscribeToServiceStatusUpdateAsync
         *                                  unsubscribeToServiceStatusUpdate
         *                                  startService
         *                                  startServiceAsync
         * @param consumerId a consumer ID. Must be unique string for the component
         * @return  a shared pointer to a new SoaConsumer instance
         *          a nullptr if
         *              1. messageManager is a nullptr
         *              2. serviceDirectoryManager is a nullptr
         *              3. consumerId is an empty string
         */
        static SharedPtr createSoaConsumer(
                const SoaMessageManager::SharedPtr& messageManager,
                const SoaServiceDirectoryManager::SharedPtr&
serviceDirectoryManager,
                const std::string& consumerId);
```

2.6 Shared Services

2.7 SERVICE_MANAGER Service request

FCI Test App requests access to SERVICE_MANAGER service.

Component developers should do that if they create an instance of SoaServiceDirectoryManager to pass it to SoaConsumer::createSoaConsumer(...) facrtory method.

However latest SOA code allows to pass a nullptr instead of a SoaServiceDirectoryManager instance if a component do not need to access SERVICE_MANAGER service.

Looks like now only FCI uses SERVICE_MANAGER service to discover FTCP command handlers.

If you do not create a SoaServiceDirectoryManager instance in your componenet code you do not need to request access to SERVICE_MANAGER service.

3 Descriptor Example

3.1 Service Manager Descriptor

Example of platform component providing services.

Provides two RPC services:

1. SERVICE_MANAGER_LIFECYCLE
2. SERVICE_MANAGER

```
<?xml version="1.0" encoding="utf-8"?>
<platform-client-descriptor  uid="61" version="1.0.0">
  <services>
    <service id="SERVICE_MANAGER_LIFECYCLE">
      <request-endpoint>SERVICES/REQUEST/FNV/SOA/SERVICEMANAGER/SERVICELIFECYCLE</request-endpoint>
    </service>
    <service id="SERVICE_MANAGER">
      <request-endpoint>SERVICES/REQUEST/FNV/SOA/SERVICEMANAGER</request-endpoint>
    </service>
  </services>
</platform-client-descriptor>
```

3.2 FCI Descriptor

Another one example of platform component providing services.and registering consumers.

```
<?xml version="1.0" encoding="utf-8"?>
<platform-client-descriptor uid="190" version="1.0.0">
  <services>
    <service id="FNV_ECG_FCI_GENSERVICE" group-id="FNV_ECG_FCI">
      <request-endpoint>SERVICES/REQUEST/FNV/FCI/GENSERVICE</request-endpoint>
      <data-endpoints>
        <data-endpoint>SERVICES/DATA/FNV/FCI/BROADCAST</data-endpoint>
        <data-endpoint>SERVICES/DATA/FNV/FCI/SMS_BROADCAST</data-endpoint>
      </data-endpoints>
    </service>
    <service id="FNV_ECG_FCI_FTCP_RESPONSE" group-id="FNV_ECG_FCI">
      <request-endpoint>SERVICES/REQUEST/FNV/FCI/FTCP/RESPONSE</request-endpoint>
    </service>
    <service id="FNV_ECG_FCI_FTCP_ALERT" group-id="FNV_ECG_FCI">
      <request-endpoint>SERVICES/REQUEST/FNV/FCI/FTCP/ALERT</request-endpoint>
    </service>
    <service id="FNV_ECG_FCI_FTCP_QUERY" group-id="FNV_ECG_FCI">
      <request-endpoint>SERVICES/REQUEST/FNV/FCI/FTCP/QUERY</request-endpoint>
    </service>
  </services>
  <soa-permissions>
    <soa-consumer-permissions>
      <endpoint>SERVICES/RESPONSE/FNV/FCI/CONSUMER</endpoint>
      <soa-service-requests>
        <soa-service-request service-id="SERVICE_MANAGER" access="RPC"/>
      </soa-service-requests>
    </soa-consumer-permissions>
    <soa-consumer-permissions>
      <endpoint>SERVICES/RESPONSE/FNV/FCI/SERVICEDIR</endpoint>
      <soa-service-requests>
        <soa-service-request service-id="SERVICE_MANAGER" access="RPC"/>
      </soa-service-requests>
    </soa-consumer-permissions>
  </soa-permissions>
</platform-client-descriptor>
```

FCI provides four services:

1. FNV_ECG_FCI_GENSERVICE, listening for requests on "SERVICES/REQUEST/FNV/FCI/GENSERVICE" topic and broadcasting data to "SERVICES/DATA/FNV/FCI/BROADCAST" and "SERVICES/DATA/FNV/FCI/SMS_BROADCAST" topics.
2. FNV_ECG_FCI_FTCP_RESPONSE
3. FNV_ECG_FCI_FTCP_ALERT
4. FNV_ECG_FCI_FTCP_QUERY

FCI has a consumer requesting assess to Service Manager "SERVICE_MANAGER" service declared in Service Manager Descriptor. Similar consumer should be included in the most of the component descriptors. The topic is a topic used to create a SoaServiceDirectoryManager instance.

```
SoaClientEndpoint::SharedPtr sdmEndpoint = SoaMessageManager::createClientEndpoint(
"SERVICES/RESPONSE/FNV/FCI/SERVICEDIR" );

SoaServiceDirectoryManager::SharedPtr svcDirMan =
SoaServiceDirectoryManager::create( messageManager, sdmEndpoint );
```

3.3 FCI Test App Descriptor

Example of an application or a platform component descriptor providing FTCP command handler.

Sample SOA Clent Descriptor

```
<?xml version="1.0" encoding="utf-8"?>

<platform-client-descriptor  uid="777" version="1.0.0">

  <fci-clients>

    <fci-client id="CONTROLMYCAR">

      <ftcp-commands>

        <command id="vehiclestatusupdate"/>

        <command id="initiateremotestart"/>

        <command id="cancelremotestart"/>

        <command id="lock"/>

        <command id="unlock"/>

               /* not listing all the commands */

        <command id="blemprovreq"/>

        <command id="subscriptionstatenotification"/>

      </ftcp-commands>

    </fci-client>

  </fci-clients>

  <soa-permissions>

    <soa-consumer-permissions>

      <endpoint>SERVICES/RESPONSE/FNV/FCITESTAPP/CONSUMER</endpoint>

      <soa-service-requests>

               <soa-service-group-request group-id="FNV_ECG_FCI"/>

      </soa-service-requests>

    </soa-consumer-permissions>
    <soa-consumer-permissions>
      <endpoint>SERVICES/RESPONSE/FNV/FCITESTAPP/SERVICEDIR</endpoint>
      <soa-service-requests>
        <soa-service-request service-id="SERVICE_MANAGER" access="RPC"/>
      </soa-service-requests>
    </soa-consumer-permissions>
  </soa-permissions>
</platform-client-descriptor>
```

The descriptor contains:

1. A "CONTROLMYCAR" FTCP command handler.
2. An FCI service consumer: "SERVICES/RESPONSE/FNV/FCITESTAPP/CONSUMER"
3. A service manager consumer: "SERVICES/RESPONSE/FNV/FCITESTAPP/SERVICEDIR"

NOTE 1. fci client id is an id currently used to register ftcp commands with FCI API.

4 Deploying a descriptor

4.1 ECG

1. Create a descriptor file and add it to "fnv/soa-descriptor" repo
    1. For consistency name the descriptor file *<component-name>-descr.xml*
2. You can run a unit test on QNX VM to verify the descriptor integrity.
3. Update the make file to add new descriptor to SRCS

   MakeFile

   ```
   ifeq ($(FORD_ECU),ecg)
     # shoudl use a dummy directory to avoid deploying all files from current
     # EXPORTED_DESCR_DIR is defauilted to current dir
     EXPORTED_DESCR_DIR = NOT_EXISTING_DERICTORY
     # ecg descriptors should be added to SRCS explicetly
     SRCS += $(CURDIR)/ecgswuagent-descr.xml
        # ... other existing descriptors
     # add your descriptor
     SRCS += $(CURDIR)/your-component-descr.xml


     include $(FORD_COMMON_BUILD)/publish_soa_descriptors.mk
   endif
   ```

   The make file will publish all the added descriptors to "system/etc/soa-sm/descriptors"output directory.

4. On the ECG, whitelist the descriptor in system_build.xml using service manager uid(61) and gid(61). For other ECU's ensure that the file is included in the ECU image.

   ```
   <add_file src="etc/soa-sm/descriptors/soa-sm-descr.xml" tgt="etc/soa-
   sm/descriptors/soa-sm-descr.xml" perms="440" uid="61" gid="61"/>
   ```

5. Add Sergeev, Alexey (A.) and Liang, Su (S.) as a reviews for pull request.
6. Required: run Hardware Test and copy/paste an fdp log message containing "SM client initialized" text.
7. For ECU's running the SOA Gateway, the Gateway has to be configured with the directory that contains the descriptor files.

4.1.1 Hardware test, Required for code reviews. This way is not challenging due to missed logs. Please use the alternative one.

1. You can run a unit test on your QNX VM to ensure the descriptor has not errors and duplicates.

2. Ensure you have the latest repo version.
3. Make ECG build with your changes.
4. Upload the build to ECG target.
5. Power on ECG.
6. Ensure your descriptor file is deployed to "etc/soa-sm/descriptors/"
7. Ensure Service Manger started without errors: in fdp log search for "ClientDataManager: registerEcgDescriptors: ECG database update success" message, application: "serviceManager". "SM initialization error report:" indicates a descriptor related error. Please check the error list

4.1.2 Alternative Hardware test. Does not require creating a build and re-flashing ECG. Limited. Only tests descriptors from FNV/soa-descriptor.

1. Get the latest ECG bundle from jenkins
2. Copy your descriptor to "etc/soa-sm/descriptors/" on target
3. Use "client db_recovery" EDT command. Service manager will remove existing database and create a new one.
4. Ensure Service Manger executed DB recovery without errors: in fdp log search for "ClientDataManager: registerEcgDescriptors: ECG database update success" message, application: "serviceManager".

4.1.3 Unit test

You can locally test your new descriptor by executing "smunittest --gtest_filter=SmStartupTest.loadDescriptorsTest". You can find the test binary in "\system\bin".

The test does not fail. However in case of any descriptor related error the test logs a report. You can either find a "ClientDataManager: registerEcgDescriptors: ECG database update success" message at the end of the log or "SM initialization error report:"

The test verifies a descriptor can be successfully loaded and contains valid metadata.

Momentics launch configuration: sm-local-utest.launch. You will need to update the paths.

The test runs Service Manager client to read descriptors from "/tmp/descriptors/". Update sm-local-utest.launch to copy all files from "system/etc/soa-sm/descriptors/" to "/tmp/descriptors/" on QNX VM target.

It is important to test descriptors all together to ensure a new one does not contain duplicate metadata: topics, service IDs, ftcp commands etc.

Test dependencies

FNV libraries ("system/lib"):

libservicemngr.so, libsoaunittestcommon.so, libsoaframework.so, libsmcommon.so, libsmapi.so, libsmtestcommon.so, libosssmidl.so, libsoaidl.so, libservicemanageridl.so, libtelemetry.so, libfnvthread.so, libanalytics.so, libtelemetryidl.so, libipclite.so, libkeyclient.so, libftcpidl.so, libfdtokenclient.so

List of required FNV libraries can be out of date.

External libraries compiled for FNV ("system/lib"):

libprotobuf.so.13, , libpaho-mqtt3cs.so.1

List of required FNV libraries can be out of date.

SDP libraries

libmq.so.1, libsqlite3-noicu.so.1, libxml2.so.2,

Service Manager errors:

5 Well Known Static Topics

To simplify service descriptors Service Manager automatically adds some well known data to FTCP Command handlers and ACL databases. This item contains a list of such well know cases.

5.1 FCI Client

Service Manager automatically creates database entries to allow FCI <-> FCI Client transactions.

5.2 SOA internal

5.2.1 Control Data topic

Service Manager automatically allows any component to publish/subscribe to all subtopics of "SERVICES/DATA/CONTROL/#".

5.2.1.1 Control Data topic usage

1. Service status updates