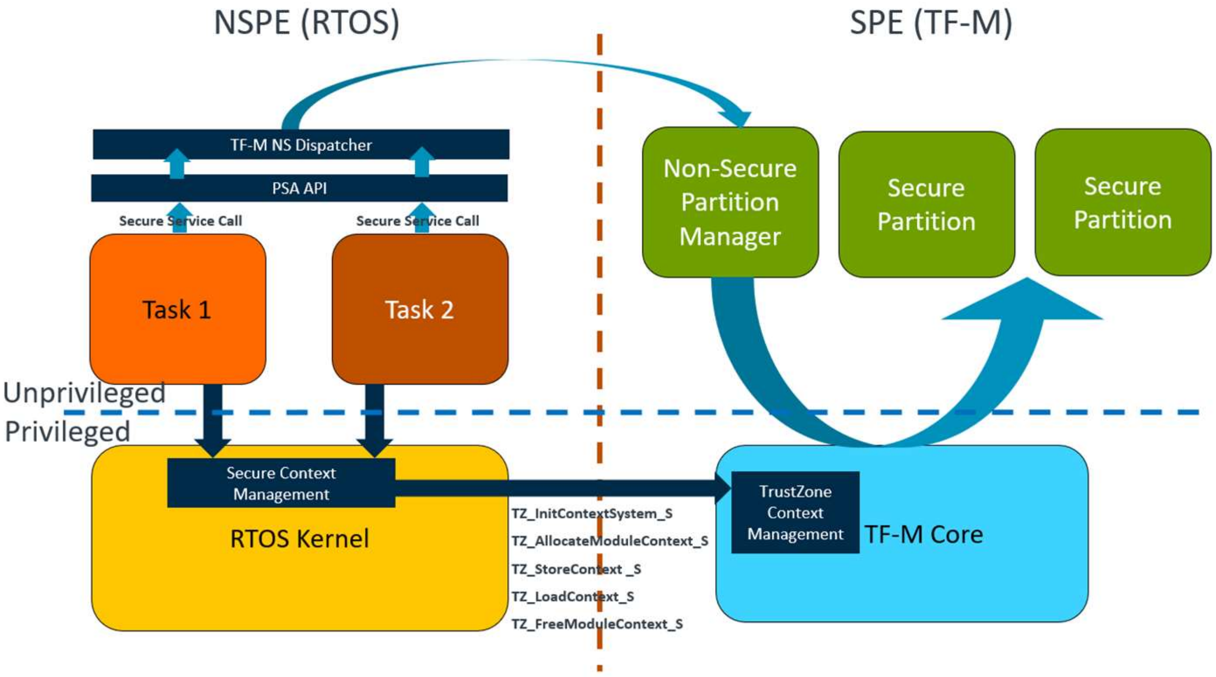
arm

TF-M Non-Secure Client Extension

For TrustZone Based Implementation

David Wang
27 May 2021

Background



- The RTOS Context Management API (a.k.a. TZ API) is not design for NS client ID management.
- NS client IDs are not assigned and partially managed by RTOS kernel
  - Has limitations to ensure the same ID is assigned to the same thread after reboot
- TF-M implemented a simple example of NS client ID generation in library model. It’s disabled by default. Ideally this work should be done by RTOS kernel who has the position to manage NS client context.

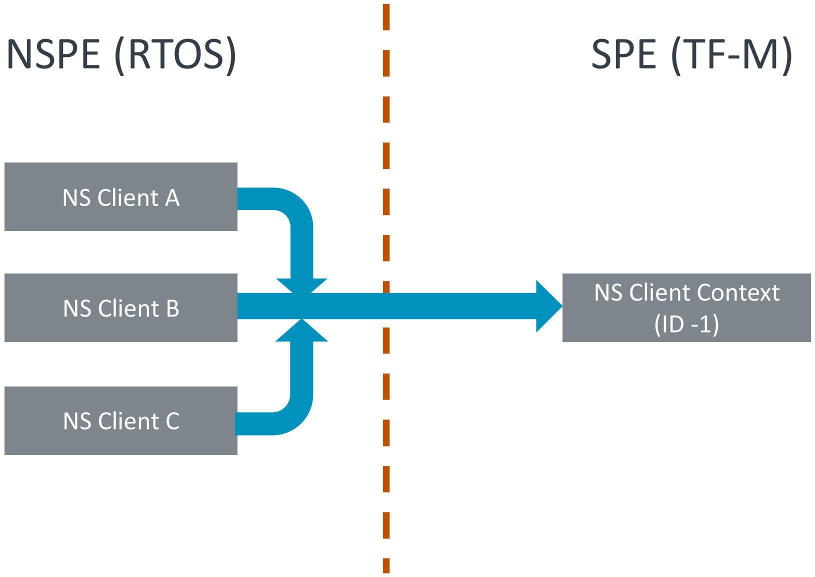
From PSA Firmware Framework Specification

- *“An NSPE client_id is provided by the NSPE OS via the SPM or directly by the SPM”*
- If SPM provides the non-secure client ID
  - *“the same negative client_id must be used for all connections.”*
- If NSPE provides the non-secure client ID

  “
  - *The NSPE operating system must provide the client_id for each connection.*
  - *The SPM must verify that the provided client_id is an NSPE client_id.*

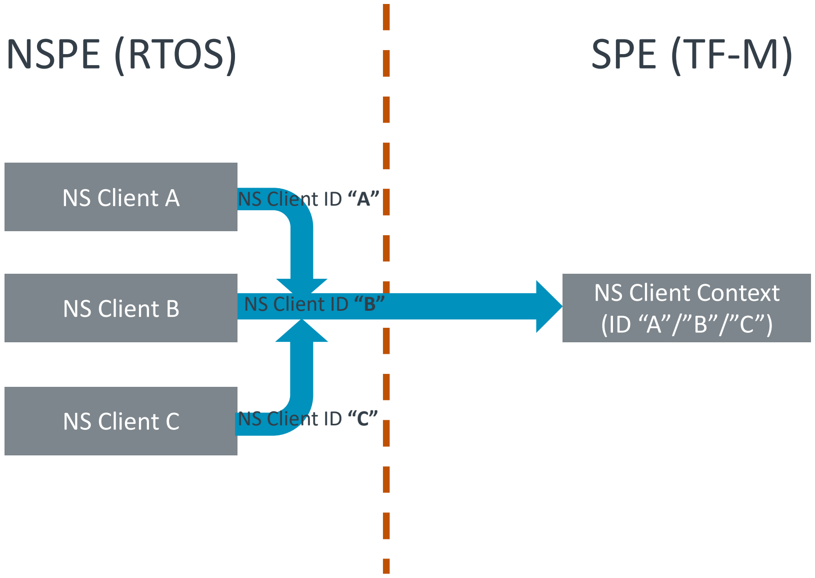
  “

Single NS Client ID - Single Secure Context



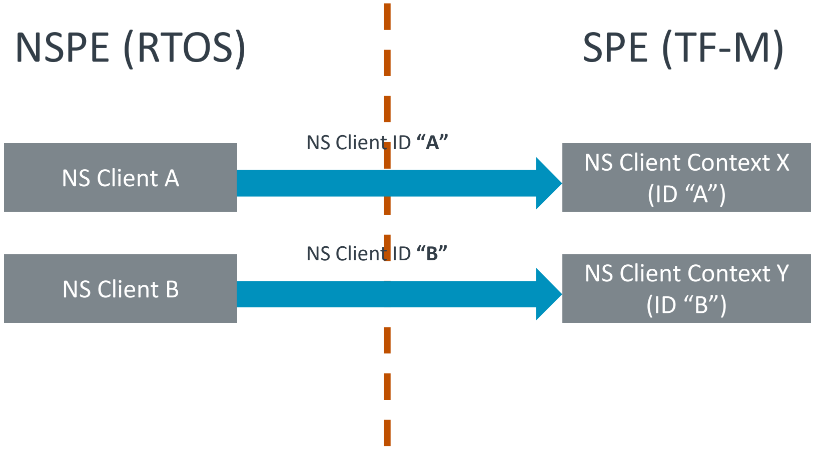
- Current default implementation
- SPM provides the NS client ID (-1)
- A service call from NS client is a blocking operation. i.e. A new service call is allowed only when current call finished – protected by RTOS mutex

Multiple NS Client ID – Single Secure Context



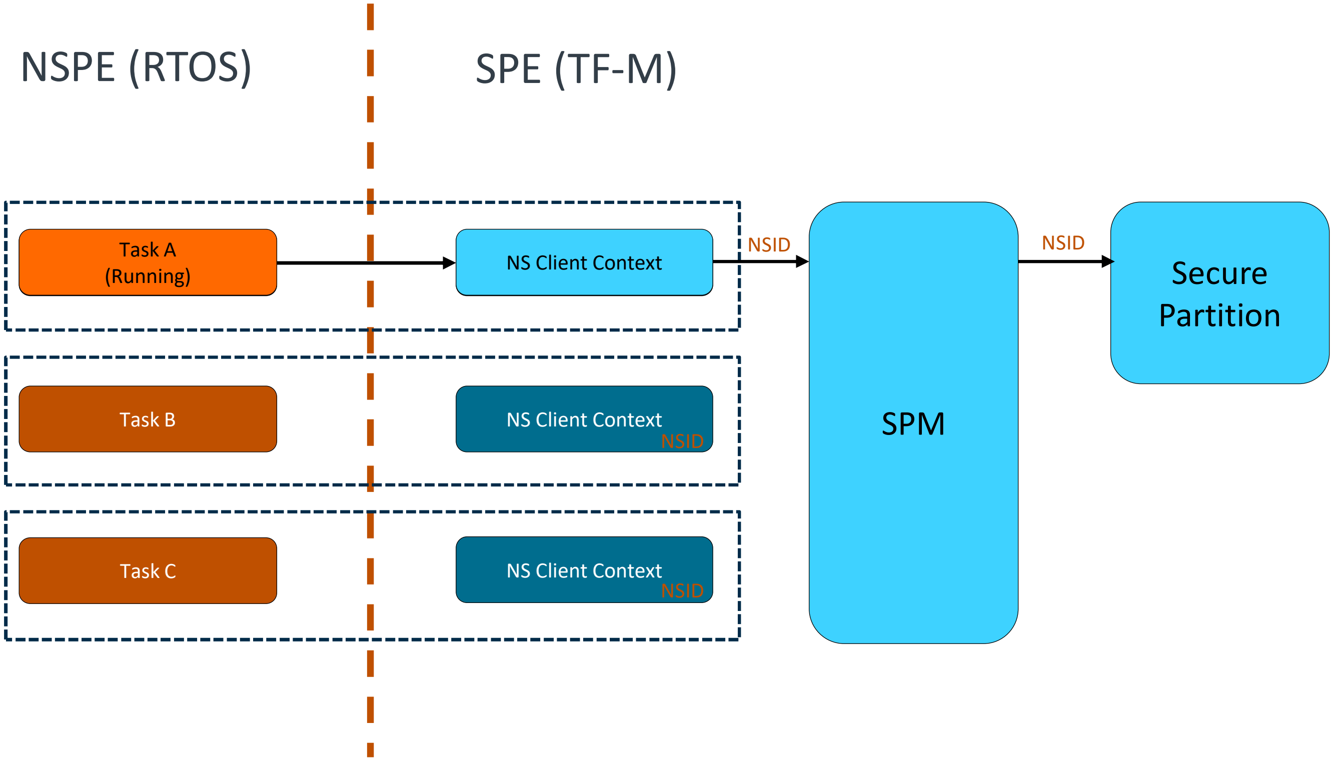
- NSPE provides the NS Client ID
- A service call from NS client is still a blocking operation as there is only one NS client context available.

Multiple NS Client ID - Multiple Secure Context (Future Plan)

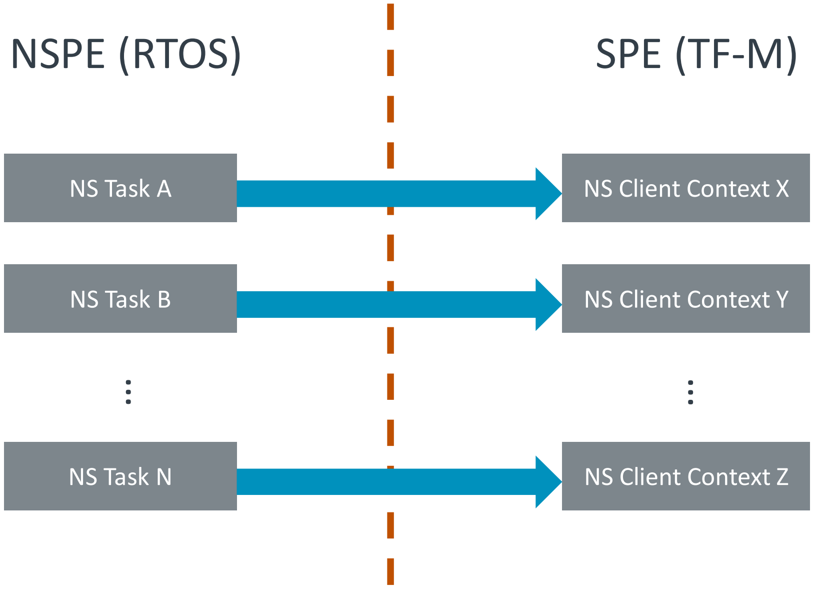


- NSPE provides the NS Client ID which will be stored in NS Client Context
- Need Multiple Secure Context support

Non-secure Client Context Mapping

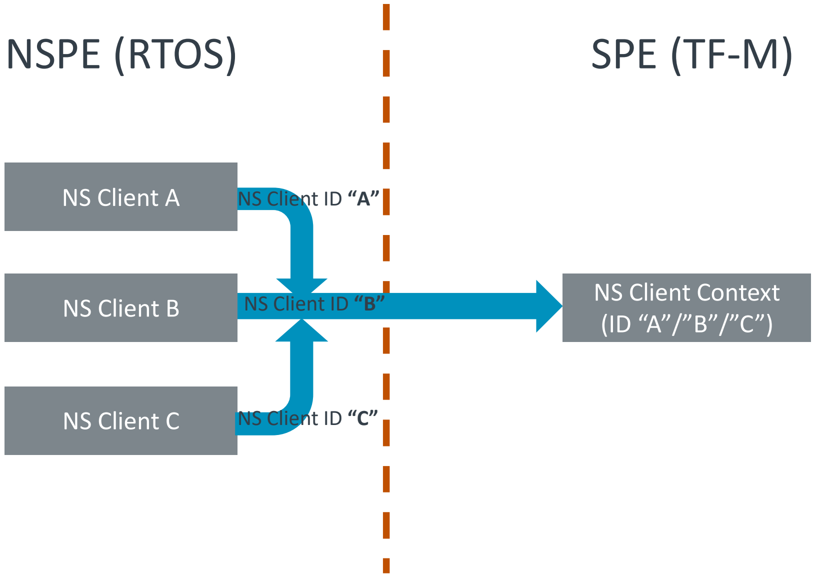


Problem of N - N Mapping



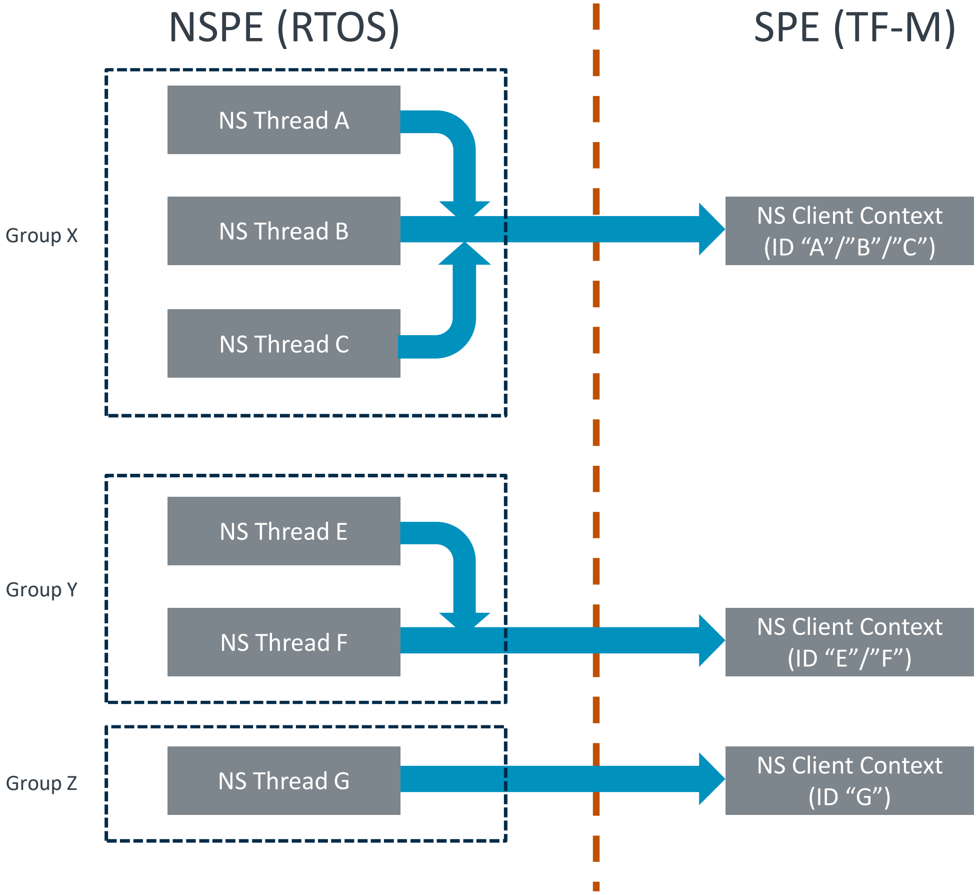
- The mapping of NS client and secure side context is usually static
- Typically, NS client (task) won’t release the assigned context in its lifecycle
- If the NS task doesn’t use secure service frequently, then the efficiency of secure context might be low

How To Support “Single Context – Multi NSID” Scenario?



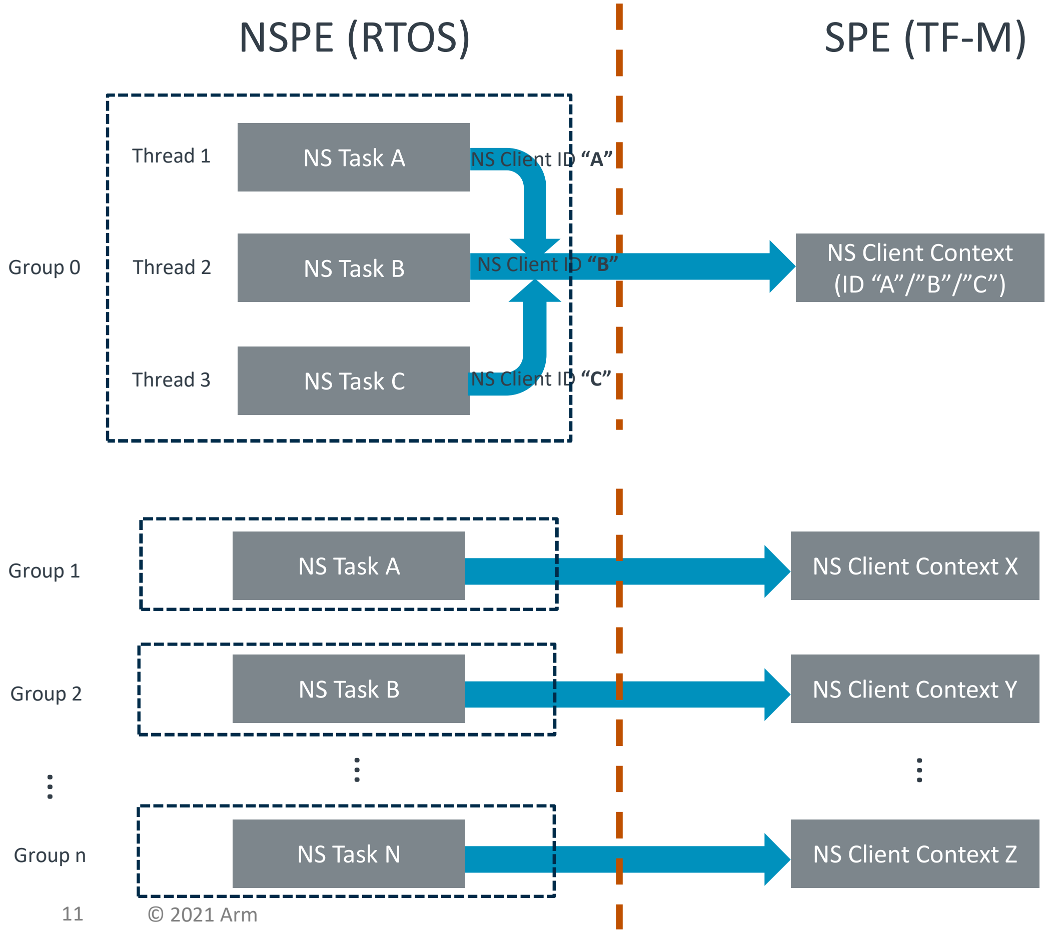
- Statically allocate different context for each NS client
  - Memory footprint ↑
  - Predefine supported numbers of NSID

Group Based NS Client Context



- Threads in the same group share one NS client context
- A thread waits for service call from other threads in the same group finished before calling a secure service – (e.g., share the existing OS mutex mechanism)
- A group takes the context until all threads in the group "release" the context.

How Could This Work for Use Scenarios



NSPE (RTOS)

SPE (TF-M)

Group 0

Thread 1 NS Task A NS Client ID **“A”**

Thread 2 NS Task B NS Client ID **“B”**

Thread 3 NS Task C NS Client ID **“C”**

NS Client Context (ID “A”/”B”/”C”)

Group 1 NS Task A → NS Client Context X

Group 2 NS Task B → NS Client Context Y

⋮

Group n NS Task N → NS Client Context Z

- “Single Context Multi NSID”, N – 1 Mapping
- Use same group ID

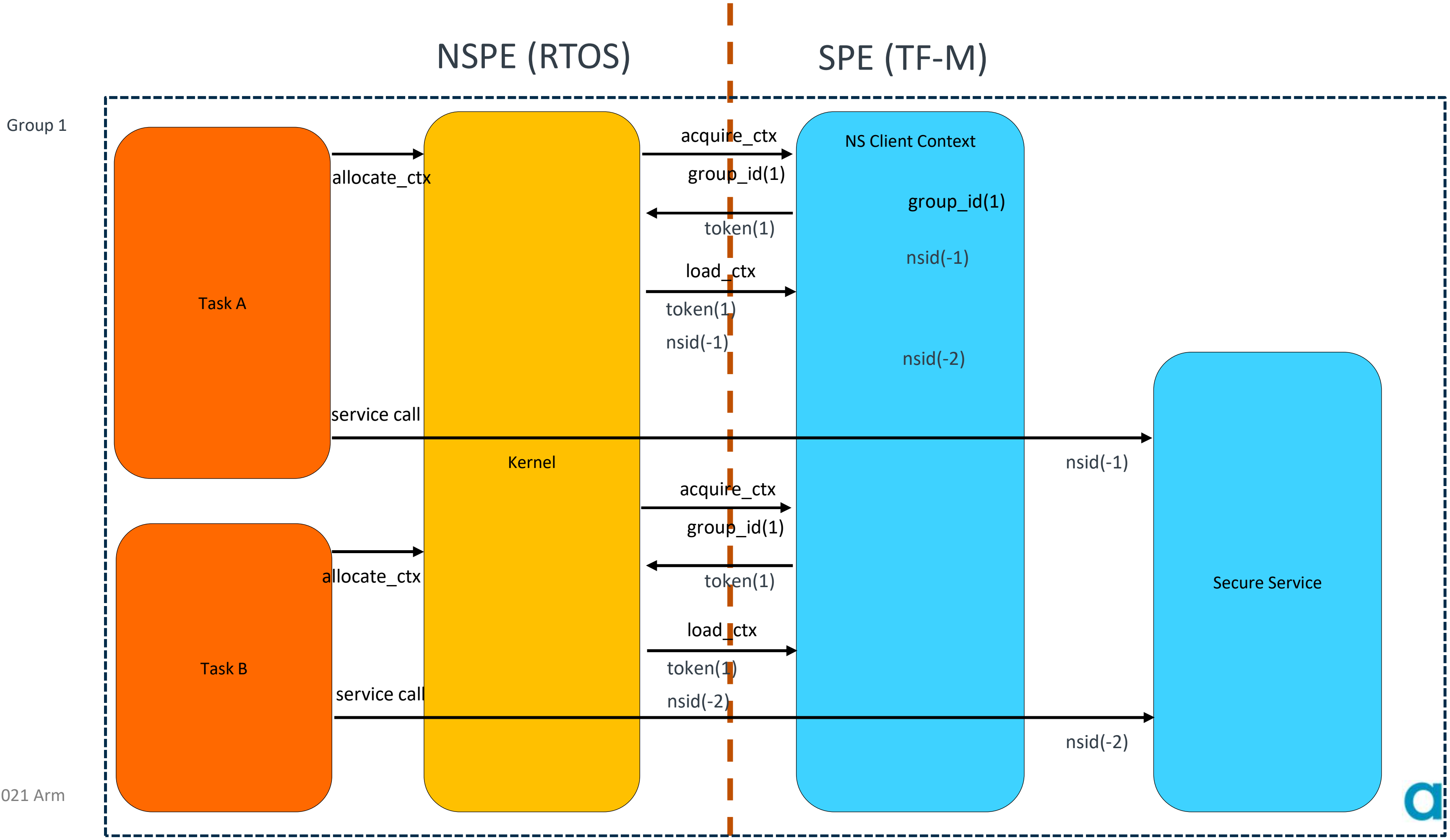
- N - N mapping
- Use different group IDs for each task

- M - N mapping is possible too

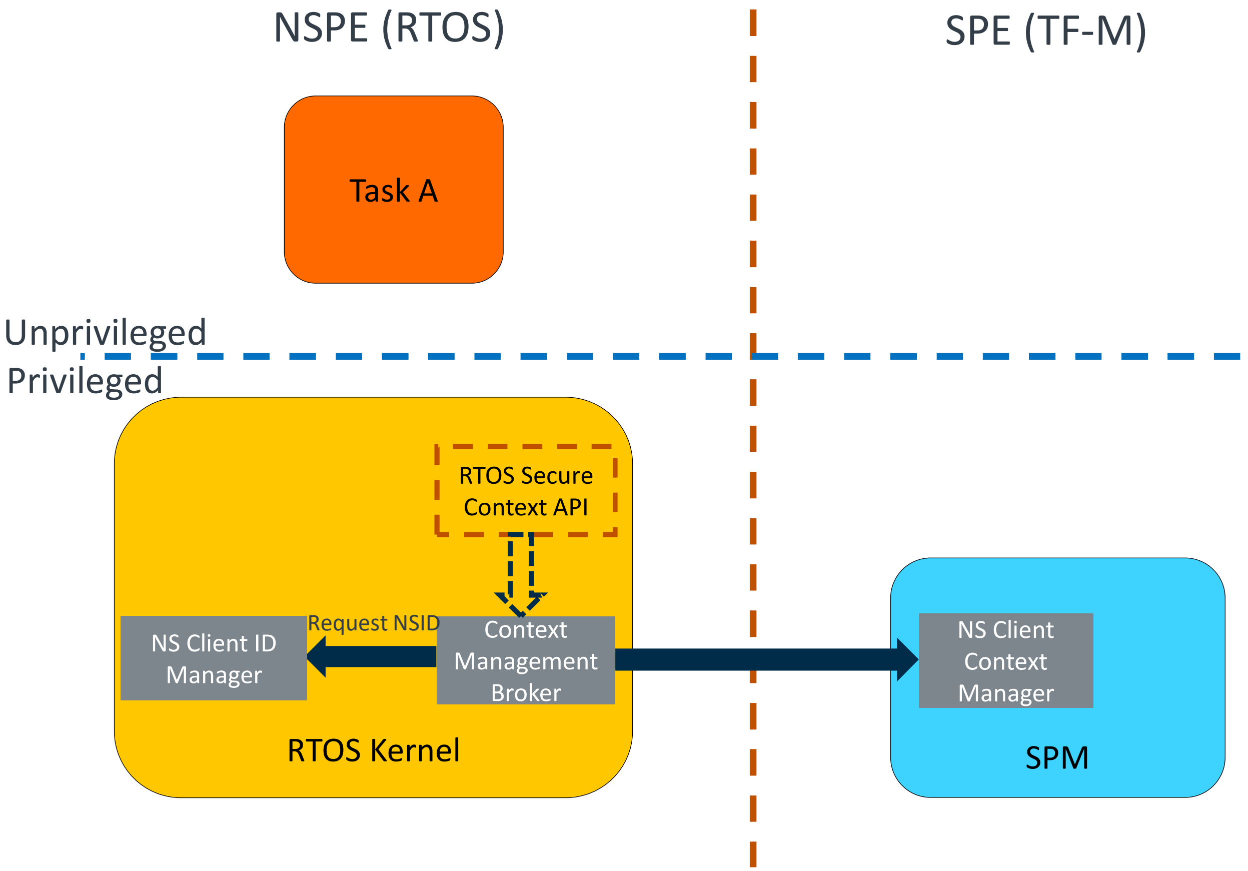
TF-M NS Client Extension API Prototype - Draft

- uint32_t ns_client_acquire_ctx(uint16_t groud_id);
- uint32_t ns_client_release_ctx(uint32_t token);
- uint32_t ns_client_load_ctx(uint32_t token, int32_t nsid);
- uint32_t ns_client_save_ctx(uint32_t token);

Example For N – 1 Mapping



Typical NS Client Extension Modules



- NS Client Context Manager
  - In TF-M
  - The implementation of NS Client Extension API
- NS Client ID Manager (provide example)
  - In RTOS Kernel
  - Manage the NSID assignment and the mapping to NS threads
- Context Management Broker (provide example)
  - In RTOS Kernel
  - Interface to RTOS Secure Context API (if exists)
  - Call NS Client Extension API when NS task scheduling for context management

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה