FF-M 1.0 – Connection-based services

• Clients make multiple calls to access the service.

• Some services only need one-shot operation
  • too many calls in each operation, runtime overhead
  • “rhandle” is unnecessary for such one-shot service
FF-M 1.1 - Stateless RoT Service

• Introduced in Firmware Framework for M 1.1, implemented by TF-M 1.3 now.

• Improve efficiency
  • Single call to the stateless service
  • No connection and disconnection messages are passed
  • “rhandle” is kept for compatibility but not accessible when accessing stateless service
API change

- Clients request the stateless service via “psa_call()” directly
  - Pass in a valid static handle value defined in the “sid.h”
  - “type” must be >= 0
  - Other parameters are the same as in FF-M 1.0

```c
status = psa_call(ROT_SERVICE_STATIC_HANDLE, type,
                   in_vec, in_len, out_vec, out_len);
```

- PROGRAMMER ERROR
  - Calling `psa_connect()`, `psa_close()` or `psa_set_rhandle()` is a PROGRAMMER ERROR.
Manifest attributes change – stateless service

• Firmware framework version of partition must be 1.1

• “connection_based”
  • Must be set if partition FF version is 1.1
  • False for stateless services
  • True for connection-based services

• “stateless_handle”
  • Used as index, must be positive from 1 to static handle maximum.
  • Can also set as “auto”. If not set, default is “auto”. 
Manifest tool change

- **Automation**
  - Duplicated and invalid static handle index check for the defined “stateless_handle”
  - Auto-allocate static handle index when “stateless_handle” attribute is set as “auto” or not set in yaml/json file.
  - Stateless handle value encoding – indicator bit, version, index

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>stateless handle indicator bit</td>
<td>bit 30</td>
</tr>
<tr>
<td>stateless client version</td>
<td>bit 15 – bit 8</td>
</tr>
<tr>
<td>stateless handle index</td>
<td>bit 7 – bit 0</td>
</tr>
</tbody>
</table>

Client handle encoded to RANGE[CLIENT_HANDLE_VALUE_MIN, 0x3FFFFFFF], no overlap with static handle.
Data structure change

• “connection_based” member added in service static data
  • False for stateless services
  • True for connection-based services

• stateless service tracking table added
  • handle index is converted to the table index (minus one).
  • sid is filled by manifest tool, *p_service is initialized while booting up

<table>
<thead>
<tr>
<th>handle index:</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[handle maximum]</th>
</tr>
</thead>
<tbody>
<tr>
<td>table index:</td>
<td>[0]</td>
<td>[1]</td>
<td>[2]</td>
<td>[handle maximum]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>*p_service</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>*p_service</td>
</tr>
<tr>
<td>sid</td>
<td>*p_service</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>sid</td>
<td>*p_service</td>
</tr>
</tbody>
</table>
Example – stateless service

• Create a partition and a stateless service. Add yaml file:

```yaml
{
  "psa_framework_version": 1.1,
  "name": "TFM_SP_FFM11",
  "type": "APPLICATION-ROT",
  "priority": "NORMAL",
  "entry_point": "tfm_ffm11_partition_main",
  "stack_size": "0x200",
  "services": [
    {
      "name": "TFM_FFM11_SERVICE1",
      "sid": "0x0000F120",
      "non_secure_clients": true,
      "connection_based": false,
      "stateless_handle": "auto",
      "version": 1,
      "version_policy": "RELAXED"
    }
  ]
}
```
Example – stateless service

• Tool generates static handle and SID

    #define TFM_FFMI1_SERVICE1_SID (0x0000F120U)
    #define TFM_FFMI1_SERVICE1_VERSION (1U)
    #define TFM_FFMI1_SERVICE1_HANDLE (0x40000101U)

• Create partition and service: print the data received from message

• Put number “0xFFFFABC1D” into the “in_vec” argument, call the example service with its static handle.

    status = psa_call(TFM_FFMI1_SERVICE1_HANDLE,
                       PSA_IPC_CALL, in_vec, 1, NULL, 0);
Example – stateless service

• Service receives the message, and outputs information:

[Example FFM11 partition] Service called! arg=ffffabcd

> Executing 'TFM_IPC_TEST_1001'
  Description: 'Accessing stateless service from secure partition'
[Example FFM11 partition] Service called! arg=ffffabcd
  TEST: TFM_IPC_TEST_1001 - PASSED!

> Executing 'TFM_IPC_TEST_1012'
  Description: 'Accessing stateless service from non-secure client'
[Example FFM11 partition] Service called! arg=ffffabcd
  TEST: TFM_IPC_TEST_1012 - PASSED!
Apply stateless service

• **Recommended:**
  • Services containing entirely stand-alone functions

• **Not recommended:**
  • API exposes some form of context from the client to be used to manage a connection handle
  • Service manages volatile state for the client – may need “rhandle”
Thank You
Danke
Gracias
谢谢
ありがとうございます
Asante
Merci
감사합니다
dhanyavad
Kiitos
شكرًا
dhanyavad
תודה