rpc_library

目次 *[Table of contents]*

図表目次 *[Table of figures]*

rpc_library *[rpc_library]*

機能概要 *[Functional overview]*

表.ユニット概要]

ユニット名*[Unit Name]*	コンポーネント名*[Component Name]*	ユニット概要*[Description]*	オーナーディレクトリ*[Owner Dir]*
rpc_library	OtherService	プログラムから別のアドレス空間にあるサブルーチンや手続きを実行することを可能にする機能を提供する。 *rpc_library provides to help the program to run the subroutine or process in other address space.*	rpc

プログラムから別のアドレス空間にあるサブルーチンや手続きを実行することを可能にする機能を提供する。

librpcはRemote Procedure Callの機能を提供する。

RPC(Remote Procedure Call)ライブラリは、Linuxユーザプロセス空間に存在するプログラム(アプリケーション、ミドルウェア、デバイスドライバなど)間のAPIコール機能を提供するものである。

APIを提供する側をサーバ、APIを利用する側をクライアントと呼び、両プログラムが互いのイベント待ちになる”デッドロック”状態を起こさないようにするため、2プログラム間のサーバ・クライアントの関係は単方向固定とし、動作中に変えられないこととする。

あるプログラムAが、プログラムBに対してはサーバとなり、プログラムCに対してクライアントとなることは可能である。

APIコールは、すべて同期的である。即ち、サーバ側での処理が完了し応答が返るまで、クライアント側には制御が戻らない。

サーバ側で返したエラーコード(関数の戻り値)は、そのままクライアントに返る。

librpcの通信は、INETドメインのデータグラム(UDP)を利用する。

クライアントはサーバが提供するAPIをコールすることで、サーバの機能を利用することができる。クライアントは、複数のサーバのAPIを利用することができる。

サーバはクライアントからのAPIコールを受け付け、要求された処理を行いクライアントに結果を通知する。サーバは、複数のクライアントからのAPIコールを受け付け、順次処理することができる。

rpc_library provides to help the program to run the subroutine or process in other address space.

librpc provides Remote Procedure Call function.

RPC(Remote Procedure Call) library provides API call function among the programs(application, middle ware, device driver etc) in Linux user process space

Server provides API and Client uses API. To avoid the deadlock state in which both programs wait for each other's event, the relation between Server and Client is the single direction and is not be changed during operation.

Program A can be Server for Program B, and can be Client for Program C at the same time.

All API calls are synchronous. That is, API doesn't return the response until Server side processing will be finished.

Error code(function return value) occurred on Server side is returned to Client side.

Communication by librpc uses UDP(User Datagram Protocol).

Client can use Server side function by calling API which Server side provides. Client can use APIs of several Servers.

Server receives API calls from Client and runs them, then returns the results to Client. Server can receives API calls from several Clients and run them sequentially.

全体構成を以下に示す。

Overall structure is as follows.

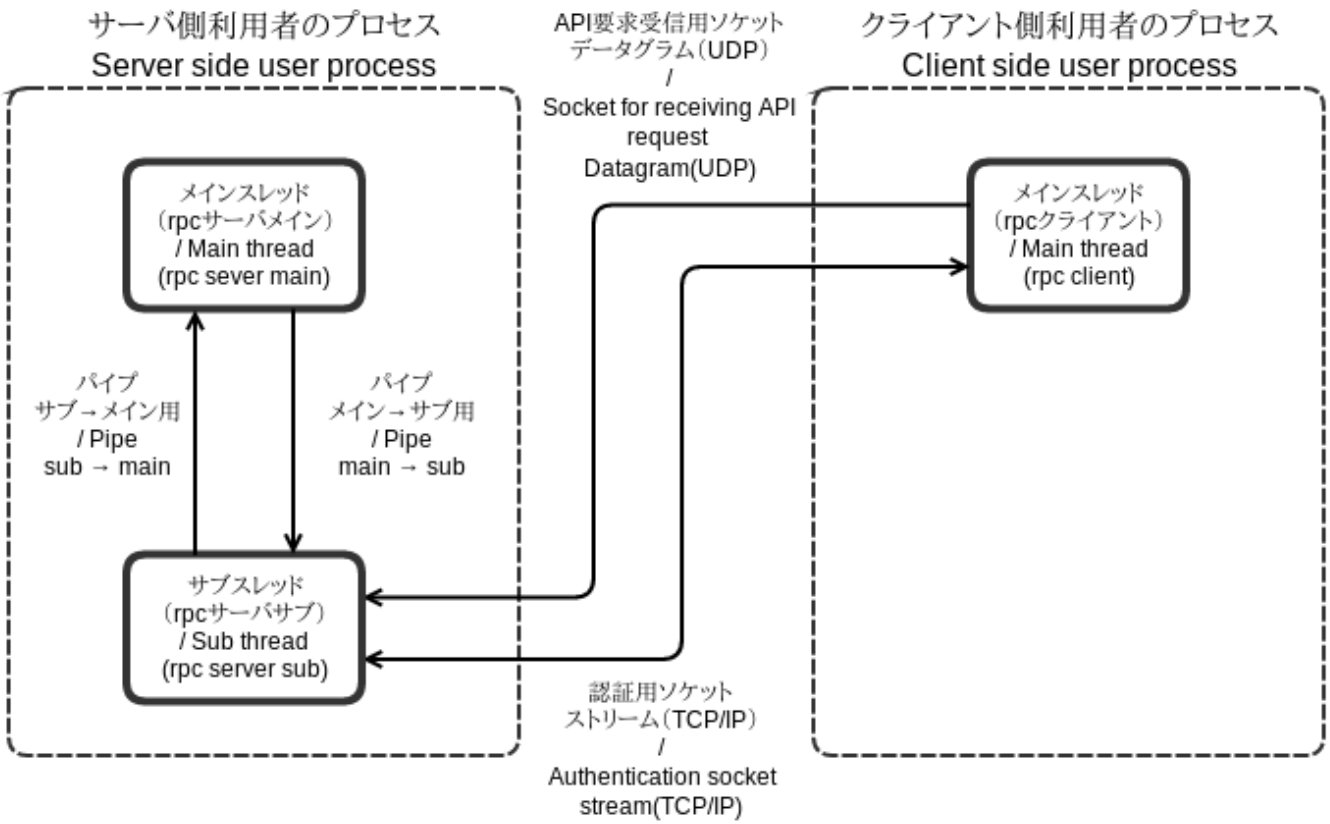


図.全体構成図 [Entire configuration diagram]

機能詳細 *[Function detail]*

RPCライブラリ使用開始・終了

RPC library start and end.

サーバ・クライアントとも、RPCライブラリの機能を使用可能とするために、使用開始処理を行う必要がある。

サーバは、RPC_IDを指定して開始処理を行う。RPC_IDはあらかじめ各サーバ・クライアントに固定的に割り当てられたものを指定する。

librpc通信が不要になった時には、RPC_ID毎に使用終了処理を行う。これにより、librpc通信用に確保したメモリやソケットを解放する。以降、そのRPC_IDを用いたRPC通信の機能は利用不可能とする。

Both Server and Client have to perform start processing to use RPC library function.

Server specifies RPC_ID and performs start processing. RPC_ID which is assigned in advance for Server and Client must be specified.

When the communication for librpc is not necessary, end processing will be done for each RPC_ID. Allocated memory and socket for librpc communication will be released.

After end processing, RPC communication function by using that RPC_ID will not be available anymore.

APIコール(クライアント側)。

API Call(Client side).

サーバの提供するAPIをクライアントから呼び出す。関数呼び出し同様、クライアントが指定したバッファにサーバ側の処理結果を格納して返すことが可能である。

サーバ側がイベント待ち状態の場合は、対応するAPI関数が即座に実行される。サーバ側プログラムがイベント待ち状態にない場合、APIコールはキューイングされ、次にイベント待ちになった時点でAPI関数が実行される。

サーバ側のAPI関数実行が終了するまで、クライント側には制御が戻らない。

API provided from Server side is called by Client. Like function call, API can return the Server side processing result to the buffer specified by Client.

When Server waits for the event, API function is executed immediately. When Server side program doesn't wait for the event, API call is cued.

Function details of this guide are as follows.

The control does not return to Client until Server side API processing will be finished.

APIコールリクエストの処理(サーバ側)。

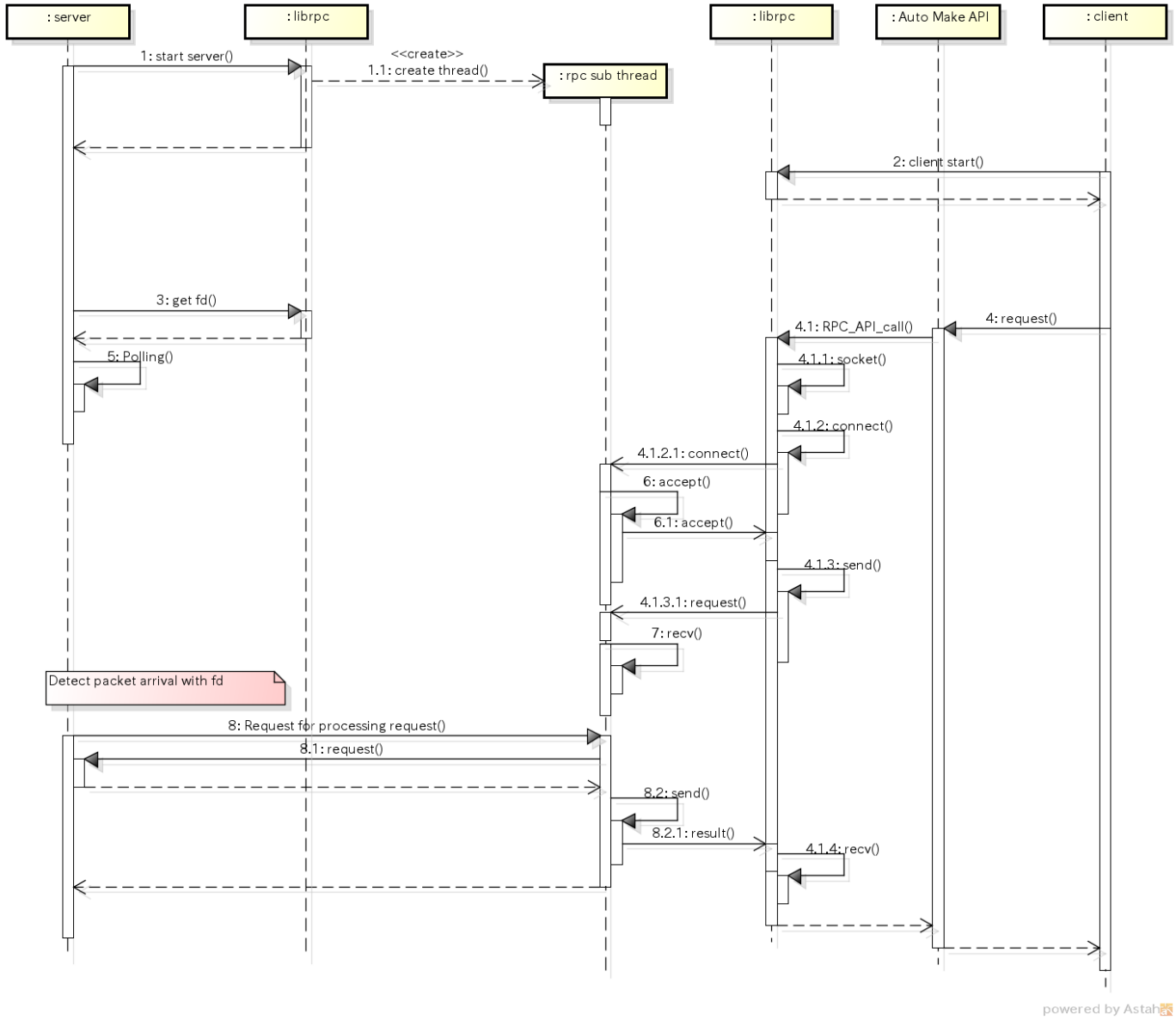
API call request processing(Server side).

クライアントから非同期にリクエストされるAPIコールリクエストを処理するため、サーバはAPIコールリクエストの有無を判定するためのファイルディスクリプタの取得処理、キューイングされたAPIコールリクエストの実行処理を用いて行う。

librpc利用者はサーバとクライアントの役割に分けられる。サーバ、クライアントlibrpcの関係は下記の通りである。

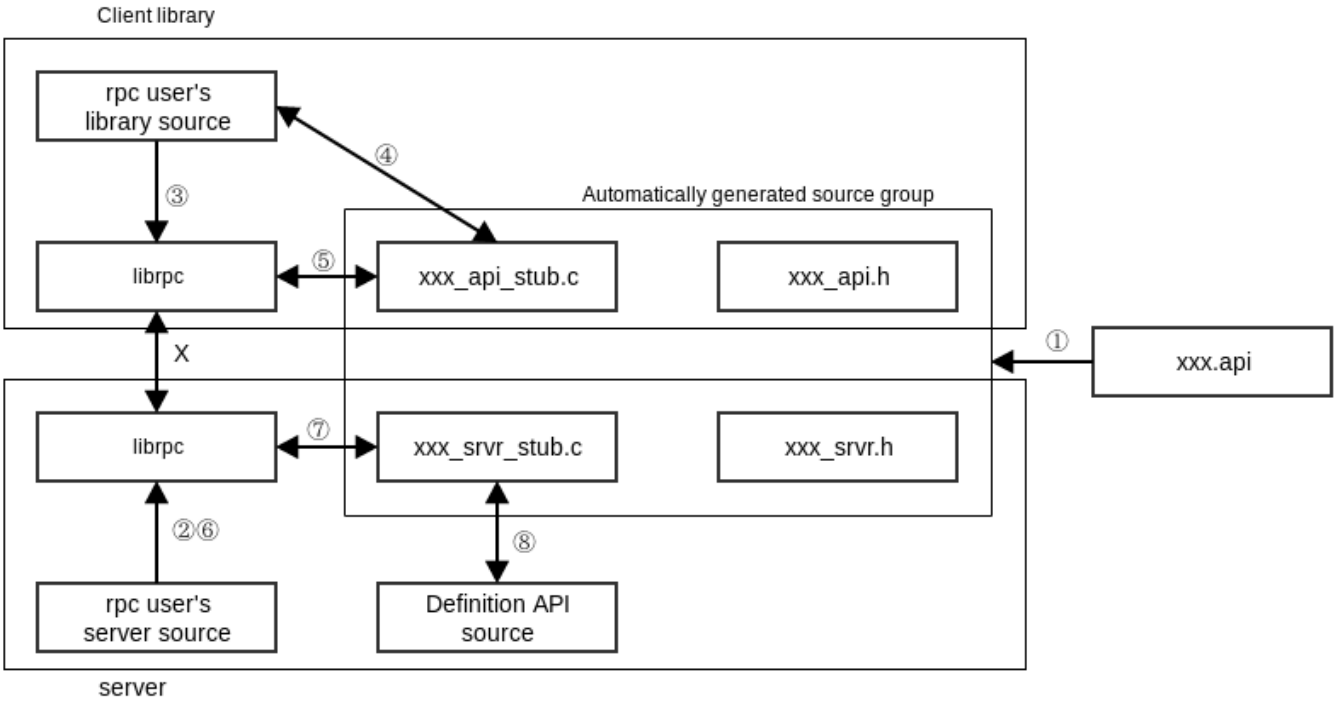
To process API call requested by Client asynchronously, after Server gets the file descriptor to judge whether there is the request of RPC-API call or not, run the API call cued.

The roles of User of librpc are Server and Client. The relation between Server and Client is as follows.



librpcの利用者はlibrpcが提供するソースコード生成ツール(rpc_apidef)を用いて、librpcの利用者のクライアントライブラリを作成する。librpc利用者、librpc、自動生成コードの関係と処理概要は下記の通りである。

User of librpc needs to create client library by using source code generator named rpc_apidef. The relation between User of librpc and librpc and source code generator is as follows.



①rpc_apidefでソースの自動生成を行う

②サーバとしてRPCライブラリを使用開始し、FDを取得し、要求を待ち受ける ．

③クライアントとしてRPCライブラリを使用開始する

④⑤RPC利用者サーバへの処理要求を行う ．

⑥FDで処理要求を検出しRPCライブラリに処理実行要求を行う ．

⑦⑧RPCライブラリ経由で要求に応じた処理を行う ．。

X：適宜TCP/UDPでソケット通信を行う Socket communication by TCP/UDP appropriately.

①Generate source code by rpc_apidef..

②Server starts start processing, gets FD and waits for the request.

③Client starts start processing..

④⑤Request the processing to RPC User's Server.

⑥Detect the processing requested by FD and request the processing to RPC library.

⑦⑧Perform the processing according to the request via RPC library

X：Socket communication by TCP/UDP appropriately.

本Unitで提供する機能の詳細を以下に示す。

Function details of this guide are as follows.

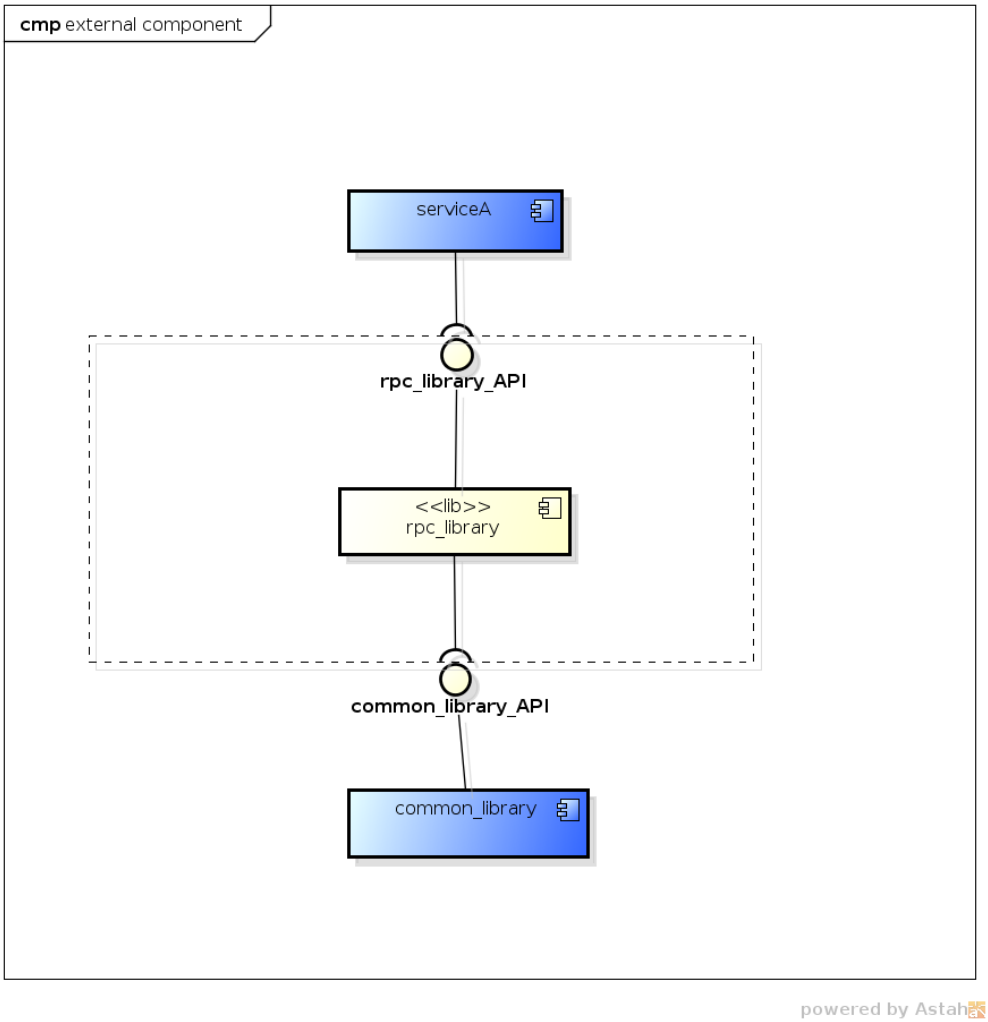
No	機能の種類 *[Type]*	機能 *[Functions]*
1	RPCライブラリ使用開始の処理 *Start RPC library*	サーバ起動(認証あり) */Start secure server* サーバ起動(認証なし) */Start server* クライアント起動 */Start client*
2	タイムアウト時間の設定(サーバ側) *Set the timeout's seconds (run in server)*	タイムアウト時間の設定 */Set the timeout's seconds*

3	認証情報(UID, GID)の登録/取得(サーバ側) *Regist/Get the credential(UID,GID) (run in server)*	認証情報(UID, GID)の登録 */Regist the credential(UID,GID)* 認証情報(UID, GID)の取得 */Get the client'credential(UID,GID)*
4	RPCサーバの動作状態確認(クライアント側) *Get the RPC server's status (run in client)*	RPCサーバの動作状態確認 */Get the RPC server's status*
5	APIコールの実行と処理 *Call/Process the server's API*	マーシャリング */Mashall arguments*
		マーシャリング用メモリの解放 */Free the memory malloced in mashall*
		引数解析 /Demashall arguments
		APIコールの実行 */Call the server's API*
		FD取得 */Get FD*
		APIコール処理 */Process the API call*
		APIコール処理の記録 */Record the API call log*
6	RPCライブラリ使用終了処理 *RPC library end*	RPCライブラリ使用終了処理 */RPC library end* プロセス終了時の強制cleanup */Force to clean up when process is over*

ソフトウェア構成図 *[software block]*

以下にソフトウェア構成を示す。

Software block is as follows.



黄色：ターゲットユニット　青色：外部ユニット

legend of above) Yellow:target units　Blue:outside units

ユースケースとAPI一覧 *[use-case and API lists]*

rpc_libraryのユースケース一覧を記載する。

Describe use-case list of rpc_library.

外部要因 ユースケース一覧*[outside factor use-case list]*

表. 外部要因ユースケース一覧 [outside factor use-case list]

ユースケース番号 *[use-case number]*	機能カテゴリ *[function category]*	ユースケース名 *[use-case name]*	ユースケースを実現するAPI *[API that realizes use-case]*
rpc_library_StartRPC_001	RPCライブラリ使用開始処理 *Start RPC library*	サーバ起動(認証あり) */Start secure server*	RPC_START_SECURE_SERVER
rpc_library_StartRPC_002		サーバ起動(認証なし) */Start server*	RPC_START_SERVER
rpc_library_StartRPC_003		クライアント起動 */Start client*	RPC_START_CLIENT
rpc_library_SetAPITimeout_001	タイムアウト時間の設定(サーバ側) *Set the timeout's time (call by server)*	タイムアウト時間の設定 */Set the timeout's time*	RPC_START_SECURE_SERVER RPC_START_SERVER RPC_set_API_timeout

rpc_library_RegistAndGetCredential_001	認証情報(UID, GID)の登録/取得(サーバ側) *Regist and get the credential(UID, GID) (call by server)*	認証情報(UID, GID)の登録 */Regist the credential(UID,GID)*	RPC_START_SECURE_SERVER RPC_START_SERVER RPC_regist_credential
rpc_library_RegistAndGetCredential_002		認証情報(UID, GID)の取得 */Get the client'credential(UID,GID)*	RPC_START_SECURE_SERVER RPC_START_SERVER RPC_get_client_credential
rpc_library_GetServerStatus_001	RPCサーバの動作状態確認(クライアント側) *Get the RPC server's status (call by client)*	RPCサーバの動作状態確認 */Get the RPC server's status*	RPC_START_CLIENT RPC_is_server_ready
rpc_library_CallAndProcessAPI_001	APIコールの実行と処理 *Call and process the server's API*	マーシャリング */Mashall arguments*	RPC_marshall_arguments
rpc_library_CallAndProcessAPI_002		マーシャリング用メモリの解放 /Free the memory malloced in mashall	RPC_marshall_arguments RPC_marshall_free
rpc_library_CallAndProcessAPI_003		引数解析 */Demashall arguments*	RPC_marshall_arguments RPC_demarshall_arguments
rpc_library_CallAndProcessAPI_004		APIコールの実行 */Call the server's API*	RPC_START_CLIENT RPC_marshall_arguments RPC_API_call RPC_free_return_string
rpc_library_CallAndProcessAPI_005		FD取得 */Get FD*	RPC_START_SECURE_SERVER RPC_START_SERVER RPC_get_fd
rpc_library_CallAndProcessAPI_006		APIコールの処理 */Process the API call*	RPC_START_SECURE_SERVER RPC_START_SERVER RPC_get_fd RPC_process_API_request RPC_demarshall_arguments RPC_marshall_arguments RPC_marshall_free
rpc_library_CallAndProcessAPI_007		APIコール処理の記録 */Record the API call log*	RPC_START_CLIENT RPC_marshall_arguments RPC_API_call RPC_record_dbg_log RPC_free_return_string
rpc_library_EndRPC_001	RPCライブラリ使用終了処理 *RPC library end*	RPCライブラリ使用終了処理 */RPC library end*	RPC_START_SECURE_SERVER RPC_START_SERVER RPC_START_CLIENT RPC_end
rpc_library_EndRPC_002		プロセス終了時の強制cleanup */Force to clean up when process is over*	RPC_START_SECURE_SERVER RPC_START_SERVER RPC_START_CLIENT RPC_end_all

内部処理 エラーユースケース一覧*[internal processing error use-case list]*

表. 内部処理エラーユースケース一覧 [Internal processing error use-case list]

エラー番号 *[error number]*	機能カテゴリ *[function category]*	ユースケース名 *[use-case name]*	戻り値 *[return value]*	備考 *[notes]*
rpc_library_StartRPC_Error_001	RPCライブラリ使用開始処理 *Start RPC library*	RPC_START_SECURE_SERVER Info Create Fail	RPC_ERR_Fatal	RPC_START_SECURE_SERVERで情報生成に失敗した場合
rpc_library_StartRPC_Error_002		RPC_START_SECURE_SERVER RPC Start Already Issue	RPC_ERR_Fatal	RPC_START_SECURE_SERVERで既にRPC_startが発行済みの場合
rpc_library_StartRPC_Error_003		RPC_START_SECURE_SERVER Area Get Fail	RPC_ERR_Fatal	RPC_START_SECURE_SERVERで領域確保に失敗した場合
rpc_library_StartRPC_Error_004		RPC_START_SECURE_SERVER Socket Create Fail	RPC_ERR_Fatal	RPC_START_SECURE_SERVERでソケットの生成に失敗した場合
rpc_library_StartRPC_Error_005		RPC_START_SECURE_SERVER Socket Assign Fail	RPC_ERR_Fatal	RPC_START_SECURE_SERVERでソケットの割り当てに失敗した場合
rpc_library_StartRPC_Error_006		RPC_START_SECURE_SERVER Pipe Create Fail	RPC_ERR_Fatal	RPC_START_SECURE_SERVERでパイプの作成に失敗した場合
rpc_library_StartRPC_Error_007		RPC_START_SERVER Info Create Fail	RPC_ERR_Fatal	RPC_START_SERVERで情報生成に失敗した場合
rpc_library_StartRPC_Error_008		RPC_START_SERVER RPC Start Already Issue	RPC_ERR_Fatal	RPC_START_SERVERで既にRPC_startが発行済みの場合
rpc_library_StartRPC_Error_009		RPC_START_SERVER Area Get Fail	RPC_ERR_Fatal	RPC_START_SERVERで領域確保に失敗した場合
rpc_library_StartRPC_Error_010		RPC_START_SERVER Socket Create Fail	RPC_ERR_Fatal	RPC_START_SERVERでソケットの生成に失敗した場合
rpc_library_StartRPC_Error_011		RPC_START_SERVER Socket Assign Fail	RPC_ERR_Fatal	RPC_START_SERVERでソケットの割り当てに失敗した場合
rpc_library_StartRPC_Error_012		RPC_START_SERVER Create Fail	RPC_ERR_Fatal	RPC_START_SERVERでパイプの作成に失敗した場合
rpc_library_StartRPC_Error_013		RPC_START_CLIENT RPC Library Info Get Error	RPC_ERR_Fatal	RPC_START_CLIENTで自スレッド用rpcライブラリ情報取得に失敗した場合
rpc_library_StartRPC_Error_014		RPC_START_CLIENT ID Info Area Get Error	RPC_ERR_Fatal	RPC_START_CLIENTでID情報の領域確保に失敗した場合
rpc_library_StartRPC_Error_015		RPC_START_CLIENT Socket Create Fail	RPC_ERR_Fatal	RPC_START_CLIENTでソケットの生成に失敗した場合
rpc_library_StartRPC_Error_016		RPC_START_CLIENT Socket Assign Fail	RPC_ERR_Fatal	RPC_START_CLIENTでケットの割り当てに失敗した場合
rpc_library_StartRPC_Error_017		RPC_START_CLIENT Socket Name Get Fail	RPC_ERR_Fatal	RPC_START_CLIENTでソケット名の取得に失敗した場合
RPC_set_API_timeout_Error_001～012	タイムアウト時間の設定(サーバ側) *Set the timeout's time (call by server)*	rpc_library_StartRPC_Error_001～012と同等	同左	同左
RPC_set_API_timeout_Error_013		RPC_set_API_timeout RPC Library Info Get Error	RPC_ERR_Fatal	自スレッド用rpcライブラリ情報取得に失敗した場合
RPC_set_API_timeout_Error_014		RPC_set_API_timeoutl Server ID Info Not Exist	RPC_ERR_Fatal	サーバーid情報が存在しない場合
RPC_set_API_timeout_Error_015		RPC_set_API_timeout API Call Receive Info Not Exist	RPC_ERR_Fatal	APIコール受け付け処理のための情報が存在しない場合
rpc_library_RegistAndGetCredential_Error_001～012	認証情報(UID, GID)の登録/取得(サーバ側) *Regist and get the credential(UID,GID) (call by server)*	rpc_library_StartRPC_Error_001～012と同等	同左	同左
rpc_library_RegistAndGetCredential_Error_013		RPC_regist_credential RPC Library Info Get Error	RPC_ERR_Fatal	自スレッド用rpcライブラリ情報取得に失敗した場合
rpc_library_RegistAndGetCredential_Error_014		RPC_regist_credential Server ID Info Not Exist	RPC_ERR_Fatal	サーバーid情報が存在しない場合

rpc_library_RegistAndGetCredential_Error_015		RPC_regist_credential API Call Receive Info Not Exist	RPC_ERR_Fatal	APIコール受け付け処理のための情報が存在しない場合
rpc_library_RegistAndGetCredential_Error_016		RPC_regist_credentiall Client Check Unnecessary	RPC_ERR_Fatal	クライアント認証チェックが不要な場合
rpc_library_RegistAndGetCredential_Error_017		RPC_regist_credential Certification Info Already Registrated	RPC_ERR_Fatal	認証情報が既に登録されている場合
rpc_library_RegistAndGetCredential_Error_018		RPC_regist_credential Memory Area Get Error	RPC_ERR_Fatal	UID,GIDリストのそれぞれ登録個数分のメモリ領域を確保に失敗した場合
rpc_library_RegistAndGetCredential_Error_019		RPC_get_client_credential Server ID Info Not Exist	RPC_ERR_Fatal	サーバーid情報が存在しない場合
rpc_library_RegistAndGetCredential_Error_020		RPC_get_client_credential API Call Receive Info Not Exist	RPC_ERR_Fatal	APIコール受け付け処理のための情報が存在しない場合
rpc_library_RegistAndGetCredential_Error_021		RPC_get_client_credential Client Check Unnecessary	RPC_ERR_Fatal	クライアント認証チェックが不要な場合
rpc_library_RegistAndGetCredential_Error_022		RPC_get_client_credential Running Client Not Exist	RPC_ERR_Fatal	実行中のクライアントが無い場合
rpc_library_RegistAndGetCredential_Error_023		RPC_get_client_credential Running Client Info Unmatched	RPC_ERR_Fatal	実行中のクライアント情報該当なしの場合
rpc_library_GetServerStatus_Error_001～005	RPCサーバの動作状態確認(クライアント側) *Get the RPC server's status (call by client)*	rpc_library_StartRPC_Error_013～017と同等	同左	同左
rpc_library_CallAndProcessAPI_Error_001～017	APIコールの実行と処理 *Call and process the server's API*	rpc_library_StartRPC_Error_001～017と同等	同左	同左
rpc_library_CallAndProcessAPI_Error_018		RPC_marshall_arguments Work Area Get Fail	NULL	marshallワーク領域確保に失敗した場合
rpc_library_CallAndProcessAPI_Error_019		RPC_marshall_arguments Msg Buffer Get Fail	NULL	変換後のメッセージバッファの領域確保に失敗した場合
rpc_library_CallAndProcessAPI_Error_020		RPC_API_call RPC Library Info Get Error	RPC_ERR_Fatal	自スレッド用rpcライブラリ情報取得に失敗した場合
rpc_library_CallAndProcessAPI_Error_021		RPC_API_call RPC Library Info Create Error	RPC_ERR_Fatal	自スレッド用rpcライブラリ情報生成に失敗した場合
rpc_library_CallAndProcessAPI_Error_022		RPC_API_call RPC ID Info Not Exist	RPC_ERR_Fatal	RPC用ID情報が存在しない場合
rpc_library_CallAndProcessAPI_Error_023		RPC_API_call Initial Cert Data Receive Fail	RPC_ERR_Fatal	初回認証時にサーバからのデータ受信に失敗した場合
rpc_library_CallAndProcessAPI_Error_024		RPC_API_call Initial Cert Received Data Analysis Fail	RPC_ERR_Fatal	初回認証時に受信データの解析に失敗した場合
rpc_library_CallAndProcessAPI_Error_025		RPC_API_call Initial Cert Unexpected Command	RPC_ERR_Fatal	初回認証時に受信内容が期待しないコマンドの場合
rpc_library_CallAndProcessAPI_Error_026		RPC_API_call Initial Cert API Call Error	RPC_ERR_Fatal	初回認証時にサーバ応答がAPIコールエラーの場合
rpc_library_CallAndProcessAPI_Error_027		RPC_API_call Execute Req Data Receive Fail	RPC_ERR_Fatal	API処理要求時にサーバからのデータ受信に失敗した場合
rpc_library_CallAndProcessAPI_Error_028		RPC_API_call Execute Req Received Data Analysis Fail	RPC_ERR_Fatal	API処理要求時に受信データの解析に失敗した場合
rpc_library_CallAndProcessAPI_Error_029		RPC_API_call Execute Req Unexpected Command	RPC_ERR_Fatal	API処理要求時に受信内容が期待しないコマンドの場合
rpc_library_CallAndProcessAPI_Error_030		RPC_API_call Execute Req API Call Error	RPC_ERR_Fatal	API処理要求時にサーバ応答がAPIコールエラーの場合
rpc_library_CallAndProcessAPI_Error_031		RPC_API_call Execute Req Server Response Error	RPC_ERR_Fatal	処理結果受信時にサーバ応答がデッドロック以外のエラーの場合

rpc_library_CallAndProcessAPI_Error_032	RPC_API_call Server Stop	RPC_ERR_Fatal	サーバが終了した場合
rpc_library_CallAndProcessAPI_Error_033	RPC_API_call Initial Cert Send UDP Packet Fail	RPC_ERR_No_Response	初回認証時にUDPパケット送信に失敗した場合
rpc_library_CallAndProcessAPI_Error_034	RPC_API_call Initial Cert Response Time Out	RPC_ERR_No_Response	初回認証時にサーバからの応答でタイムアウトが発生した場合
rpc_library_CallAndProcessAPI_Error_035	RPC_API_call Execute Req Send UDP Packet Fail	RPC_ERR_No_Response	API処理要求時にUDPパケット送信に失敗した場合
rpc_library_CallAndProcessAPI_Error_036	RPC_API_call Execute Req Response Time Out	RPC_ERR_No_Response	API処理要求時にサーバからの応答でタイムアウトが発生した場合
rpc_library_CallAndProcessAPI_Error_037	RPC_API_call Initial Cert API Call Queue Overflow	RPC_ERR_Busy	初回認証時にサーバ応答がAPIコールキュー溢れの場合
rpc_library_CallAndProcessAPI_Error_038	RPC_API_calll Execute Req API Call Queue Overflow	RPC_ERR_Busy	API処理要求時にサーバ応答がAPIコールキュー溢れの場合
rpc_library_CallAndProcessAPI_Error_039	RPC_API_call Initial Cert Server Deadlock	RPC_ERR_Server_DeadLock	初回認証時にサーバ応答がデッドロックの場合
rpc_library_CallAndProcessAPI_Error_040	RPC_API_calll Execute Req Server Deadlock	RPC_ERR_Server_DeadLock	API処理要求時にサーバ応答がデッドロックの場合
rpc_library_CallAndProcessAPI_Error_041	RPC_API_call Receive Result Server Deadlock	RPC_ERR_Server_DeadLock	処理結果受信時にサーバ応答がデッドロックの場合
rpc_library_CallAndProcessAPI_Error_042	RPC_API_call Initial Cert Certification NG	RPC_ERR_Reject_connect	初回認証時にサーバ応答が認証NGの場合
rpc_library_CallAndProcessAPI_Error_043	RPC_get_fd RPC Library Info Get Error	RPC_ERR_Fatal	自スレッド用rpcライブラリ情報取得に失敗した場合
rpc_library_CallAndProcessAPI_Error_044	RPC_get_fd Sub Thread Not Exist	RPC_ERR_Fatal	RPCサブスレッドが存在しない場合
rpc_library_CallAndProcessAPI_Error_045	RPC_process_API_request RPC Library Info Get Error	RPC_ERR_Fatal	自スレッド用rpcライブラリ情報取得に失敗した場合
rpc_library_CallAndProcessAPI_Error_046	RPC_process_API_request Sub Thread Not Exist	RPC_ERR_Fatal	RPCサブスレッドが存在しない場合
rpc_library_CallAndProcessAPI_Error_047	RPC_process_API_request Send UDP Packet Fail	RPC_ERR_Fatal	UDPパケット送信失敗の場合
rpc_library_CallAndProcessAPI_Error_048	RPC_process_API_request Client API Execute Error	RPC_ERR_Fatal	クライアントからのAPI処理がエラーを返す場合

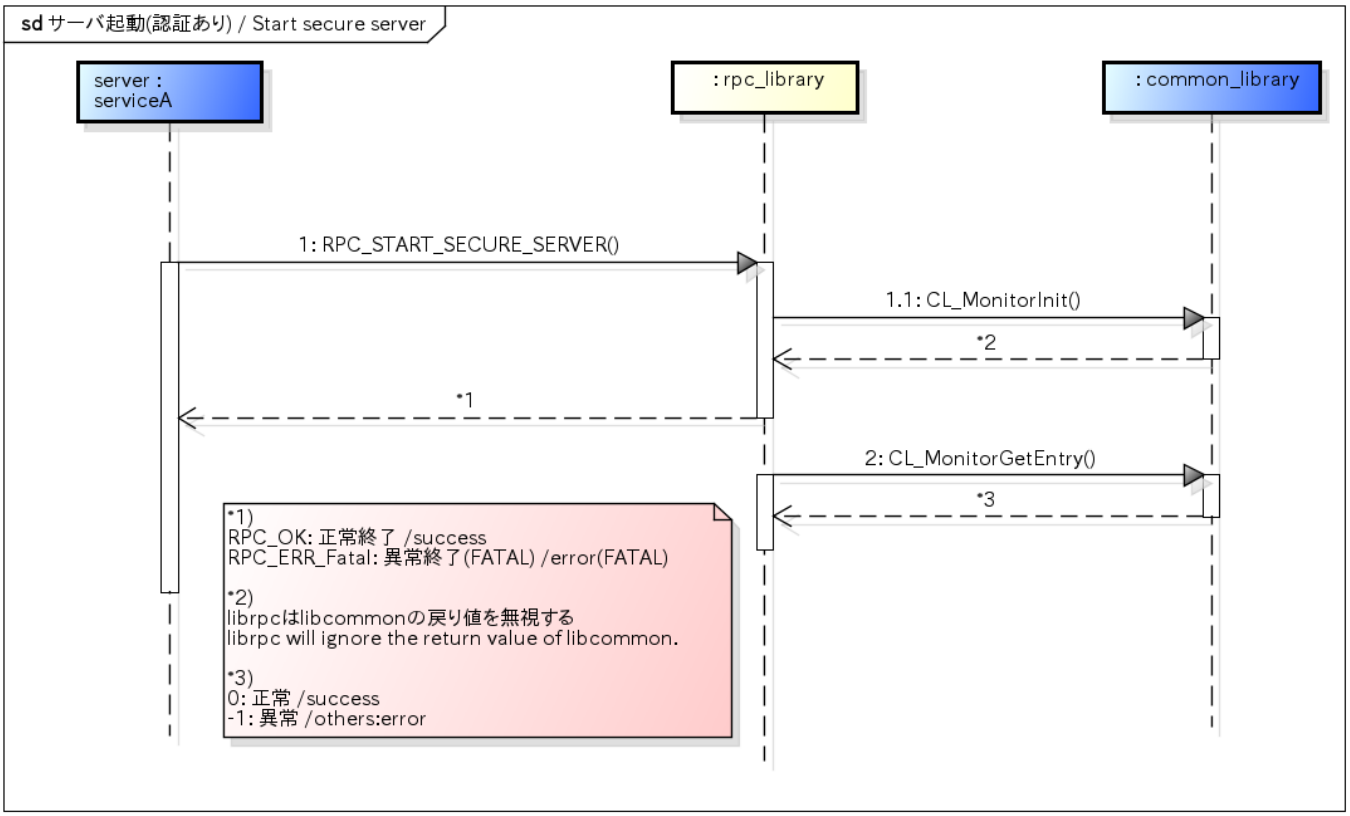
ユースケースrpc_library_StartRPC_001 *[use-case rpc_library_StartRPC_001]*

概要 *[Overview]*

RPC-APIのサーバプログラムがRPCライブラリ使用開始処理を行う場合に使用する。(認証あり)

Calling this function when RPC_API's server program start to use the RPC library.(Need secure check)

シーケンス *[Sequence]*



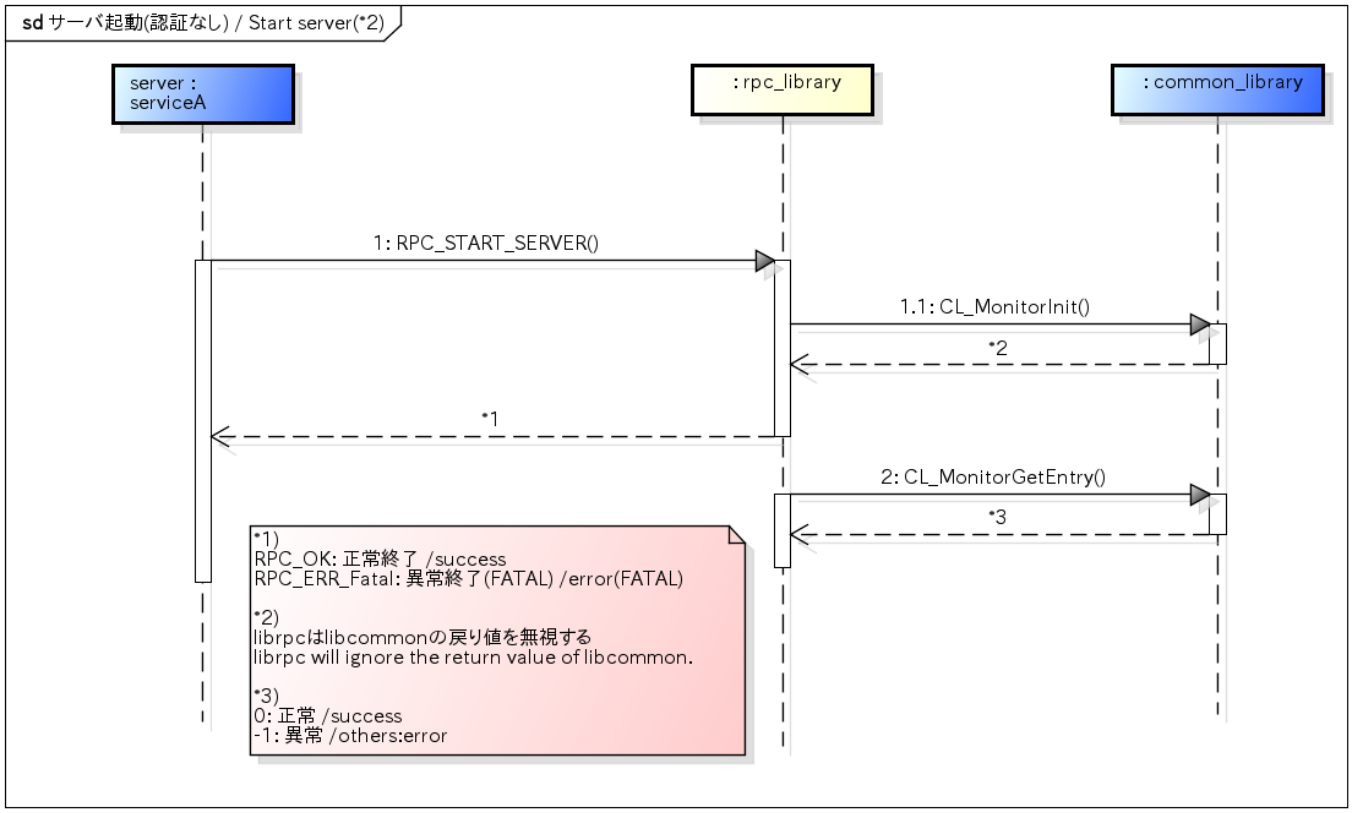
ユースケースrpc_library_StartRPC_002*[use-case rpc_library_StartRPC_002]*

概要 *[Overview]*

RPC-APIのサーバプログラムがRPCライブラリ使用開始処理を行う場合に使用する。(認証なし)

Calling this function when RPC_API's server program start to use the RPC library.(No secure check)

シーケンス *[Sequence]*



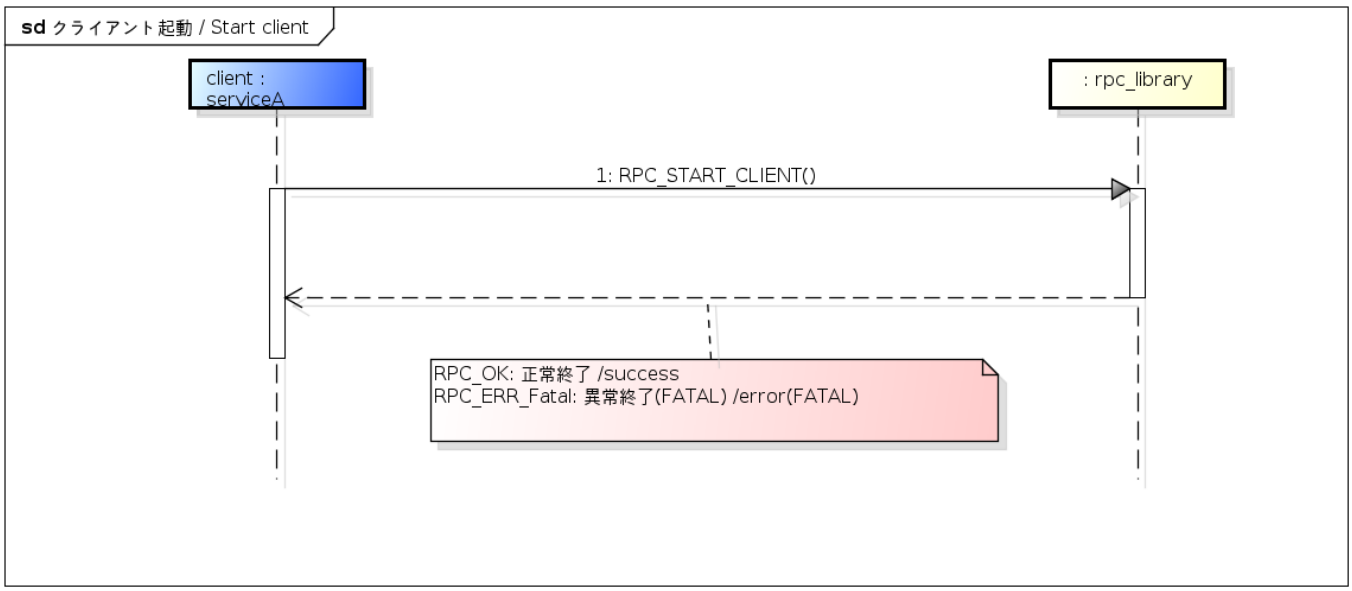
ユースケースrpc_library_StartRPC_003*[use-case rpc_library_StartRPC_003]*

概要 *[Overview]*

RPC-APIのクライアントプログラムがRPCライブラリ使用開始処理を行う場合に使用する。

Calling this function when RPC_API's client program start to use the RPC library.

シーケンス *[Sequence]*



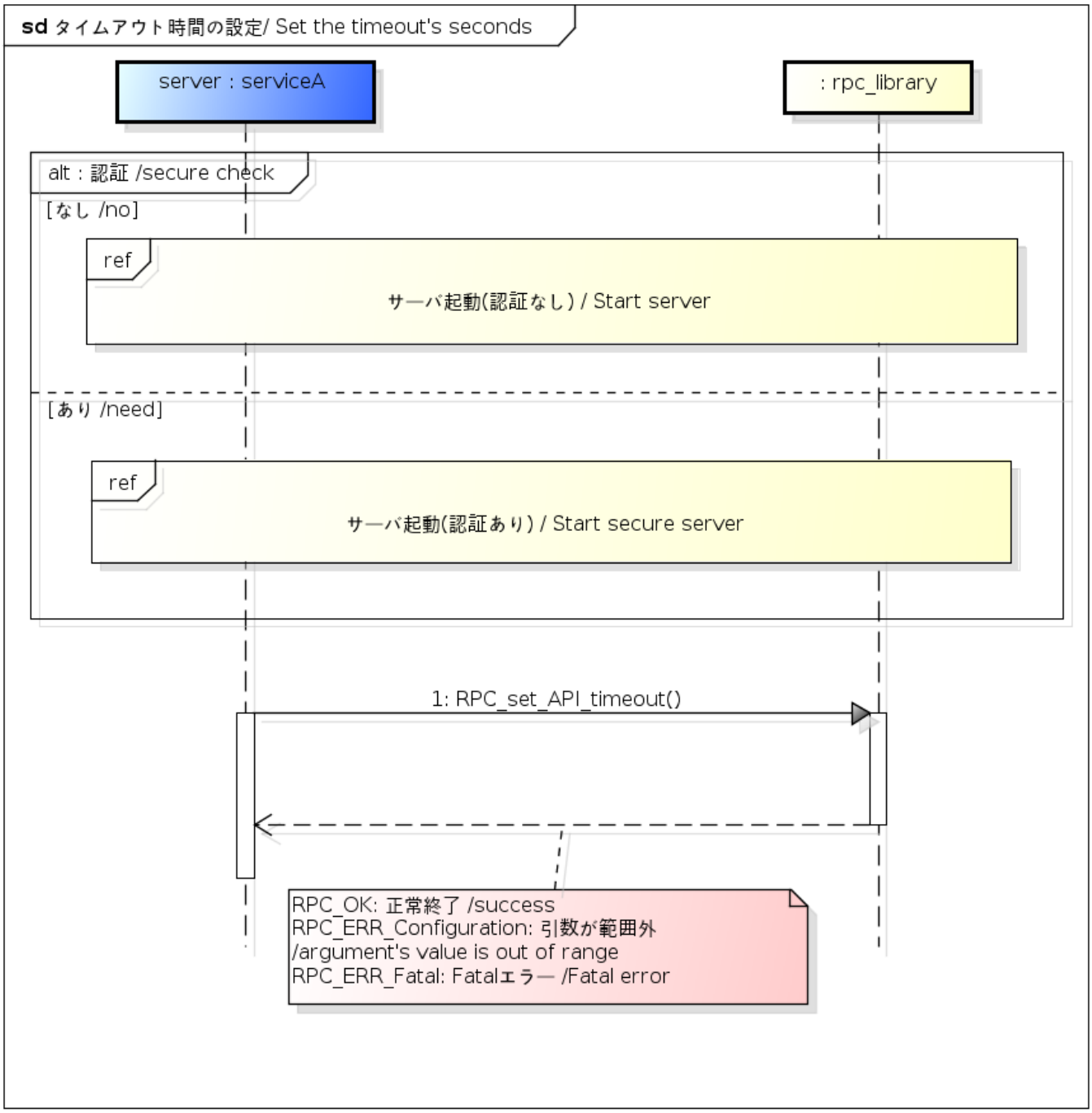
ユースケースrpc_library_*SetAPITimeout_001* *[use-case rpc_library_SetAPItimeout_001]*

概要 *[Overview]*

サーバのAPI処理時のタイムアウト時間の設定。

Setting the timeout's seconds about processing the server's API.

シーケンス *[Sequence]*



powered by Astah

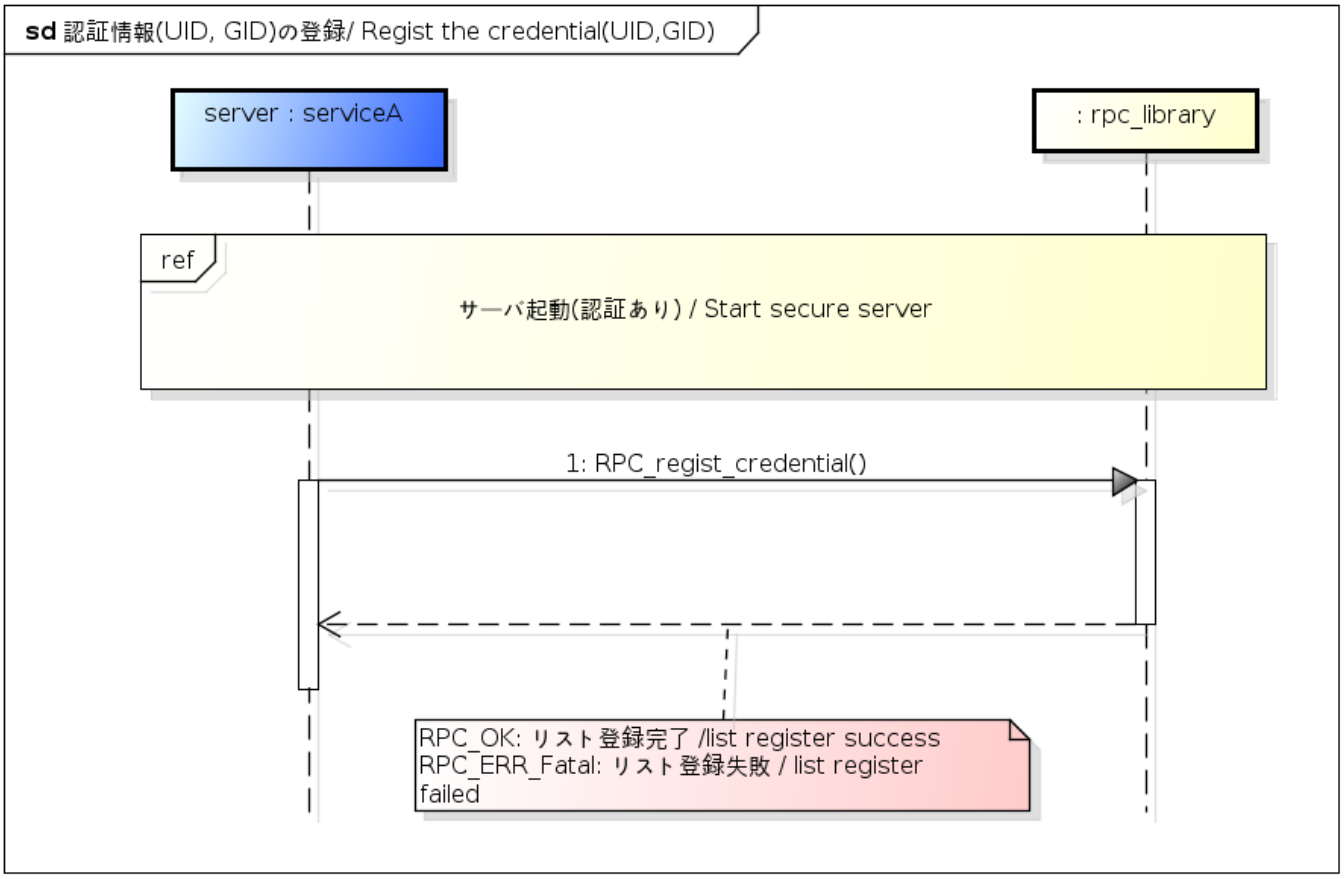
ユースケースrpc_library_RegistAndGetCredential_001 *[use-case rpc_library_RegistAndGetCredential_001]*

概要 *[Overview]*

RPCサーバが受け入れ可能なUID, GIDリストを登録する。

RPC server regist the UID,GID which can be received.

シーケンス *[Sequence]*



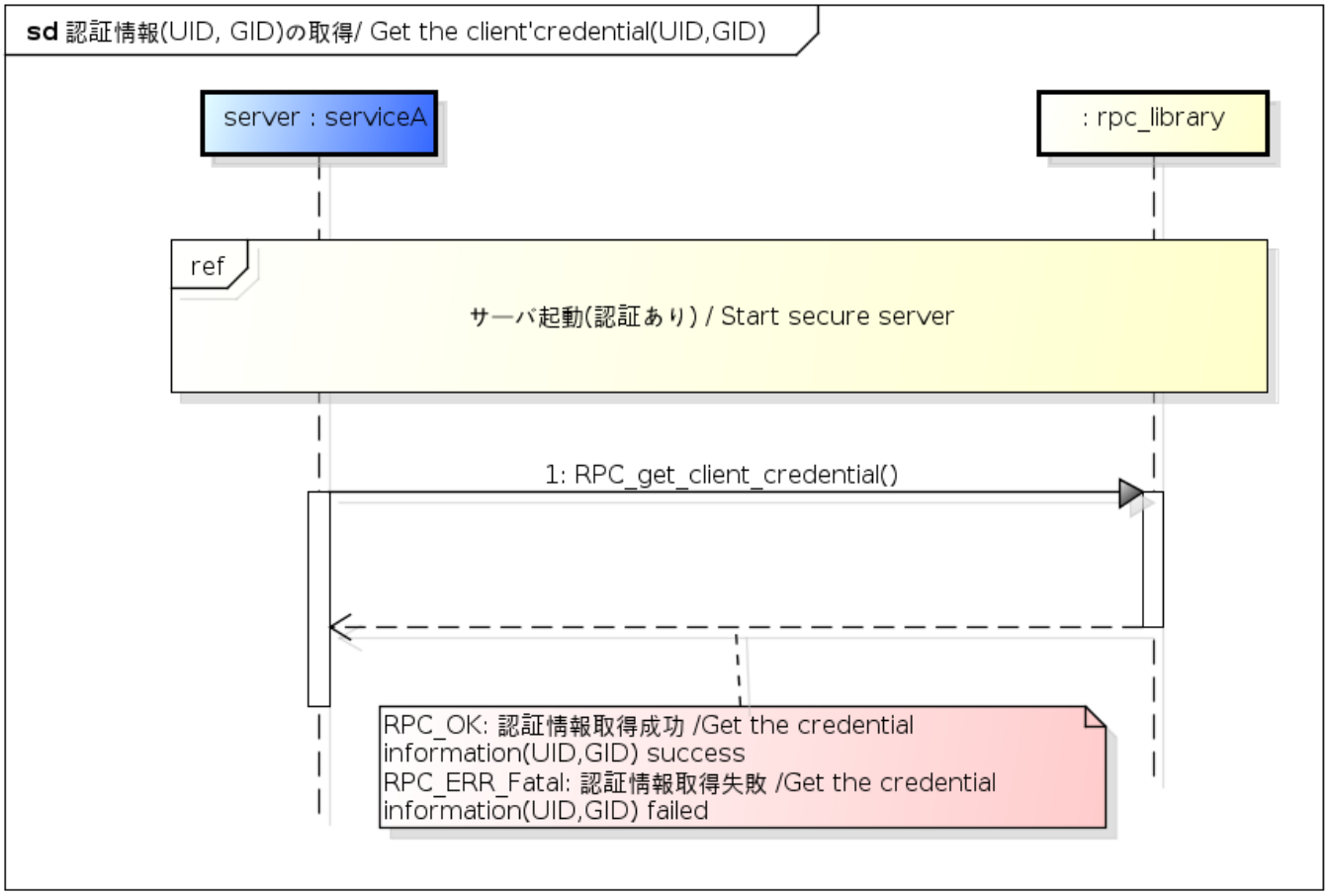
ユースケースrpc_library_RegistAndGetCredential_002*[use-case rpc_library_RegistAndGetCredential_002]*

概要 *[Overview]*

実行中API要求元クライアントの認証情報(UID, GID)を取得する。

Get the client(which called the running API) credential information(UID,GID) .

シーケンス *[Sequence]*



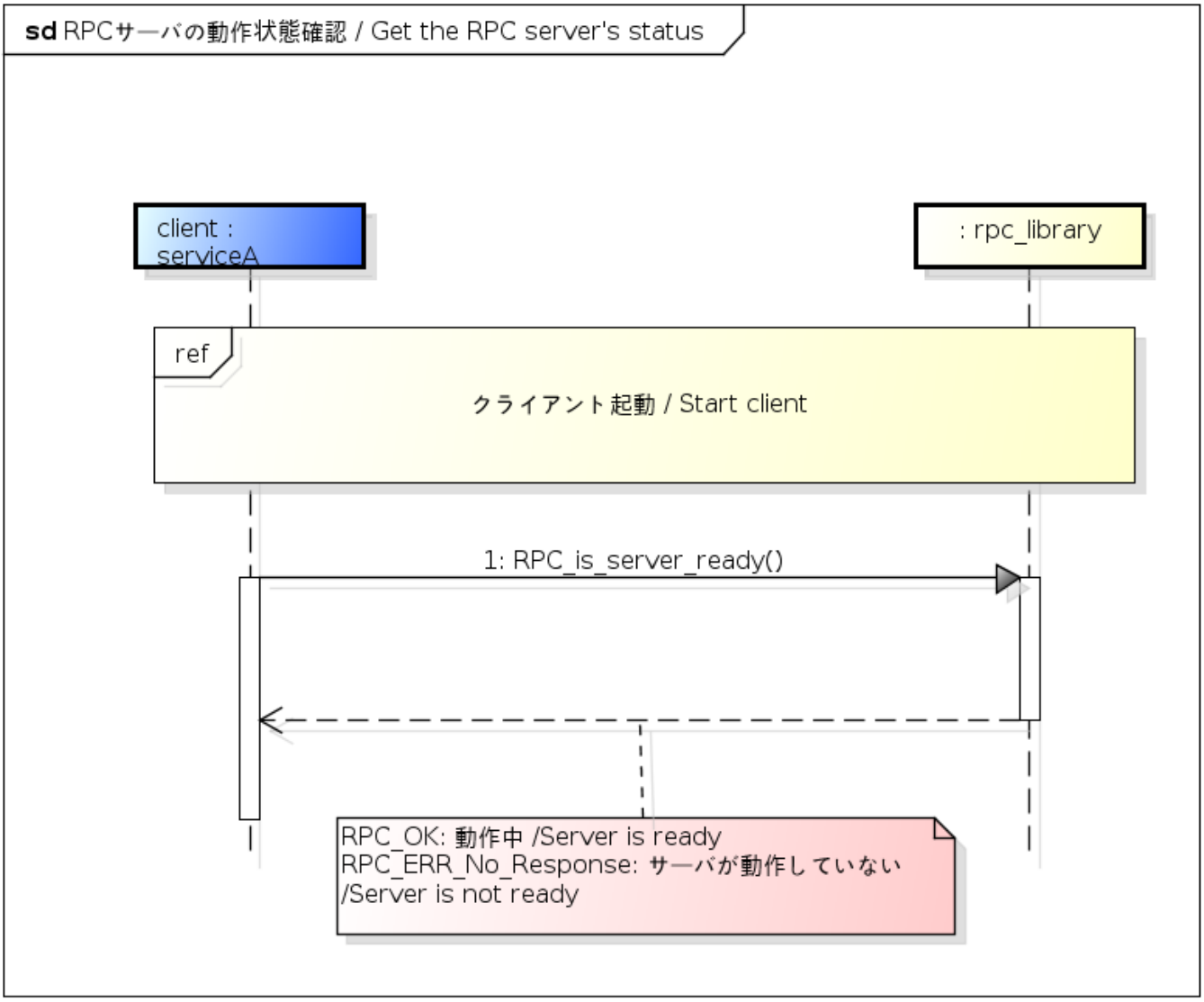
ユースケースrpc_library_GetServerStatus_001 *[use-case rpc_library_GetServerStatus_001]*

概要 *[Overview]*

RPCサーバの動作状態を返す。

Return the server's current status.

シーケンス *[Sequence]*



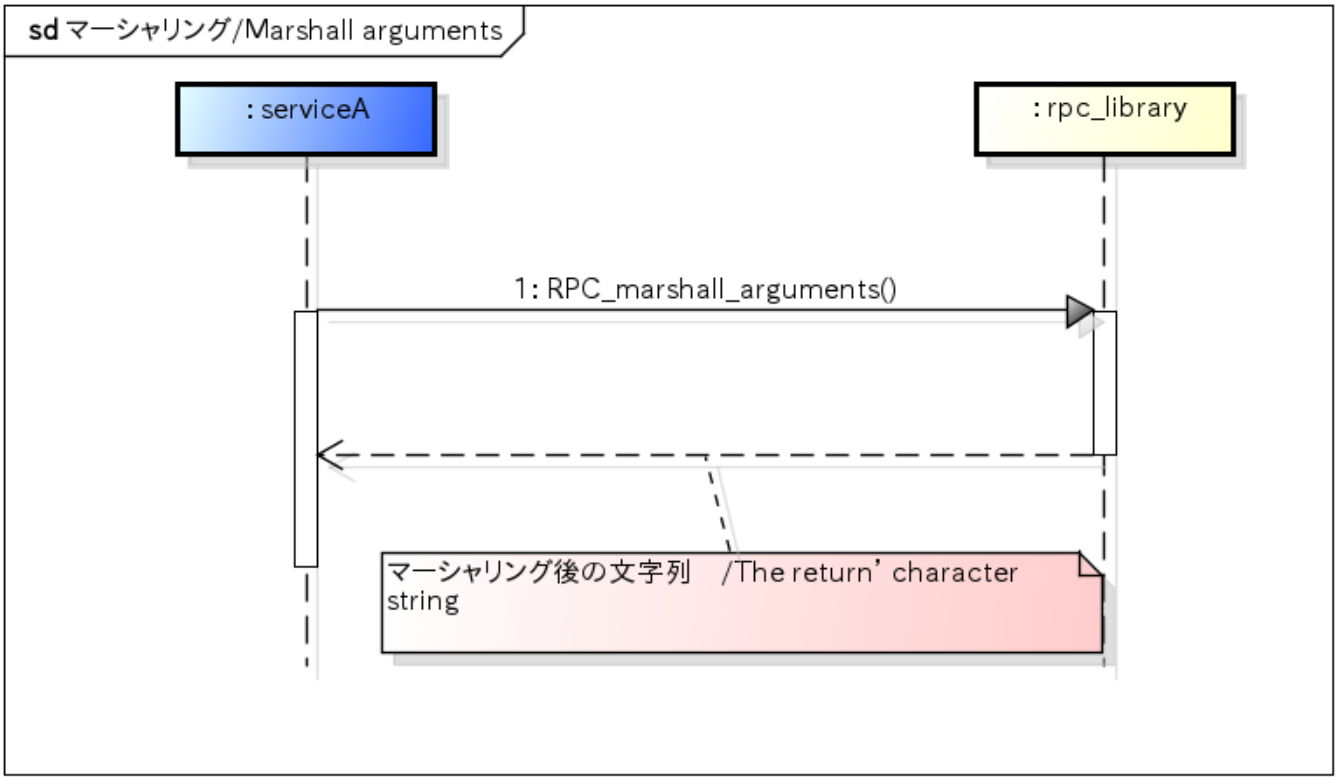
ユースケースrpc_library_CallAndProcessAPI_001 *[use-case rpc_library_CallAndProcessAPI_001]*

概要 *[Overview]*

可変引数を集め、受け取り側の仕様に合った形式に変換して、メッセージバッファに詰め込む。

Collect variable arguments, convert them to a format that matches the receiver's specification, and stuff them into the message buffer.

シーケンス *[Sequence]*



powered by Astah

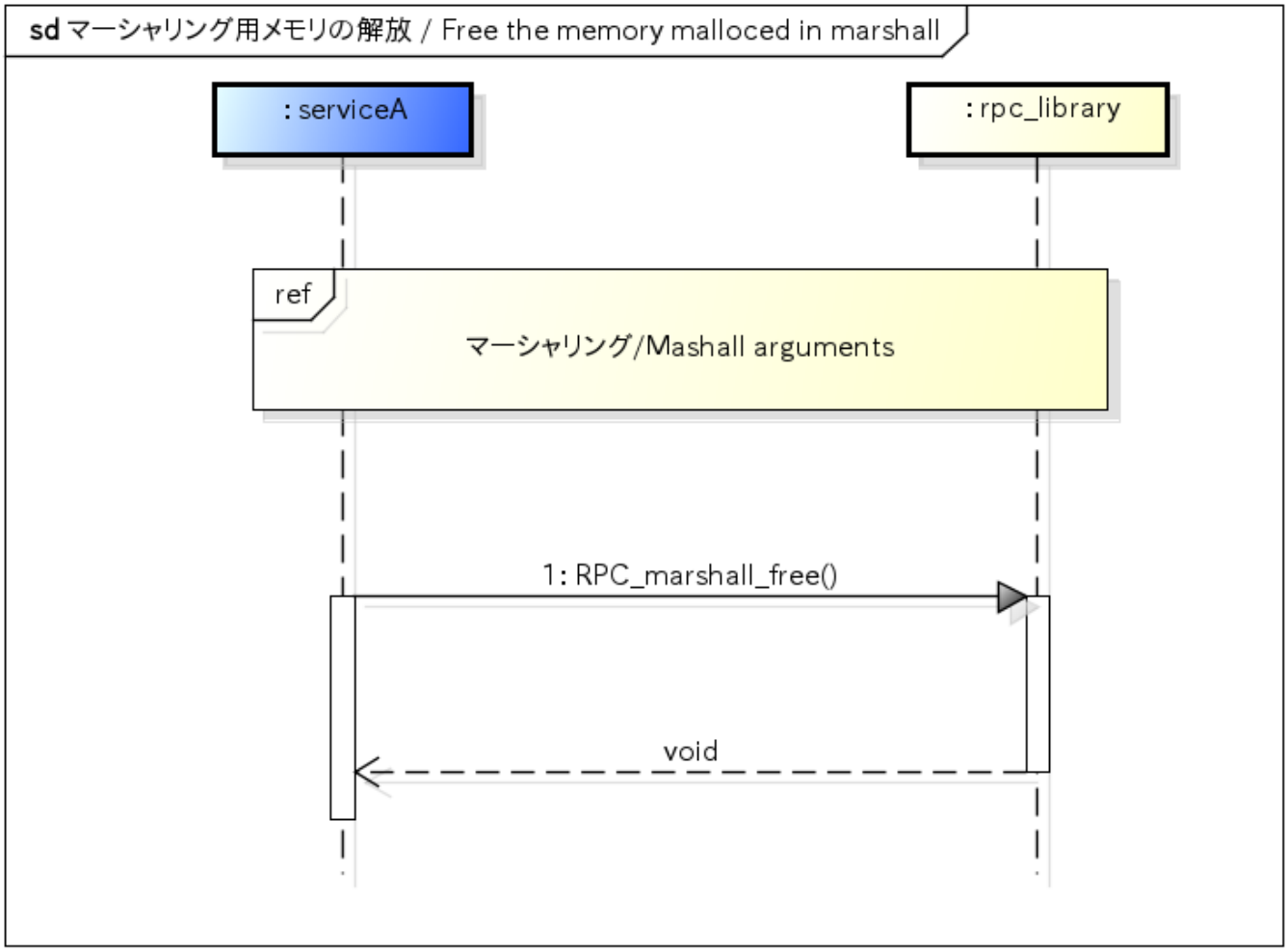
ユースケースrpc_library_CallAndProcessAPI_002*[use-case rpc_library_CallAndProcessAPI_002]*

概要 *[Overview]*

マーシャリング用メモリを解放する。

Free the memory malloced by mashall the arguments.

シーケンス *[Sequence]*

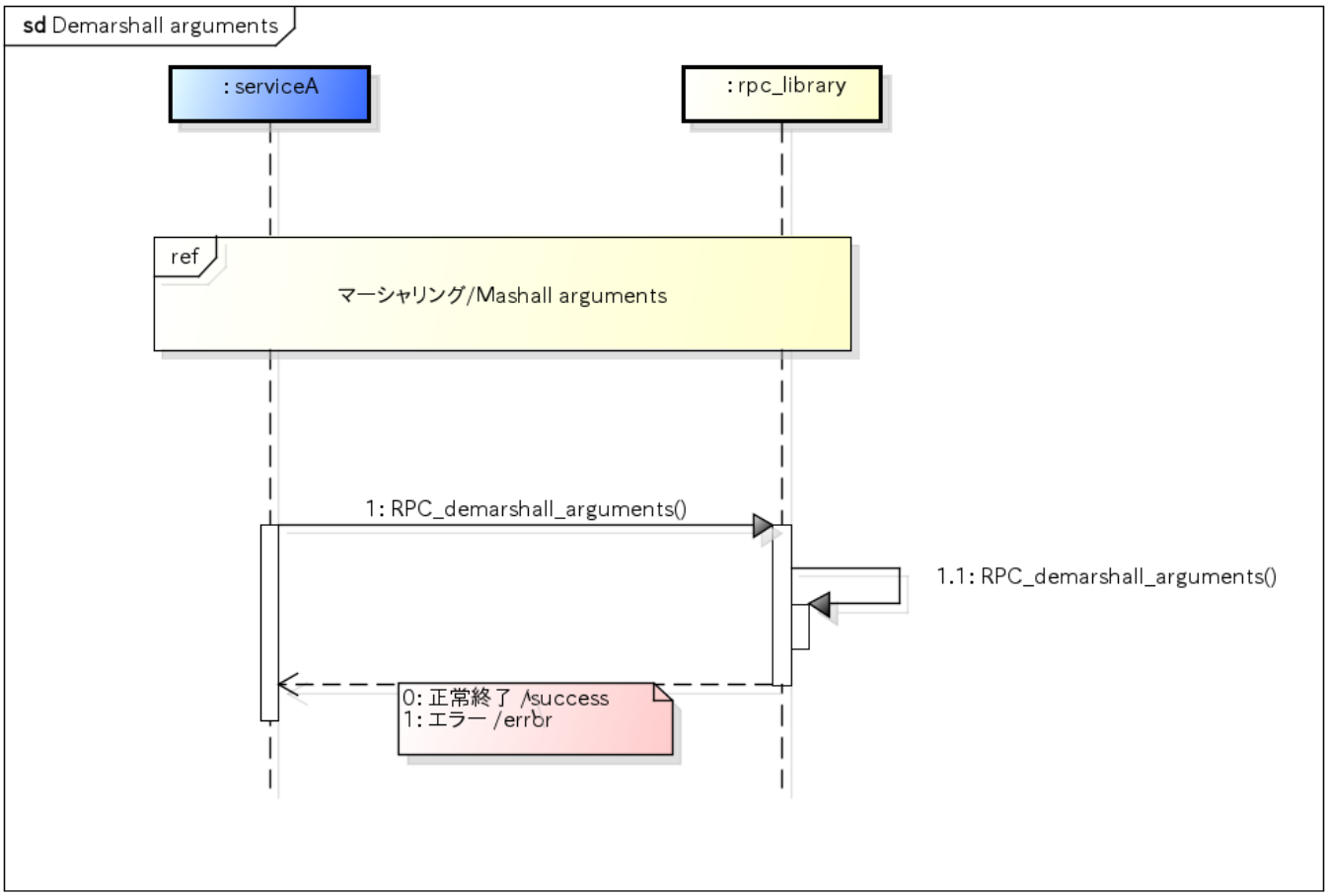


ユースケースrpc_library_CallAndProcessAPI_003*[use-case rpc_library_CallAndProcessAPI_003]*

概要 *[Overview]*

入力文字列が規定されているフォーマットに準拠しているかどうかを確認する。

シーケンス *[Sequence]*



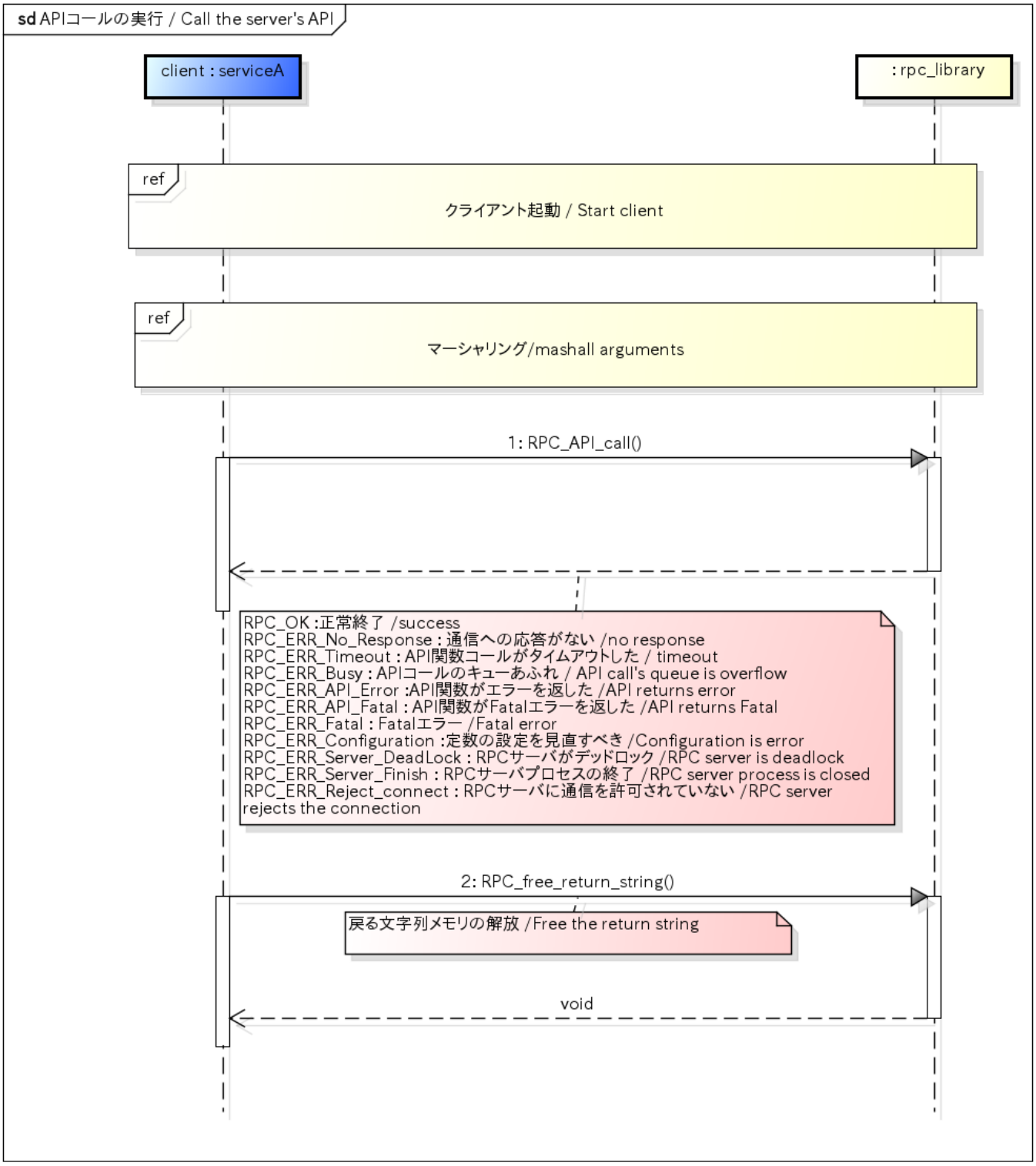
ユースケースrpc_library_CallAndProcessAPI_004*[use-case rpc_library_CallAndProcessAPI_004]*

概要 *[Overview]*

APIコールを実行する(クライアント側)。

Call the server's API call(call by client).

シーケンス *[Sequence]*



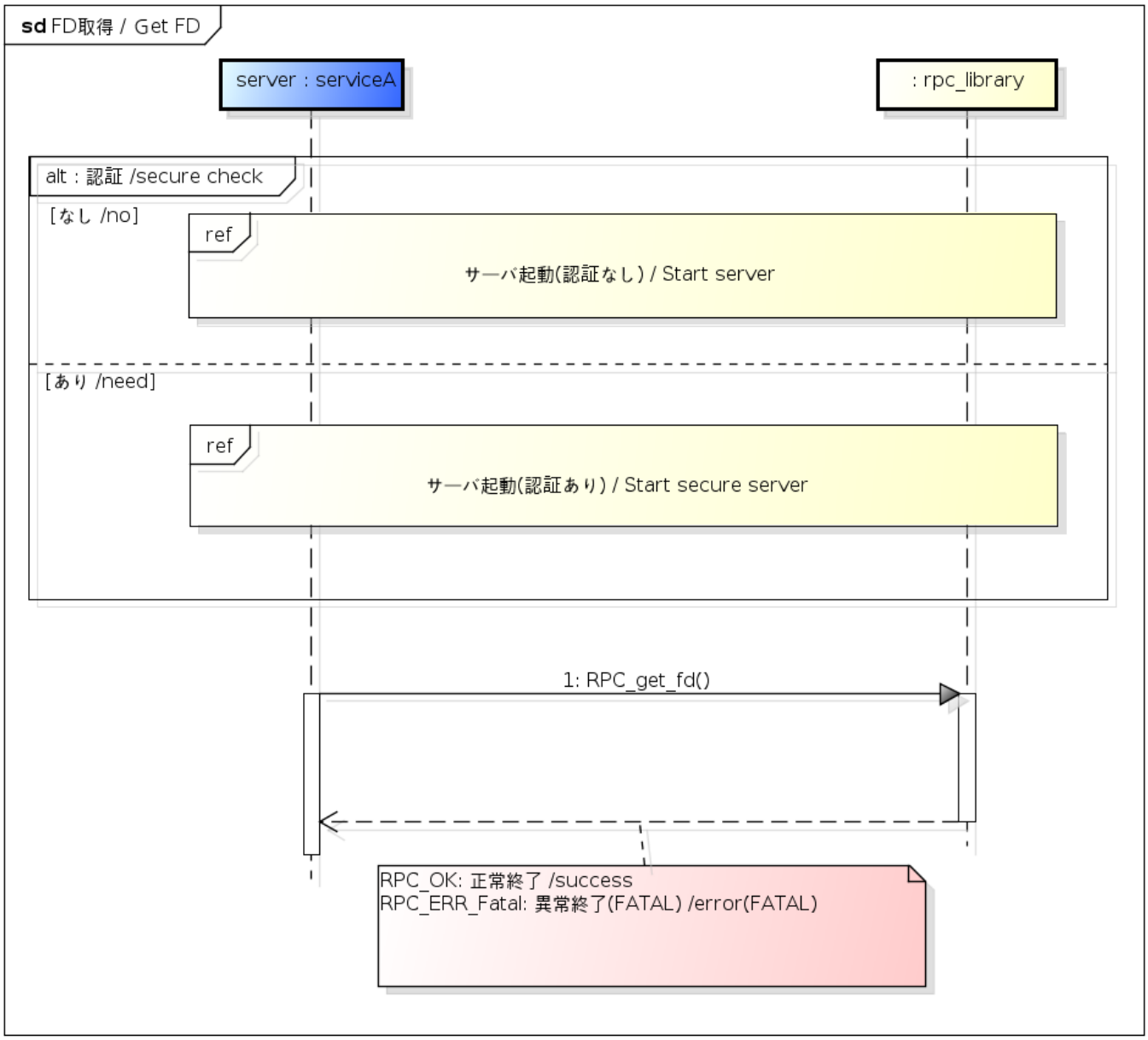
ユースケースrpc_library_CallAndProcessAPI_005*[use-case rpc_library_CallAndProcessAPI_005]*

概要 *[Overview]*

APIコール要求の有無判定のためのFDを取得する。

Get the FD which can use to judge whether the API call's request comes.

シーケンス *[Sequence]*



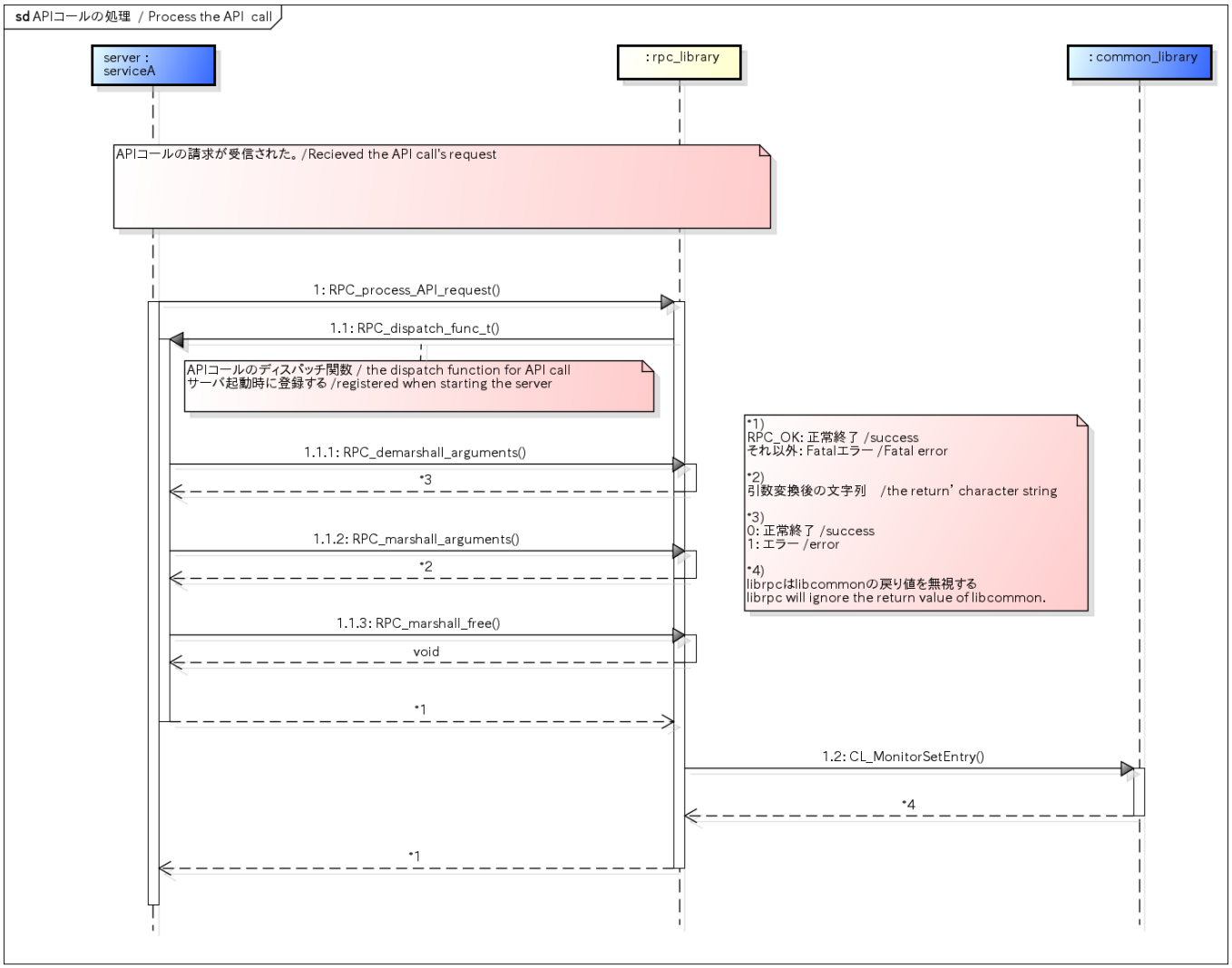
ユースケースrpc_library_CallAndProcessAPI_006*[use-case rpc_library_CallAndProcessAPI_006]*

概要 *[Overview]*

APIコール処理を実行する。

Process the API call's request.

シーケンス *[Sequence]*



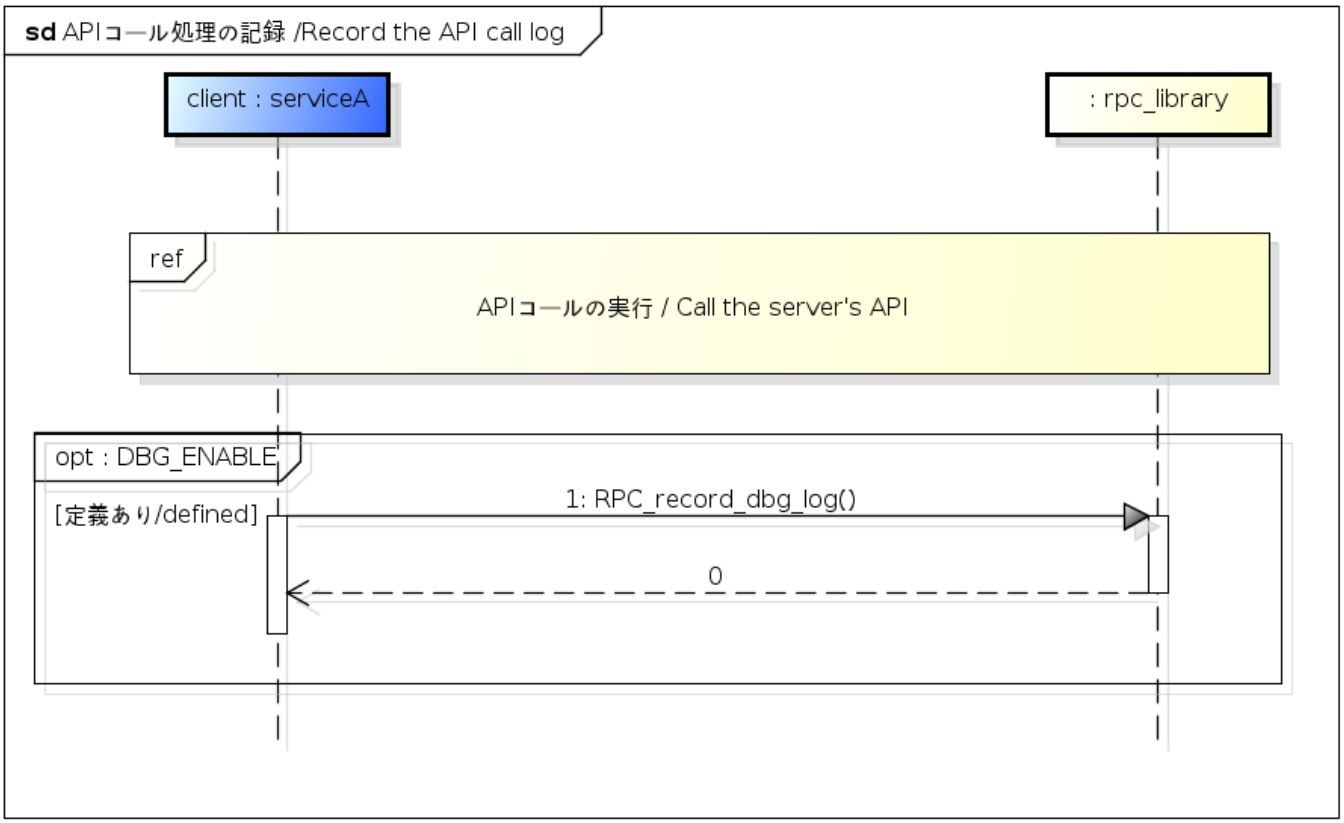
ユースケースrpc_library_CallAndProcessAPI_007 *[use-case rpc_library_CallAndProcessAPI_007]*

概要 [Overview]

APIコールが行われたクライアントのファイル名、関数名、行数、及びコールしたAPI関数名を受け取り記録を行う。

Record the client's filename,function name,line and apiname when running the API call.

シーケンス [Sequence]



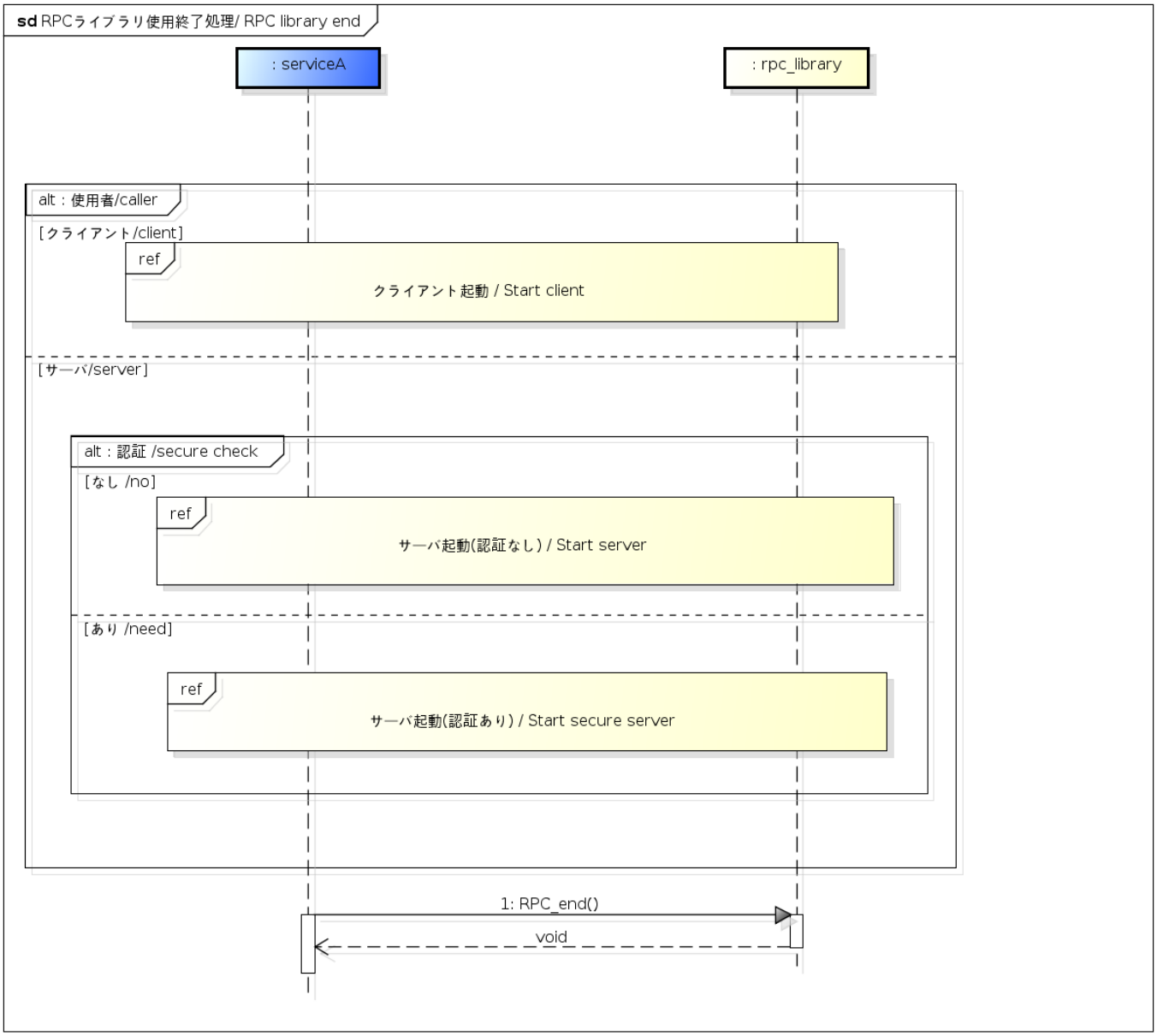
ユースケースrpc_library_EndRPC_001 *[use-case rpc_library_End_RPC_001]*

概要 [Overview]

オープンしたソケットのクローズ、メモリ解放など終了処理を行う。

Process the end request(Close the opened sorket,release the memory etc.)

シーケンス [Sequence]



ユースケースrpc_library_End_RPC_002*[use-case rpc_library_End_RPC_002]*

概要 [Overview]

プロセス内でRPC通信用に作成したソケットファイルを削除する。

Delete the sorket file which maked for RPC communication in the process.

シーケンス [Sequence]