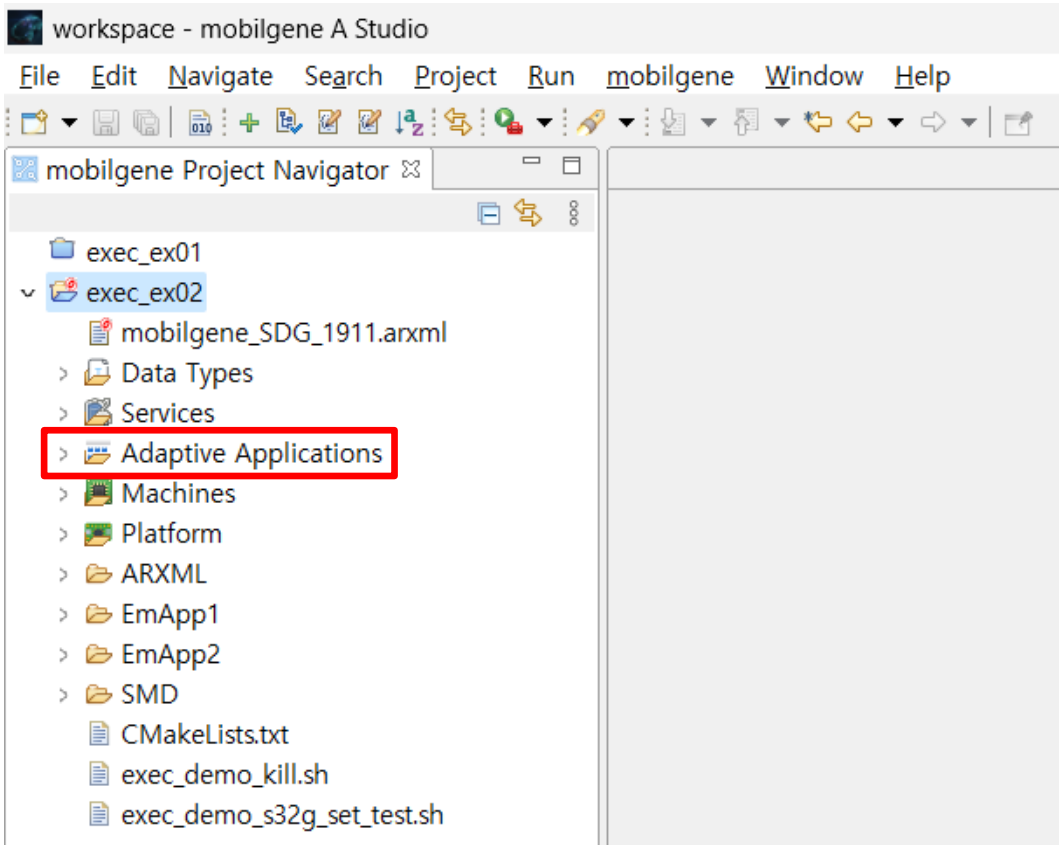
AUTOSAR AP 예제
- EXEC 02 -

Junho Kwak
Architecture and Compiler for Embedded System LAB.
School of Electronics Engineering, KNU, KOREA
2024-07-07

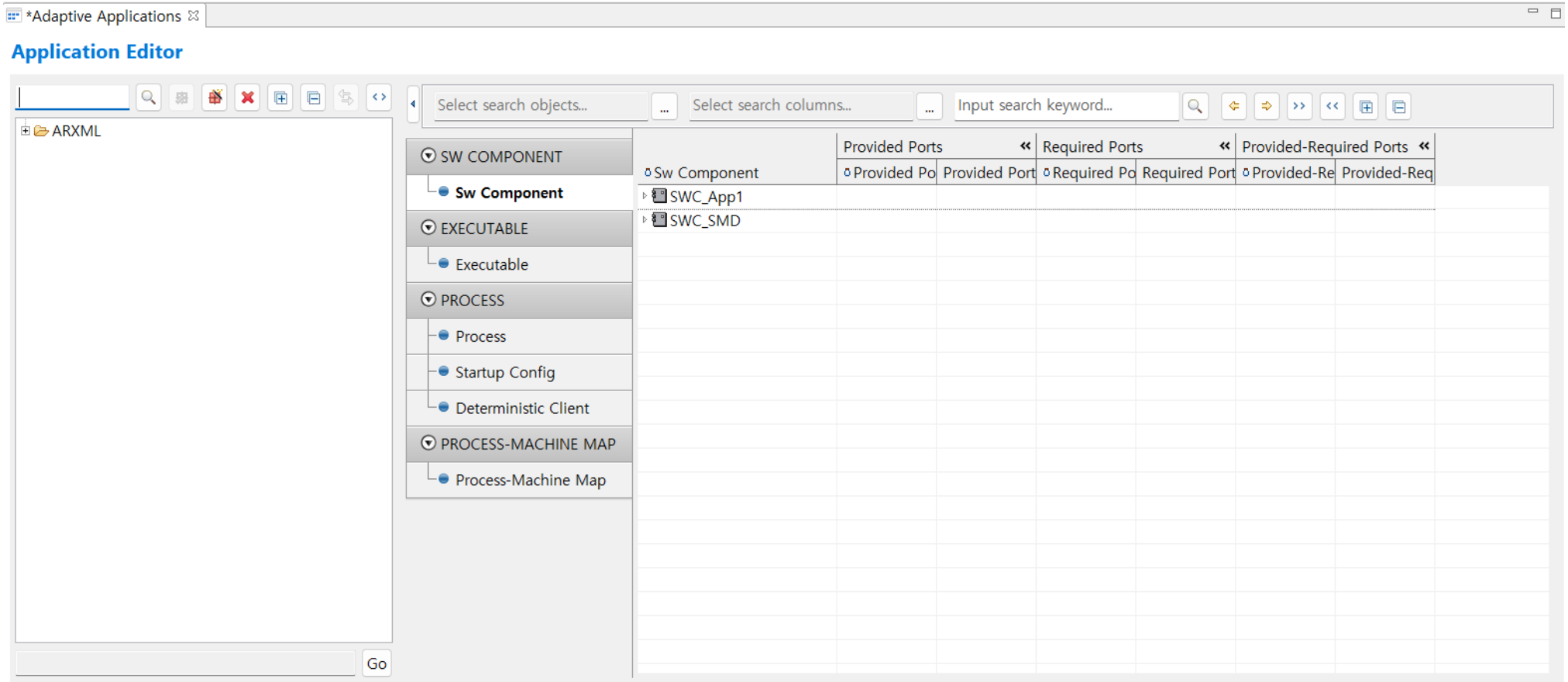
EXEC: Adaptive Applications 설정

- **Adaptive Applications Editor 활성화**
  - ✓ 왼쪽의 'mobilgene Project Navigator' 창에서 해당 Project의 'Adaptive Applications'를 더블 클릭함



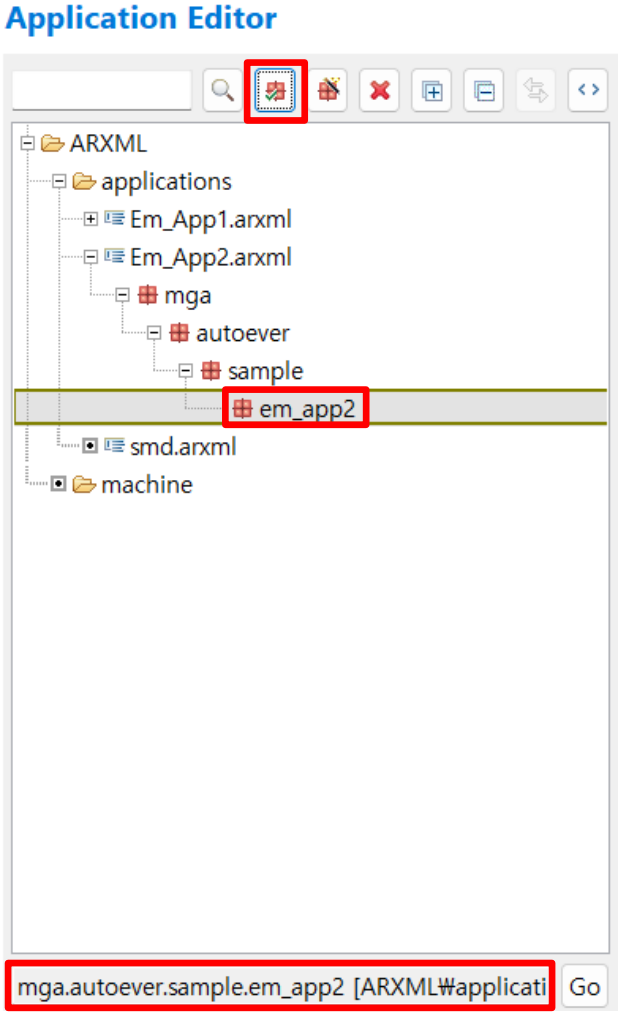
EXEC: Adaptive Applications 설정

- **Adaptive Applications Editor 활성화 확인**
  - ✓ 활성화 된 Adaptive Applications Editor를 확인함



EXEC: Adaptive Applications 설정

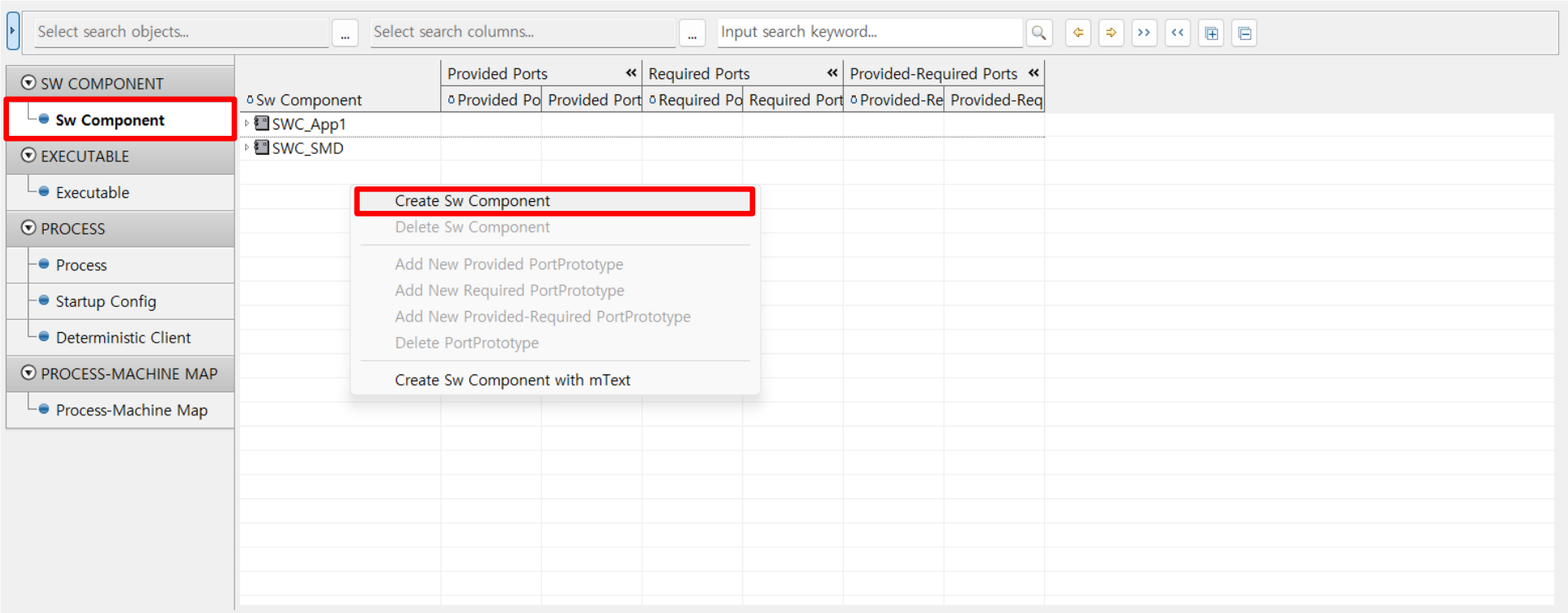
- **Default Package 설정**
  - ✓ 설정이 저장될 Default Package를 설정함
    - ✓ Default Package로 설정할 'em_app2' Package를 선택함
    - ✓ 우측 상단의 'Set Default Package' 버튼을 클릭함
    - ✓ 하단에 설정된 Default Package를 확인함



EXEC: Adaptive Applications 설정

- **Sw Component 추가**
  - ✓ Sw Component 추가를 위해 좌측의 'Sw Component' 탭으로 이동함
  - ✓ 빈 곳에서 우클릭을 하여 'Create Sw Component'를 클릭함

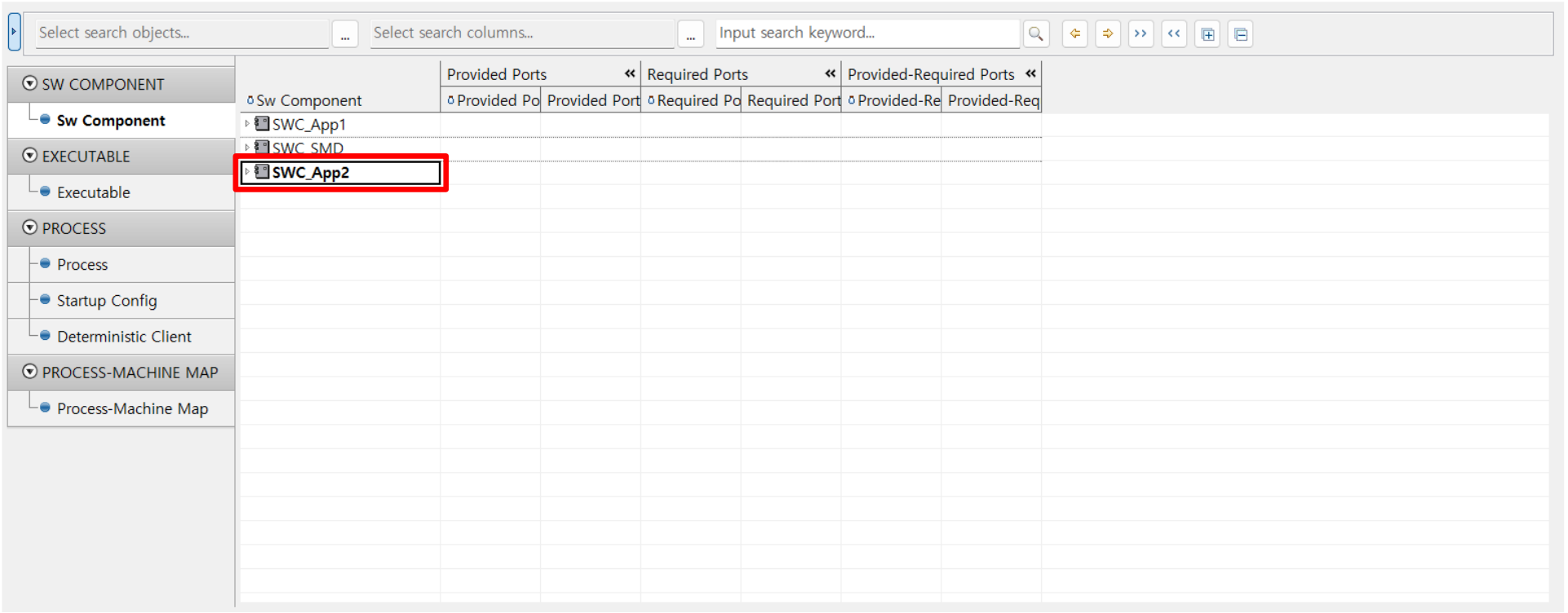
Application Editor



EXEC: Adaptive Applications 설정

- **Sw Component 추가 확인 및 설정**
  - ✓ 생성된 새로운 Sw Component를 확인하고 다음과 같이 수정함
    - ✓ Sw Component : 'SWC_App2'

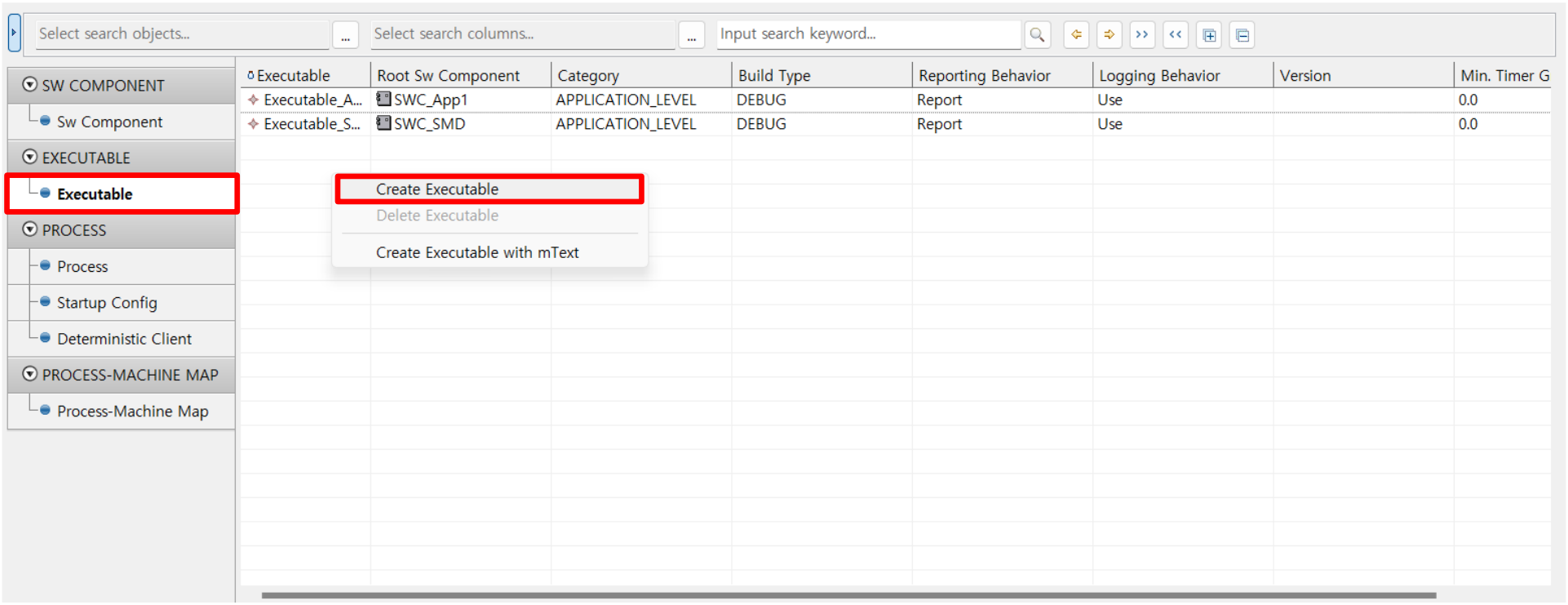
Application Editor



EXEC: Adaptive Applications 설정

- **Executable 추가**
  - ✓ Executable 추가를 위해 좌측의 'Executable' 탭으로 이동함
  - ✓ 빈 곳에서 우클릭을 하여 'Create Executable'을 클릭함

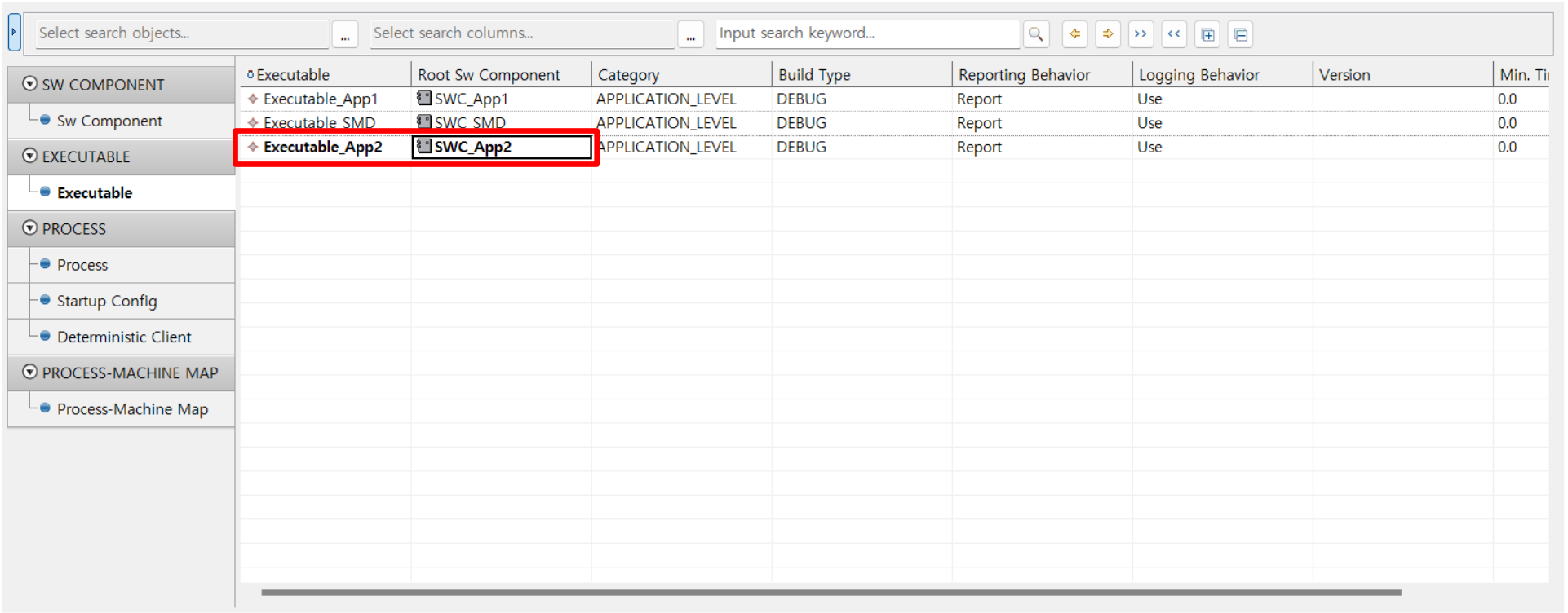
Application Editor



EXEC: Adaptive Applications 설정

- **Executable 추가 확인 및 설정**
  - ✓ 생성된 새로운 Executable을 확인하고 다음과 같이 수정함
    - ✓ Executable : 'Executable_App2'
    - ✓ Root Sw Component : 'SWC_App2'

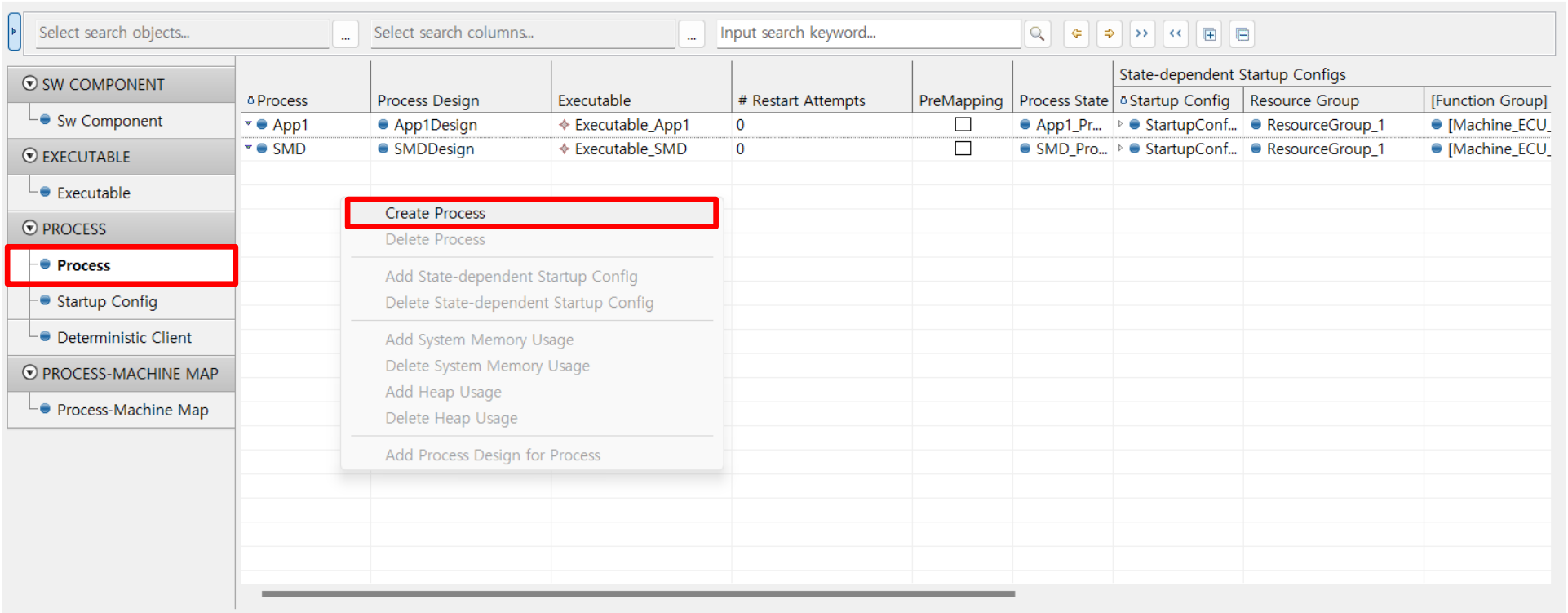
Application Editor



EXEC: Adaptive Applications 설정

- **Process 추가**
  - ✓ Process 추가를 위해 좌측의 'Process' 탭으로 이동함
  - ✓ 빈 곳에서 우클릭을 하여 'Create Process'를 클릭함

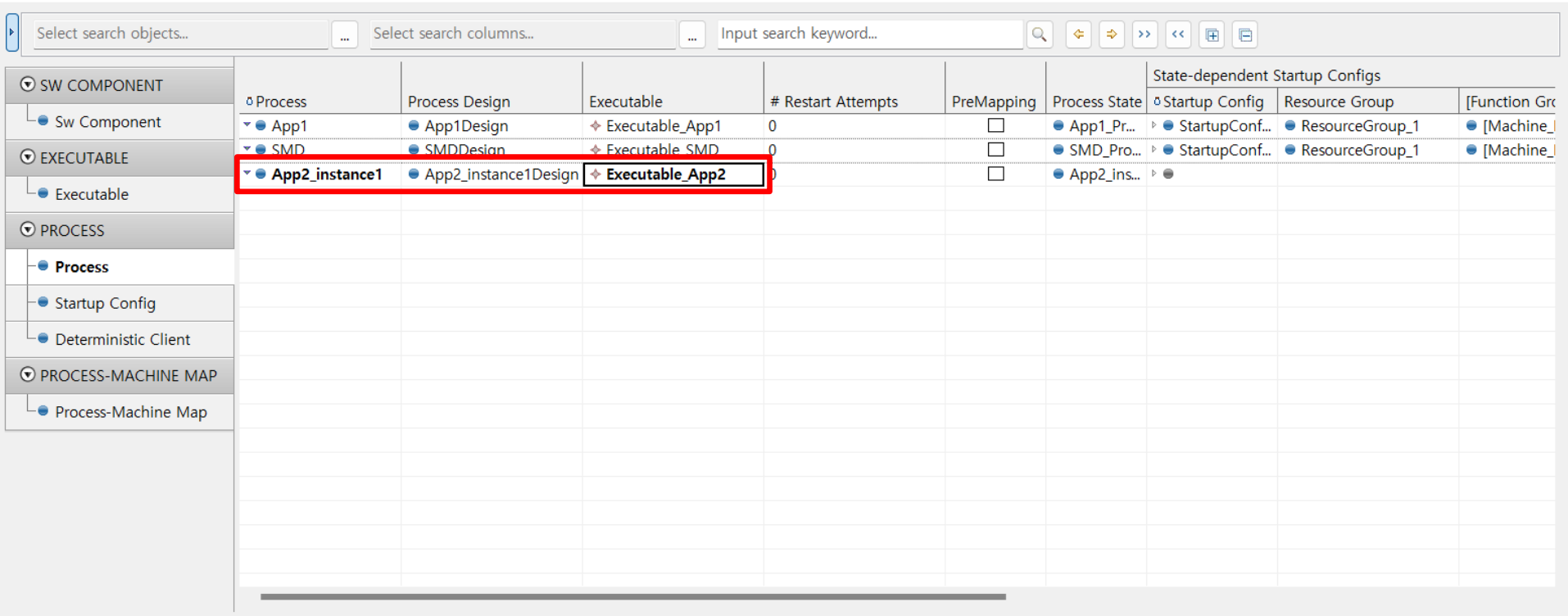
Application Editor



EXEC: Adaptive Applications 설정

- **Process 추가 확인 및 설정**
  - ✓ 생성된 새로운 Process를 확인하고 다음과 같이 수정함
    - ✓ Process : 'App2_instance1'
    - ✓ Executable : 'Executable_App2'

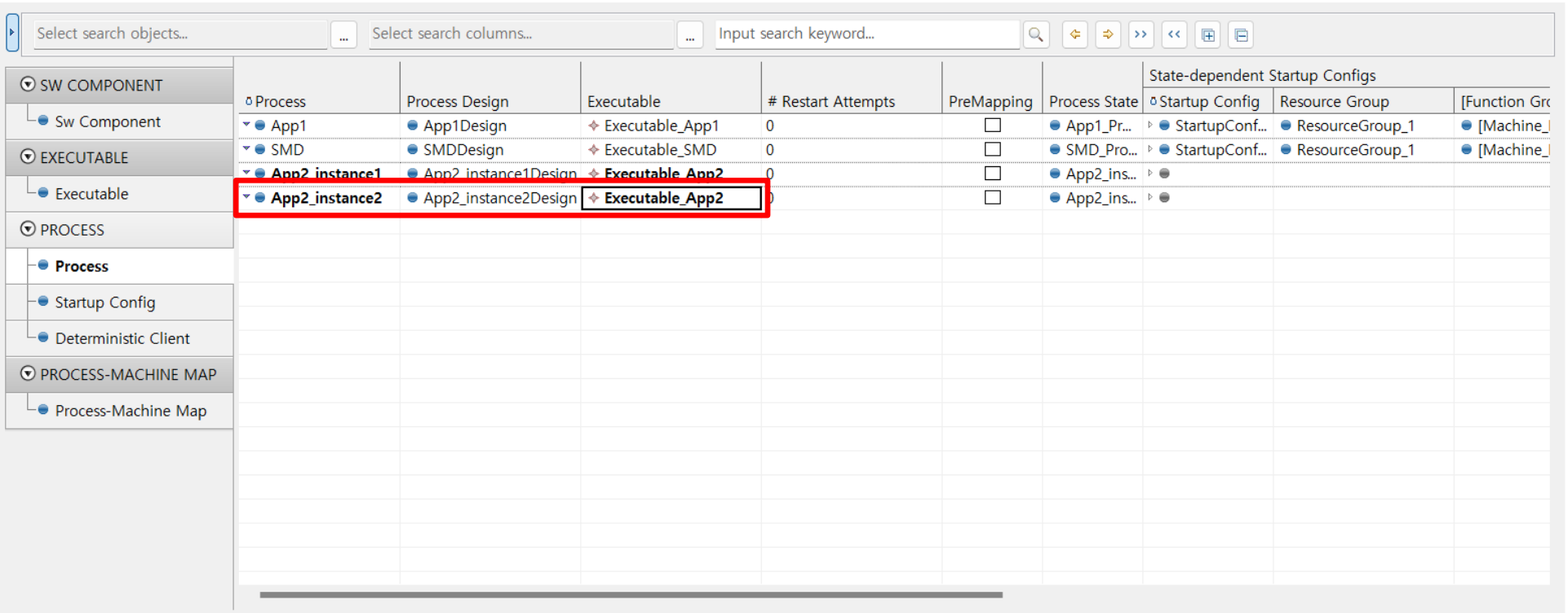
Application Editor



EXEC: Adaptive Applications 설정

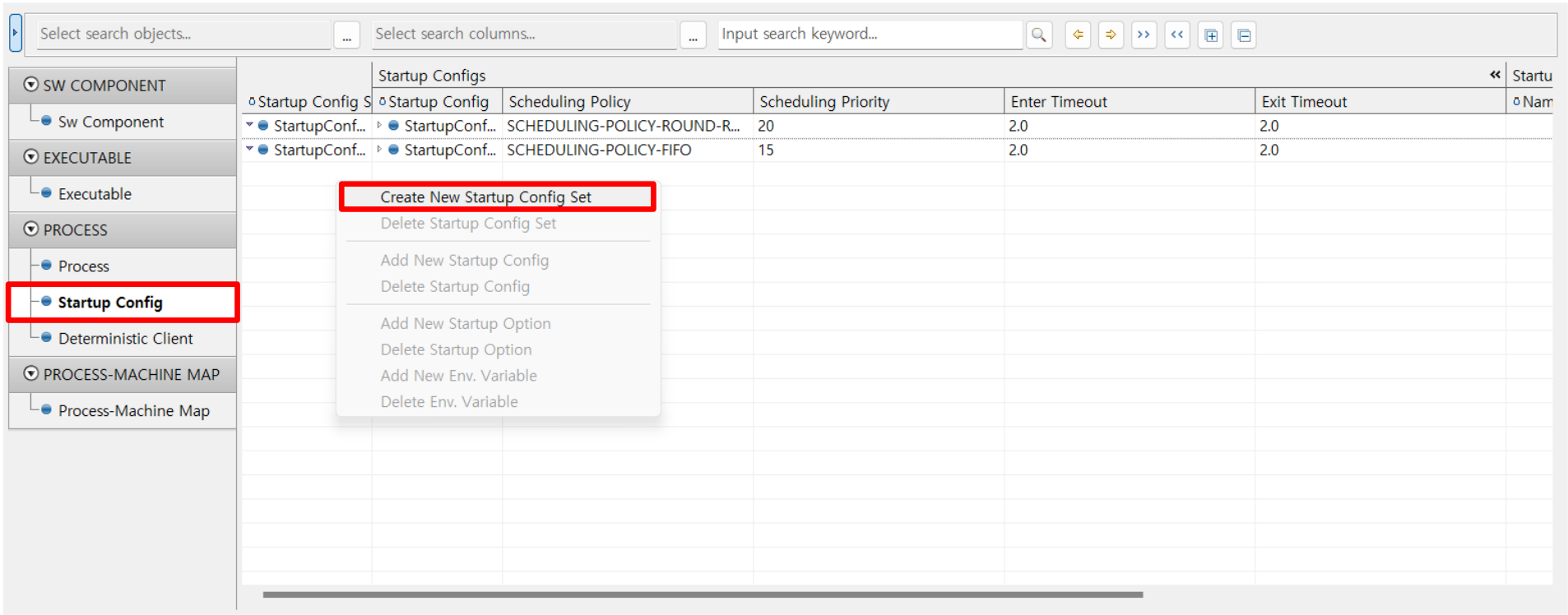
- **Process 추가 설정**
  - ✓ 동일한 Executable로부터 하나의 Process를 더 추가하고 다음과 같이 수정함
    - ✓ Process : 'App2_instance2'
    - ✓ Executable : 'Executable_App2'

Application Editor



EXEC: Adaptive Applications 설정

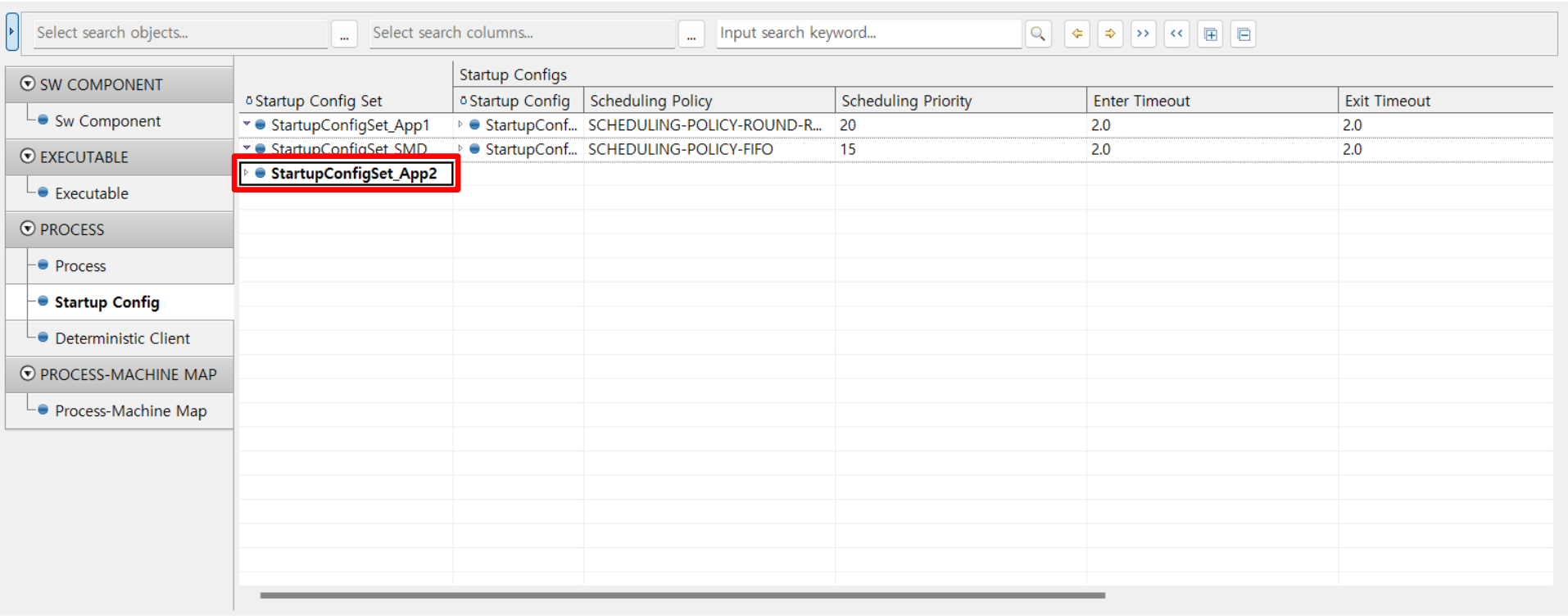
- **Startup Config Set 추가**
  - ✓ Startup Config Set 추가를 위해 좌측의 'Startup Config' 탭으로 이동함
  - ✓ 빈 곳에서 우클릭을 하여 'Create New Startup Config Set'을 클릭함



EXEC: Adaptive Applications 설정

- **Startup Config Set 추가 확인 및 설정**
  - ✓ 생성된 새로운 Startup Config Set을 확인하고 다음과 같이 수정함
    - ✓ Startup Config Set : 'StartupConfigSet_App2'

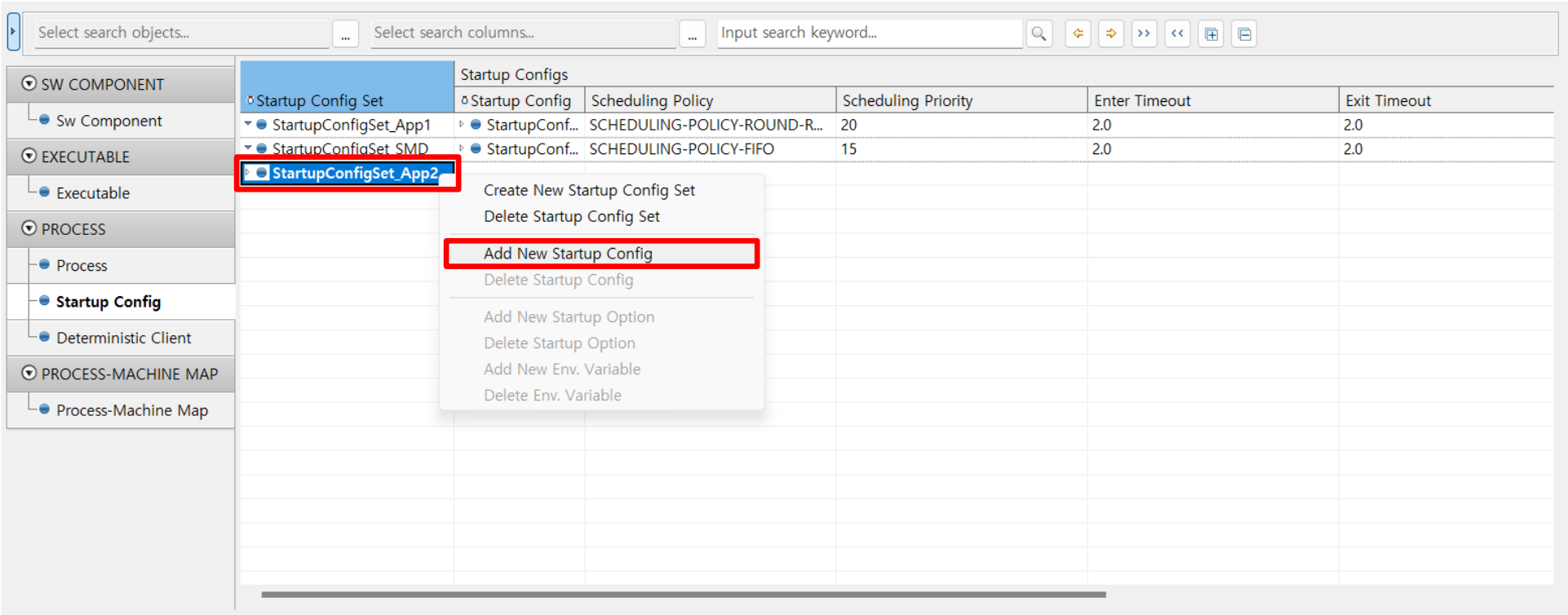
Application Editor



EXEC: Adaptive Applications 설정

- **Startup Config 추가**
  - ✓ 생성한 Startup Config Set에서 우클릭하여 'Add New Startup Config'를 클릭함

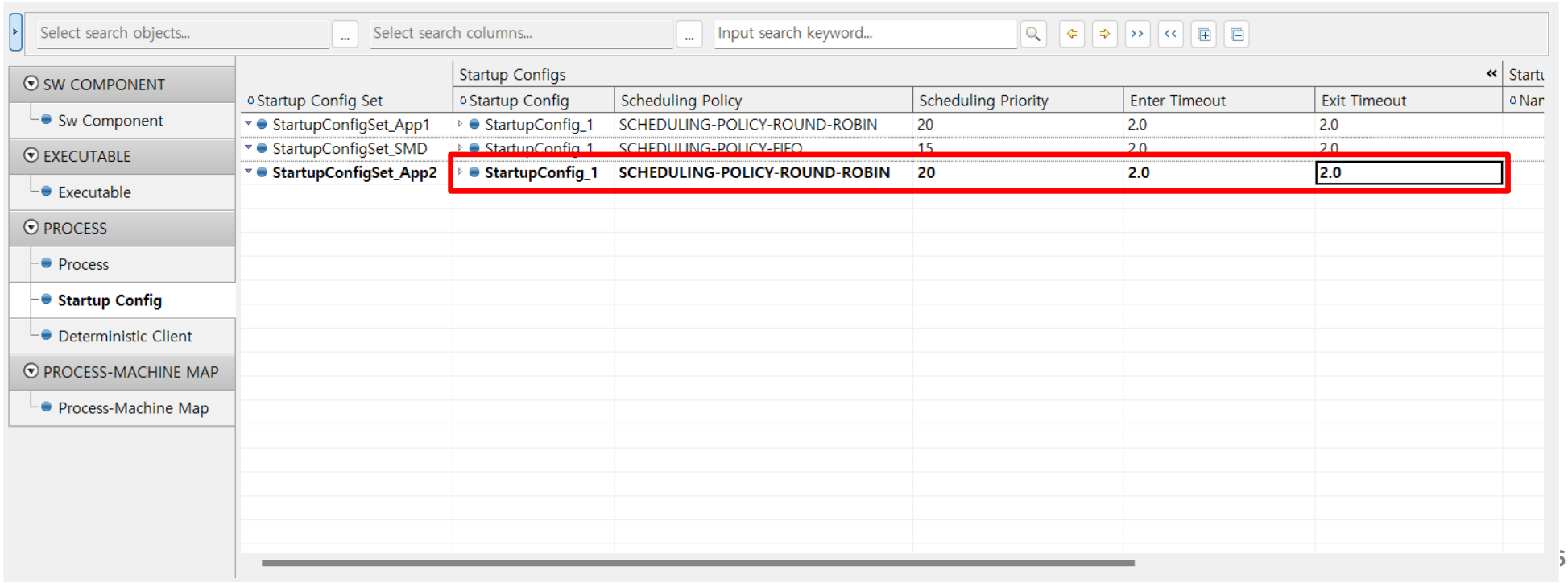
Application Editor



EXEC: Adaptive Applications 설정

- **Startup Config 추가 확인 및 설정**
  - ✓ 생성된 새로운 Startup Config를 확인하고 다음과 같이 수정함
    - ✓ Scheduling Policy : 'SCHEDULING-POLICY-ROUND-ROBIN'
    - ✓ Scheduling Priority : '20'
    - ✓ Enter/Exit Timeout : '2.0'

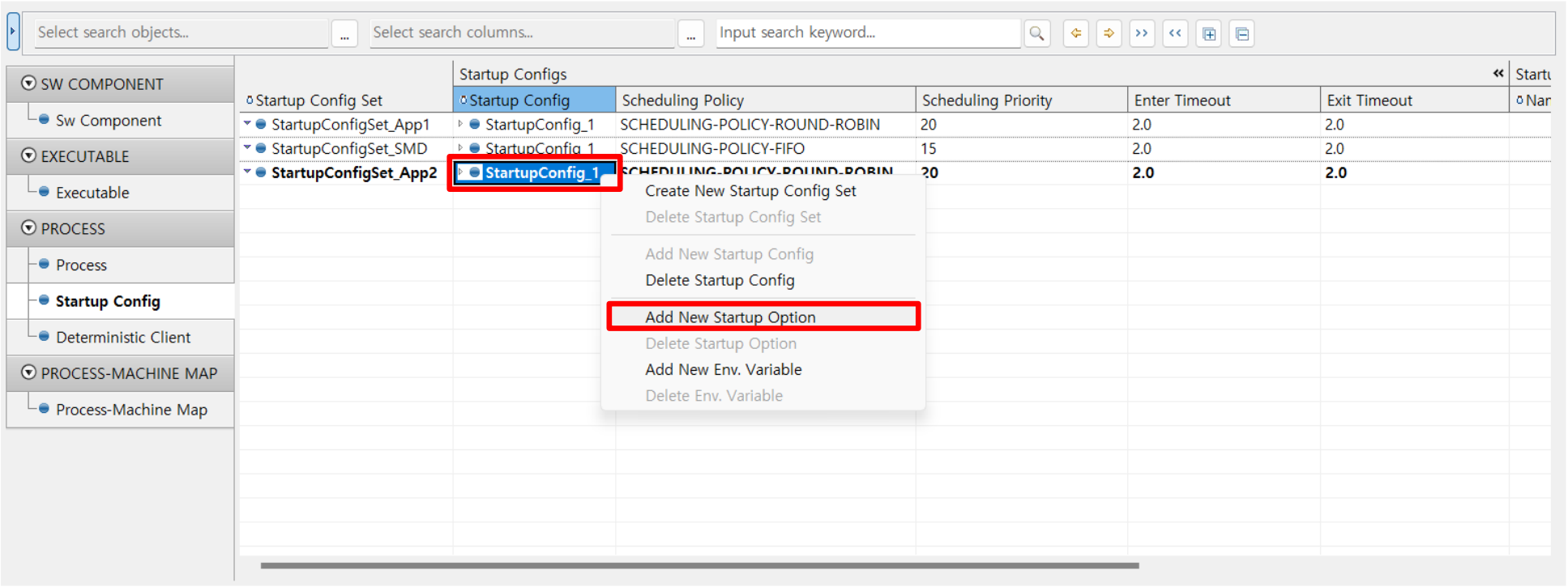
Application Editor



EXEC: Adaptive Applications 설정

- **Startup Option 추가**
  - ✓ 생성한 Startup Config에서 우클릭하여 'Add New Startup Option'을 클릭함

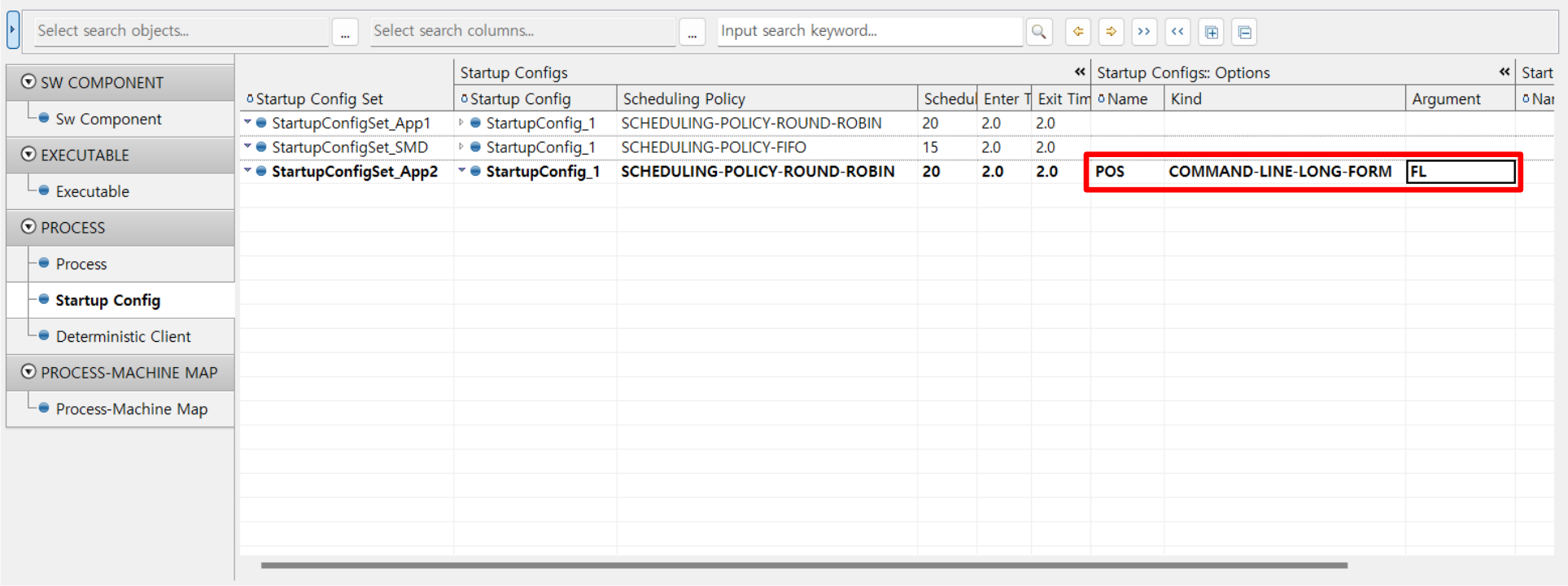
Application Editor



EXEC: Adaptive Applications 설정

- **Startup Option 추가 확인 및 설정**
  - ✓ 생성된 새로운 Startup Option을 확인하고 다음과 같이 수정함
    - ✓ Name : 'POS'
    - ✓ Kind : 'COMMAND-LINE-LONG-FORM'
    - ✓ Argument : 'FL'

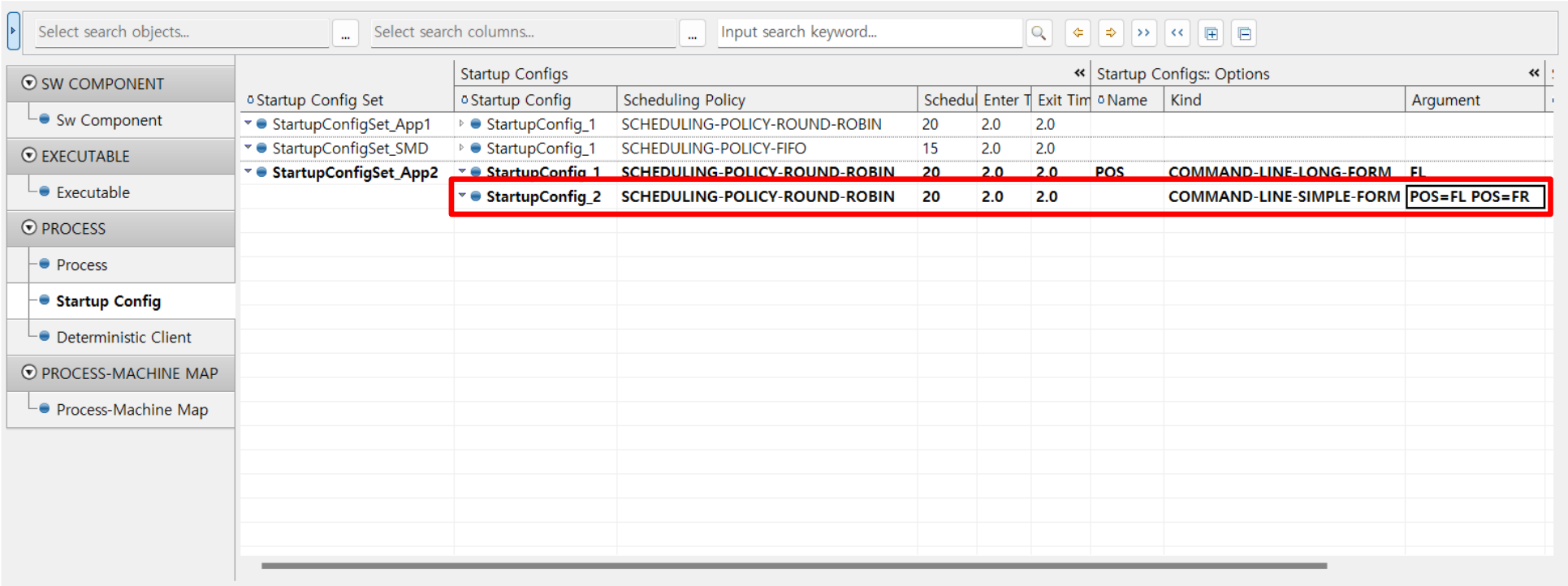
Application Editor



EXEC: Adaptive Applications 설정

- **Startup Config 추가 설정**
  - ✓ 동일한 과정을 통해 'StartupConfigSet_App2' 하위에 하나의 Startup Config를 추가하고 다음과 같이 수정함

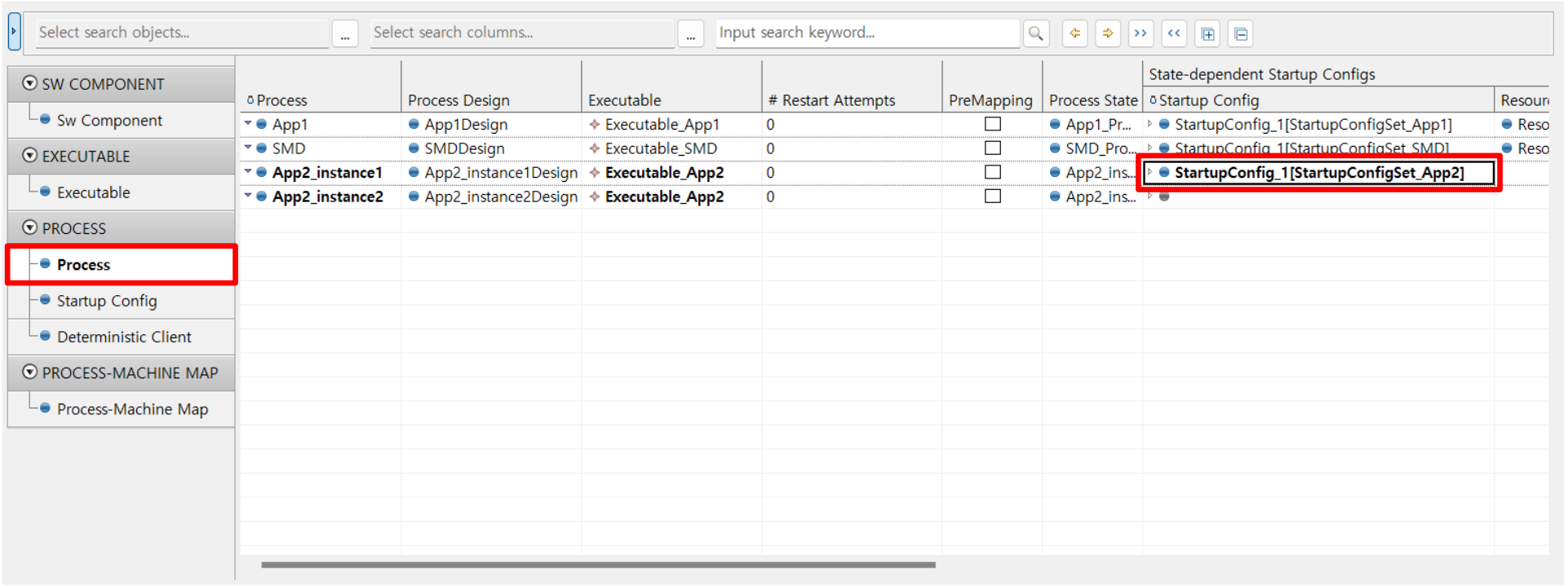
Application Editor



EXEC: Adaptive Applications 설정

- **Startup Config 연결**
  - ✓ 생성한 Startup Config를 Process에 연결하기 위해 좌측의 'Process' 탭으로 이동함
  - ✓ 'App2_instance1'의 Startup Config를 생성한 'StartupConfigSet_App2' – 'StartupConfig_1'로 수정함

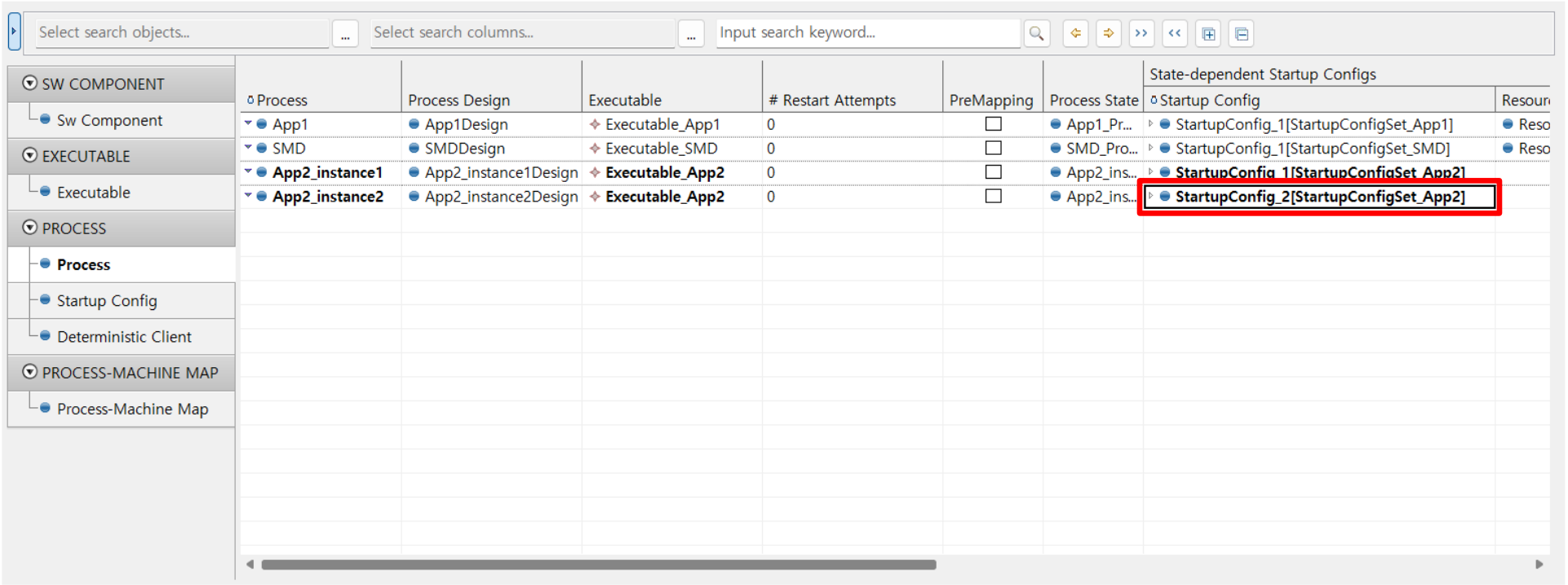
Application Editor



EXEC: Adaptive Applications 설정

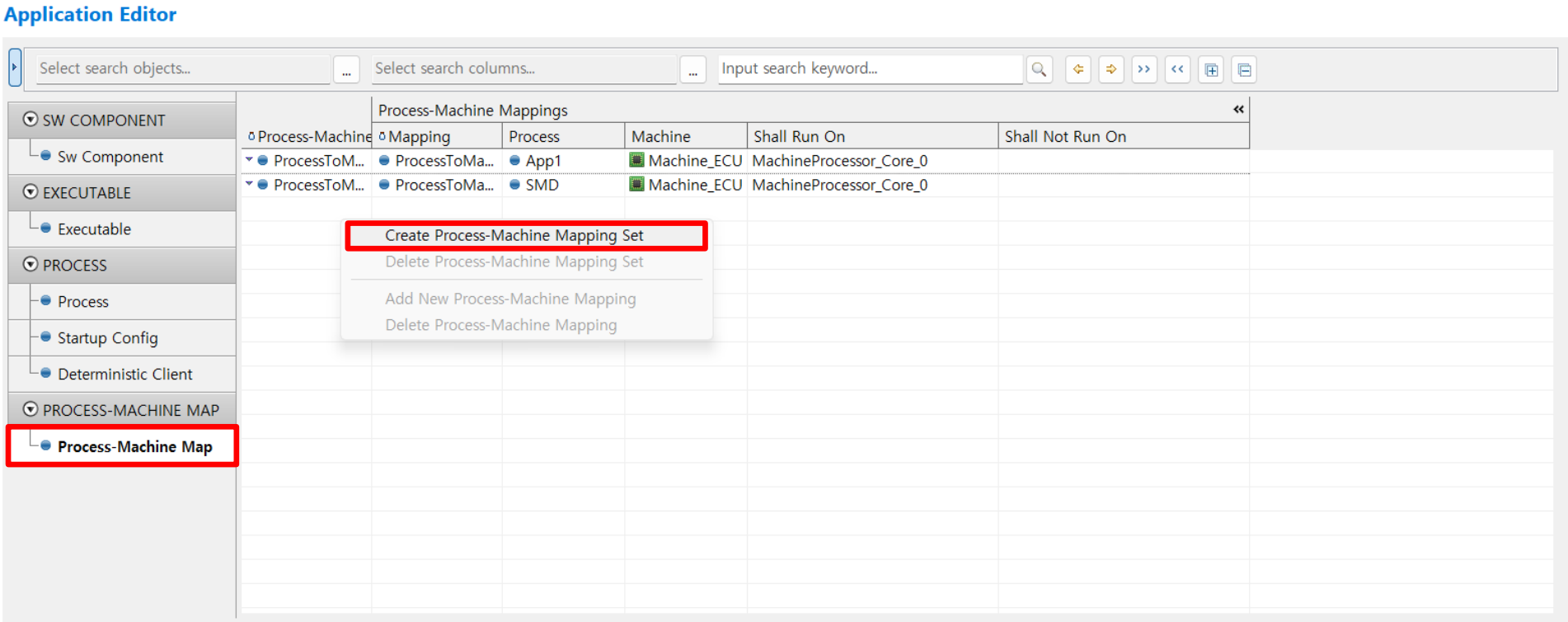
- **Startup Config 추가 연결**
  - ✓ 'App2_instance2'의 Startup Config를 생성한 'StartupConfigSet_App2' – 'StartupConfig_2'로 수정함

Application Editor



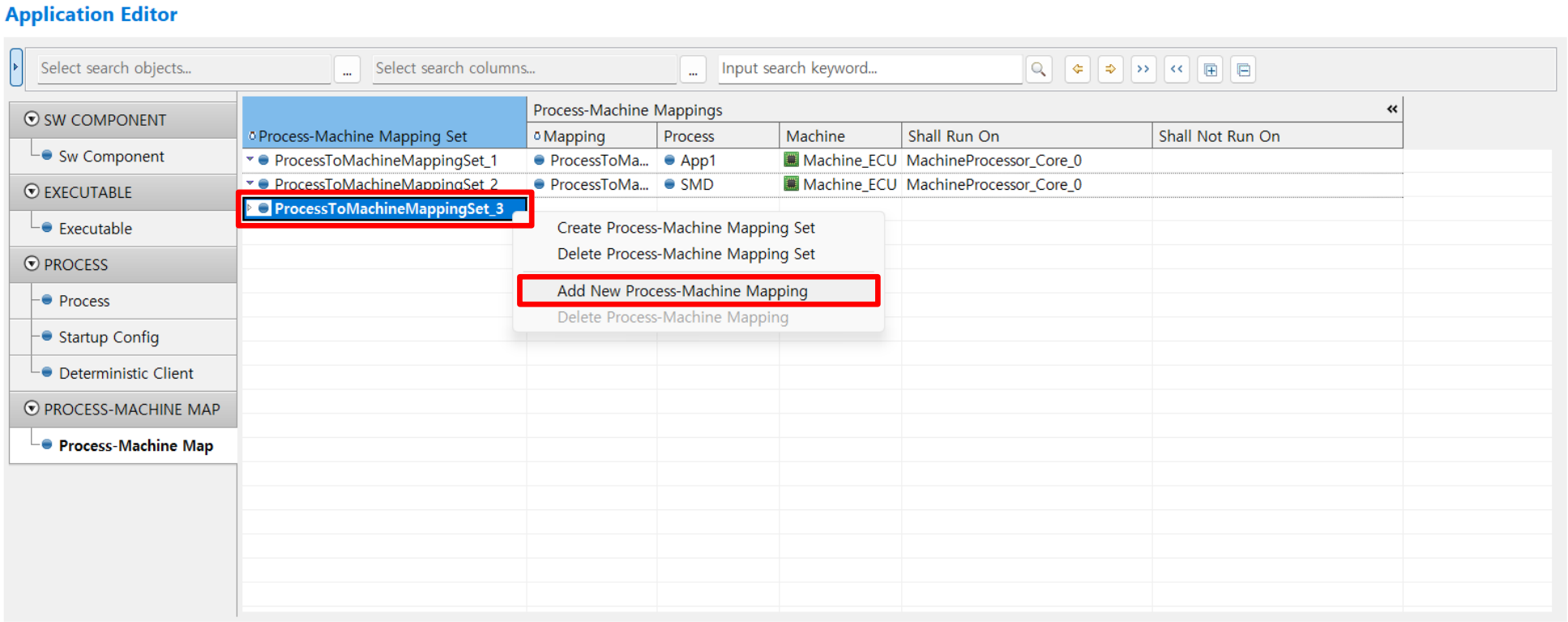
EXEC: Adaptive Applications 설정

- **Process-Machine Mapping Set 추가**
  - ✓ Process-Machine Mapping Set 추가를 위해 좌측의 'Process-Machine Map' 탭으로 이동함
  - ✓ 빈 곳에서 우클릭을 하여 'Create Process-Machine Mapping Set'을 클릭함



EXEC: Adaptive Applications 설정

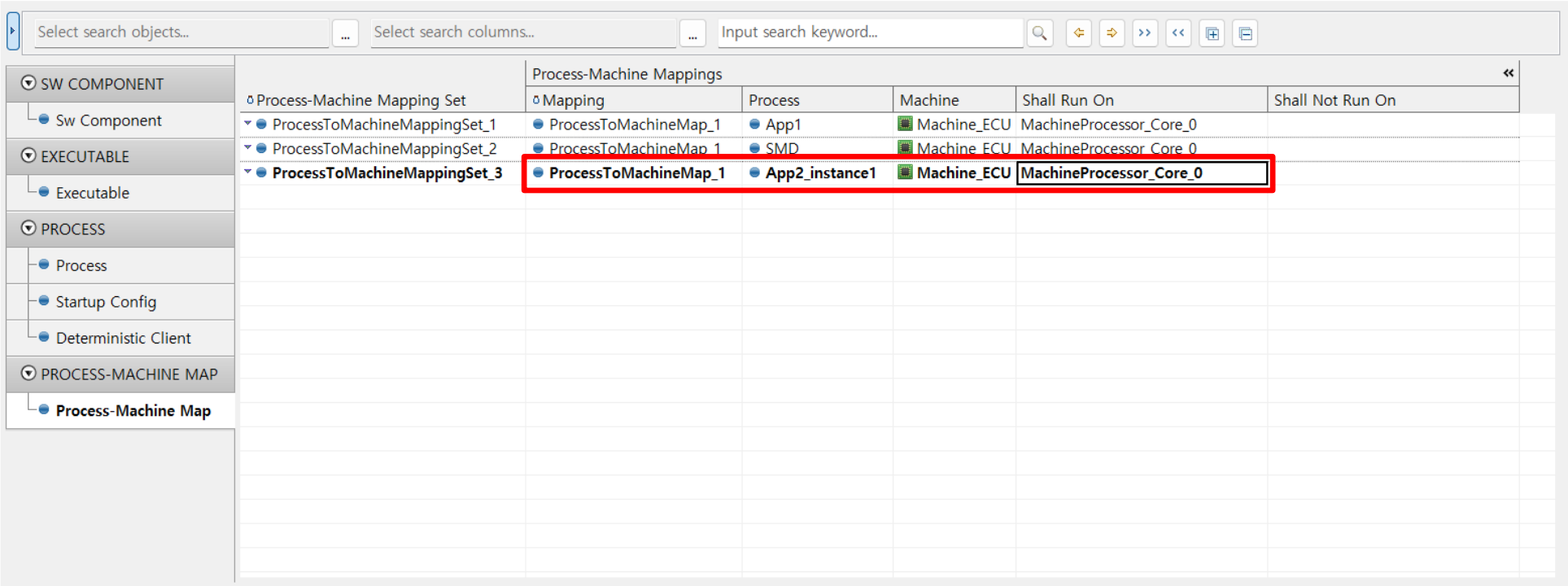
- **Process-Machine Mapping Set 추가 확인 및 Process-Machine Mapping 추가**
  - ✓ 생성한 Process-Machine Mapping Set에서 우클릭하여 'Add New Process-Machine Mapping'을 클릭함



EXEC: Adaptive Applications 설정

- **Process-Machine Mapping 추가 확인 및 설정**
  - ✓ 생성된 새로운 Process-Machine Mapping을 확인하고 다음과 같이 수정함
    - ✓ Process : ‘App2_instance1’
    - ✓ Machine : 'Machine_ECU’
    - ✓ Shall Run On : ‘MachineProcessor_Core_0’

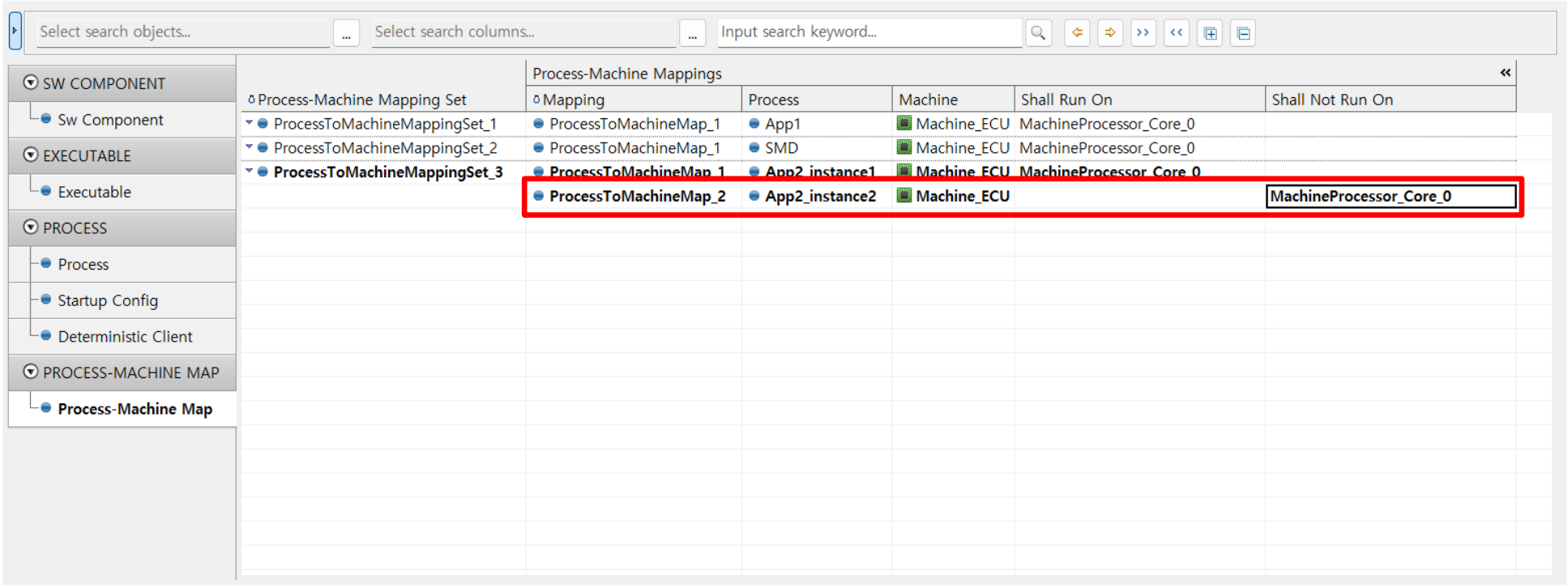
Application Editor



EXEC: Adaptive Applications 설정

- **Process-Machine Mapping 추가 설정**
  - ✓ 동일한 과정을 통해 'ProcessToMachineMappingSet_3' 하위에 하나의 Process-Machine Mapping을 추가하고 다음과 같이 수정함

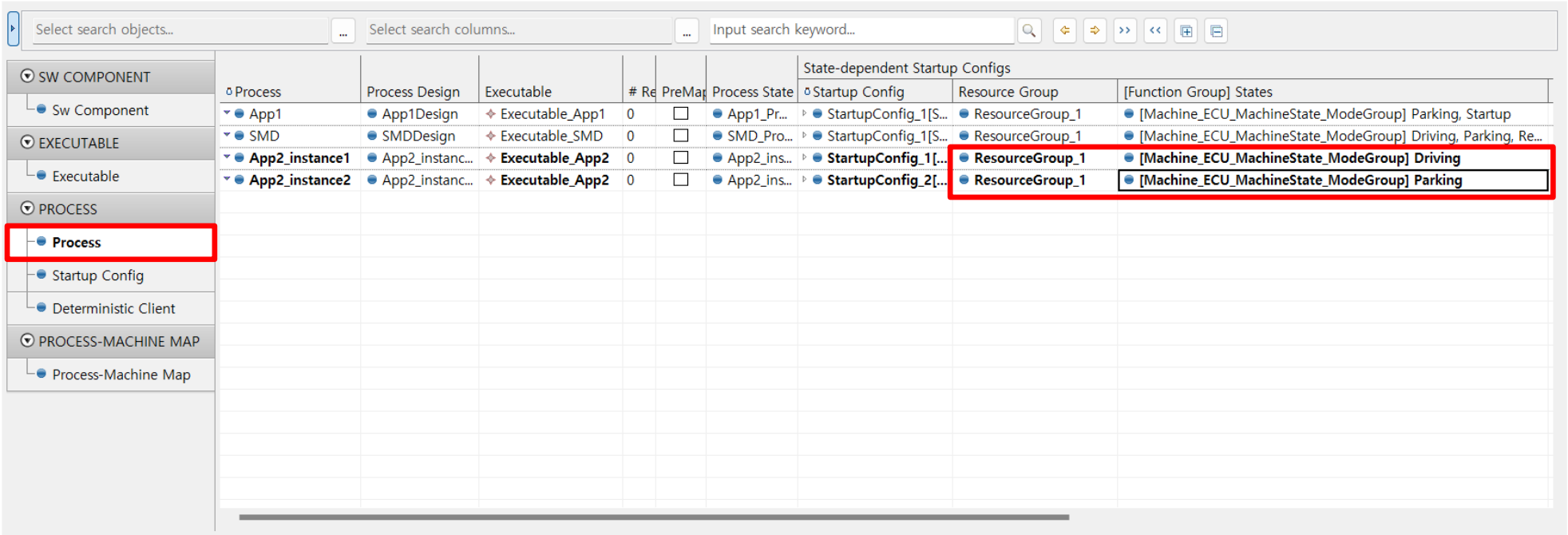
Application Editor



EXEC: Adaptive Applications 설정

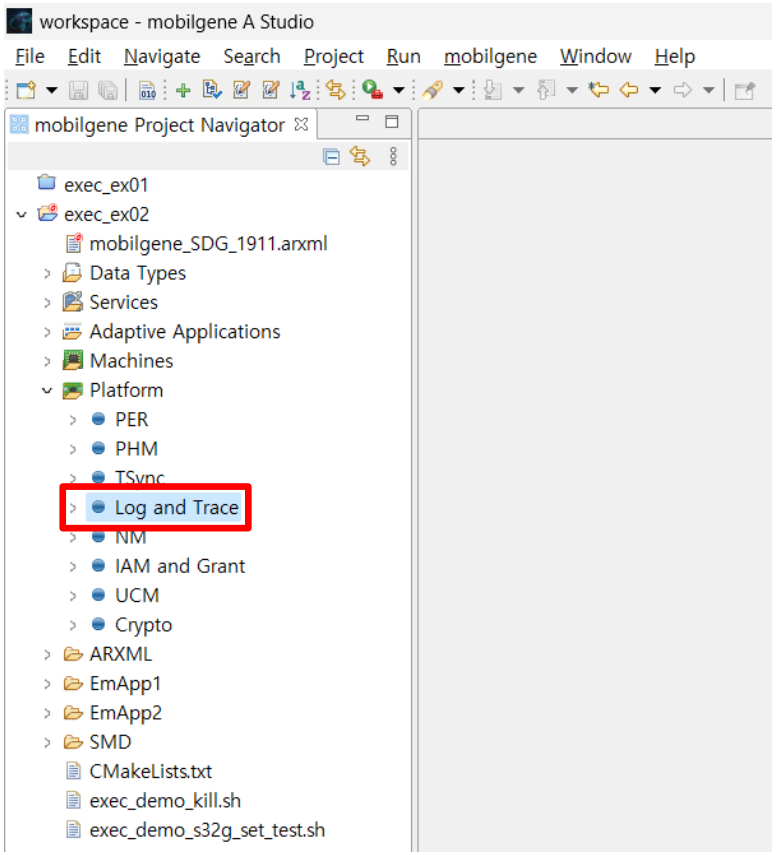
- **Machine 관련 Process 추가 설정**
  - ✓ Machine과 관련된 Process 설정을 추가하기 위해 좌측의 'Process' 탭으로 이동함
  - ✓ Process에 대해 다음과 같이 추가적으로 수정함
    - ✓ Resource Group : 'ResourceGroup_1'
    - ✓ [Function Group] States : 'Driving' (App2_instance1), 'Parking' (App2_instance2)

Application Editor



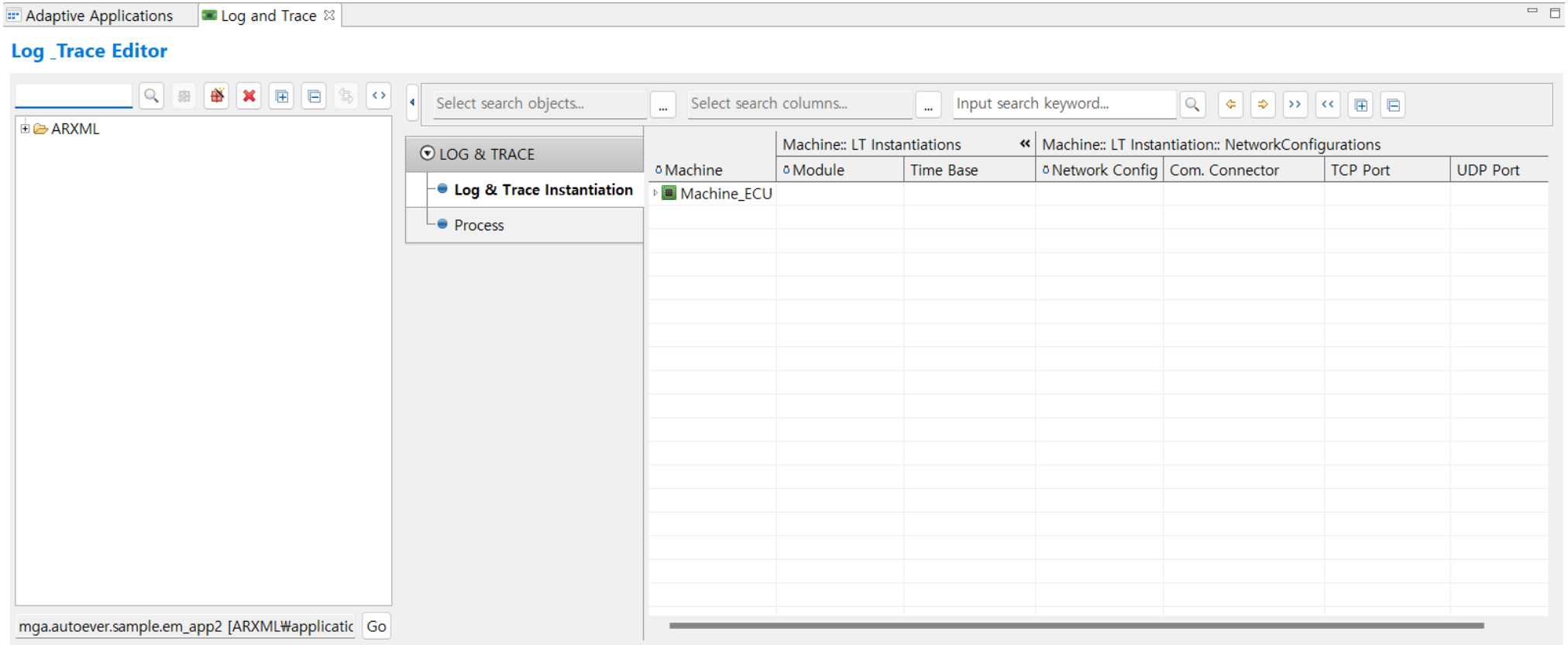
EXEC: Log 설정

- **Log and Trace Editor 활성화**
  - ✓ 왼쪽의 'mobilgene Project Navigator' 창에서 해당 Project의 'Platform' - 'Log and Trace'를 더블 클릭함



EXEC: Log 설정

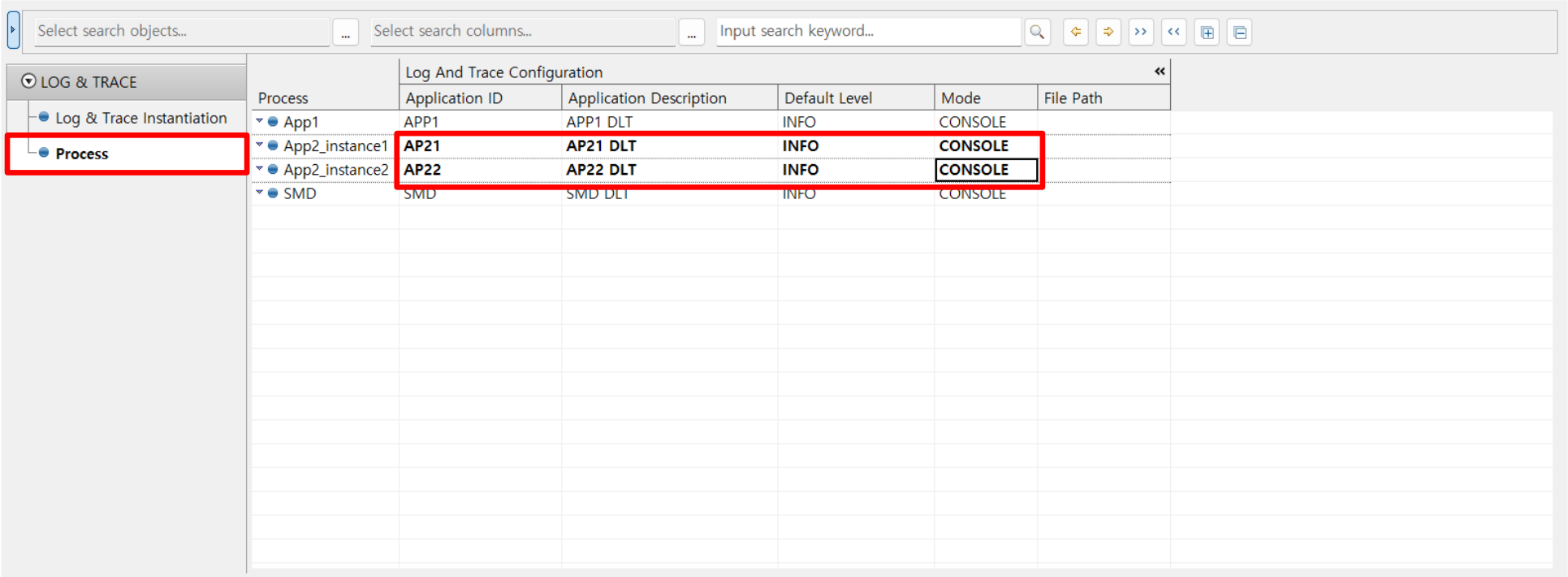
- **Log and Trace Editor 활성화 확인**
  - ✓ 활성화 된 Log and Trace Editor를 확인함



EXEC: Log 설정

- **Process Log 설정**
  - ✓ Process에 대한 Log 설정을 하기 위해 좌측의 'Process' 탭으로 이동함
  - ✓ 생성한 Process에 대한 Log 설정을 다음과 같이 수정함

Log _Trace Editor



EXEC: Code 구현

- **'em_app2.h' 파일 작성**
  - ✓ 다음과 같은 'em_app2.h' 파일 작성 ('EmApp2' - 'include' - 'em_app2.h')

Adaptive Applications | Log and Trace | em_app2.h

```
#ifndef MGA_AUTOEVER_SAMPLE_EM_APP2_H_
#define MGA_AUTOEVER_SAMPLE_EM_APP2_H_

#include <ara/log/logging.h>

namespace mga {
namespace autoever {
namespace sample {

class EmApp2 {
public:
    EmApp2(int32_t num);
    ~EmApp2();

    void Init();
    void Act();

private:
    ara::log::Logger& em_app2_logger;
    int32_t app_num;
};

}  // namespace sample
}  // namespace autoever
}  // namespace mga

#endif  // MGA_AUTOEVER_SAMPLE_EM_APP2_H_
```

EXEC: Code 구현

- **'em_app2.cpp' 파일 작성**
  - ✓ 다음과 같은 'em_app2.cpp' 파일 작성 ('EmApp2' - 'src' - 'em_app2.cpp')

Adaptive Applications | Log and Trace | em_app2.h | em_app2.cpp | main.cpp

```
/**********************************************************************************************************************
 *  INCLUDES
 *********************************************************************************************************************/
#include "em_app2.h"


/**********************************************************************************************************************
 *  VARIABLES
 *********************************************************************************************************************/


/**********************************************************************************************************************
 *  FUNCTIONS
 *********************************************************************************************************************/

namespace mga {
namespace autoever {
namespace sample {

EmApp2::EmApp2(int32_t num)
    : app_num(num),
      em_app2_logger{ara::log::CreateLogger("CTX2", "EM APP2", ara::log::LogLevel::kVerbose)} {}

EmApp2::~EmApp2() {}

void EmApp2::Init()
{
    if(app_num == 1)    em_app2_logger.LogInfo() << "APP2 Instance 1 Init Function start.";
    else                em_app2_logger.LogInfo() << "APP2 Instance 2 Init Function start.";
}

void EmApp2::Act()
{
    // em_app2_logger.LogInfo() << "APP2 Act Function start.";
}

}  // namespace sample
}  // namespace autoever
}  // namespace mga
```

EXEC: Code 구현

- **'main.cpp' 파일 작성**
  - ✓ 다음과 같은 'main.cpp' 파일 작성 ('EmApp2' - 'src' - main.cpp')

Adaptive Applications | Log and Trace | em_app2.h | em_app2.cpp | main.cpp

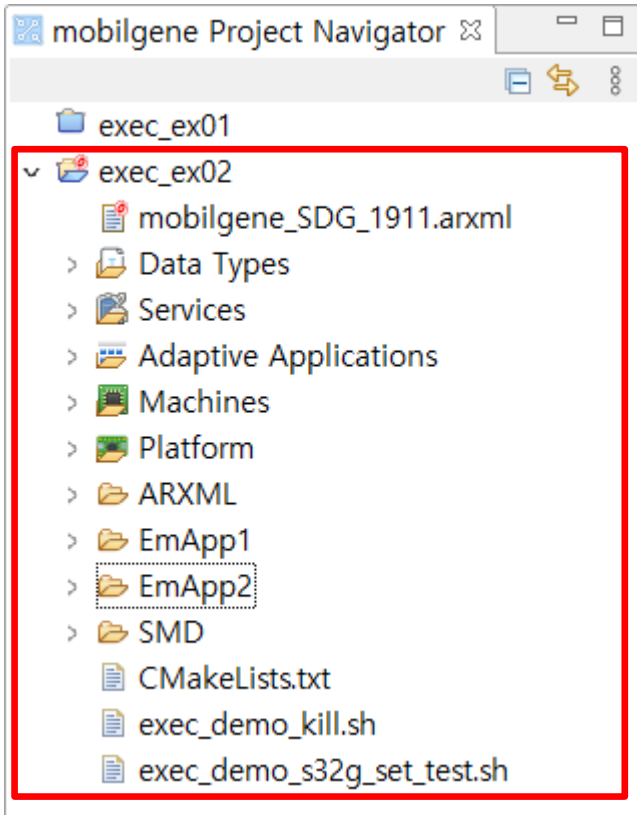
```
/**********************************************************************************************
 *  INCLUDES
 **********************************************************************************************/
#include <thread>
#include <chrono>
#include <cstdio>
#include <cstdint>
#include <csignal>

#include <ara/exec/execution_client.h>
#include <ara/log/logging.h>

#include "em_app2.h"

/**********************************************************************************************
 *  VARIABLES
 **********************************************************************************************/
ara::exec::ExecutionClient exec_client;
std::atomic<bool>exit_requested(false);
ara::log::Logger& logger{CreateLogger("CTX2", "EM APP2", ara::log::LogLevel::kVerbose)};
int32_t app_num;

/**********************************************************************************************
 *  FUNCTIONS
 **********************************************************************************************/
void ThreadAct1()
{
    logger.LogInfo() << "=================================== APP2 is running.";
    mga::autoever::sample::EmApp2 app2(app_num);
    app2.Init();

    while (1) {
        app2.Act();

        // sleep
        std::this_thread::sleep_for(std::chrono::milliseconds(400));

        if(exit_requested.load()) {
            break;
        }
    }
    logger.LogInfo() << "Application is Run exit.";
}
```

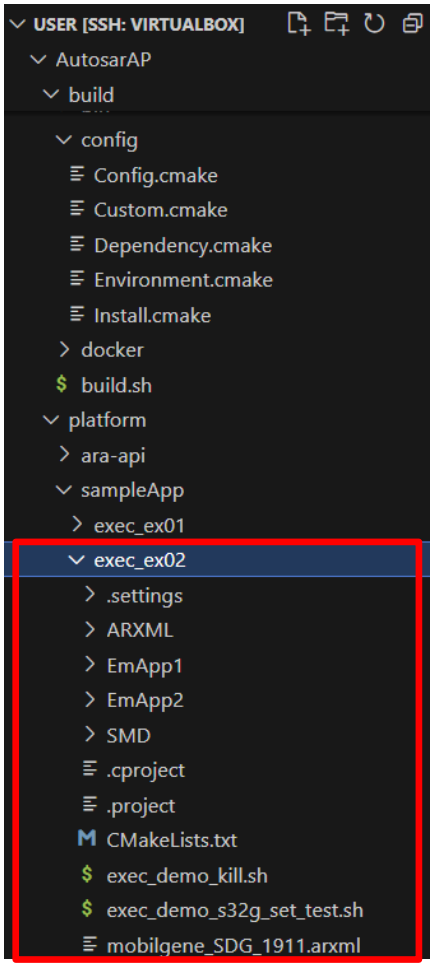
Adaptive Applications | Log and Trace | em_app2.h | em_app2.cpp | main.cpp

```
    }
    logger.LogInfo() << "Application is Run exit.";
}

void handle_signal(int _signal) {
    if(!exit_requested.load() && (_signal == SIGINT || _signal == SIGTERM)) {
        logger.LogInfo() << "received signal:" << _signal;
        exit_requested.store(true);
    }
}

int main(int argc, char* argv[])
{
    exec_client.ReportExecutionState(ara::exec::ExecutionState::kRunning);

    logger.LogInfo() << "WAIT signal:";
    signal(SIGINT, handle_signal);
    signal(SIGTERM, handle_signal);

    logger.LogInfo() << "Application will be initialized.";
    if (argc > 1) {
        if (strncmp (argv[1],"--POS=FL", strlen(argv[1])) == 0) {
            //instance1 is executed
            app_num = 1;
        }
        else {
            //instance2 is executed
            app_num = 2;
        }
    }

    std::thread act1(ThreadAct1);

    act1.join();

    logger.LogInfo() << "APP2:Application shutdown initiated.";

    exec_client.ReportExecutionState(ara::exec::ExecutionState::kTerminating);

    logger.LogInfo() << "Application shutdown is done.";

    return 0;
}
```

EXEC: Build

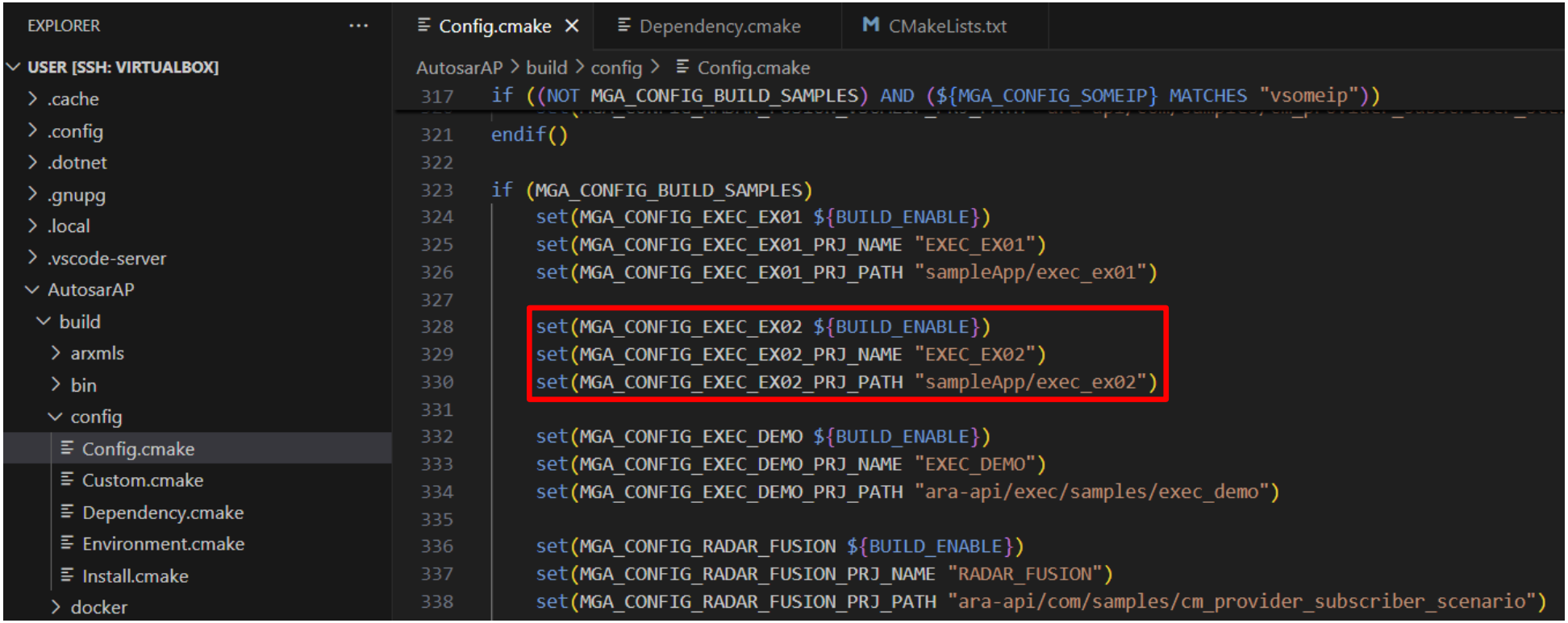
- 개발 내용 빌드 환경으로 복사
  - ✓ mobilgene A Studio에서 개발한 Adaptive Application을 빌드 환경으로 복사





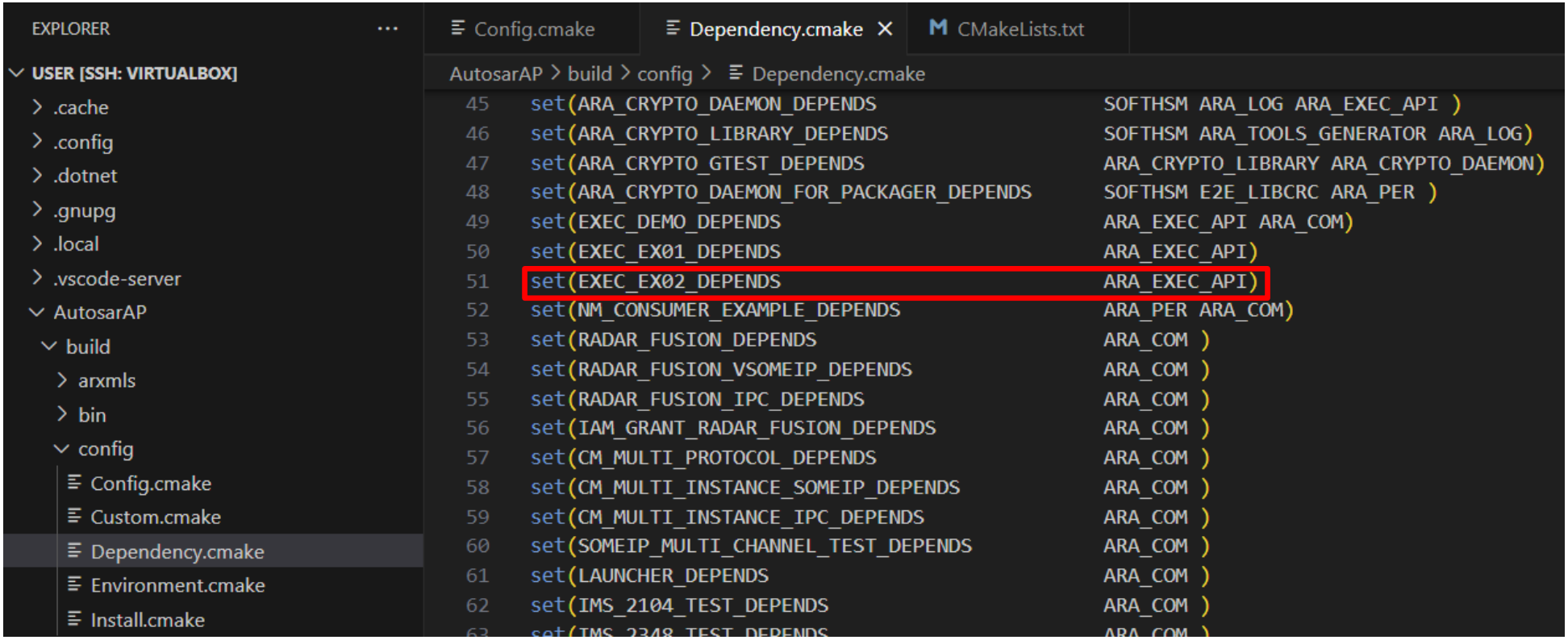
EXEC: Build

- **Adaptive Application 관련 매크로 설정 추가**
  - ✓ 'build' – 'config' - 'Config.cmake'에 추가하고자 하는 Adaptive Application 관련 매크로 설정 추가



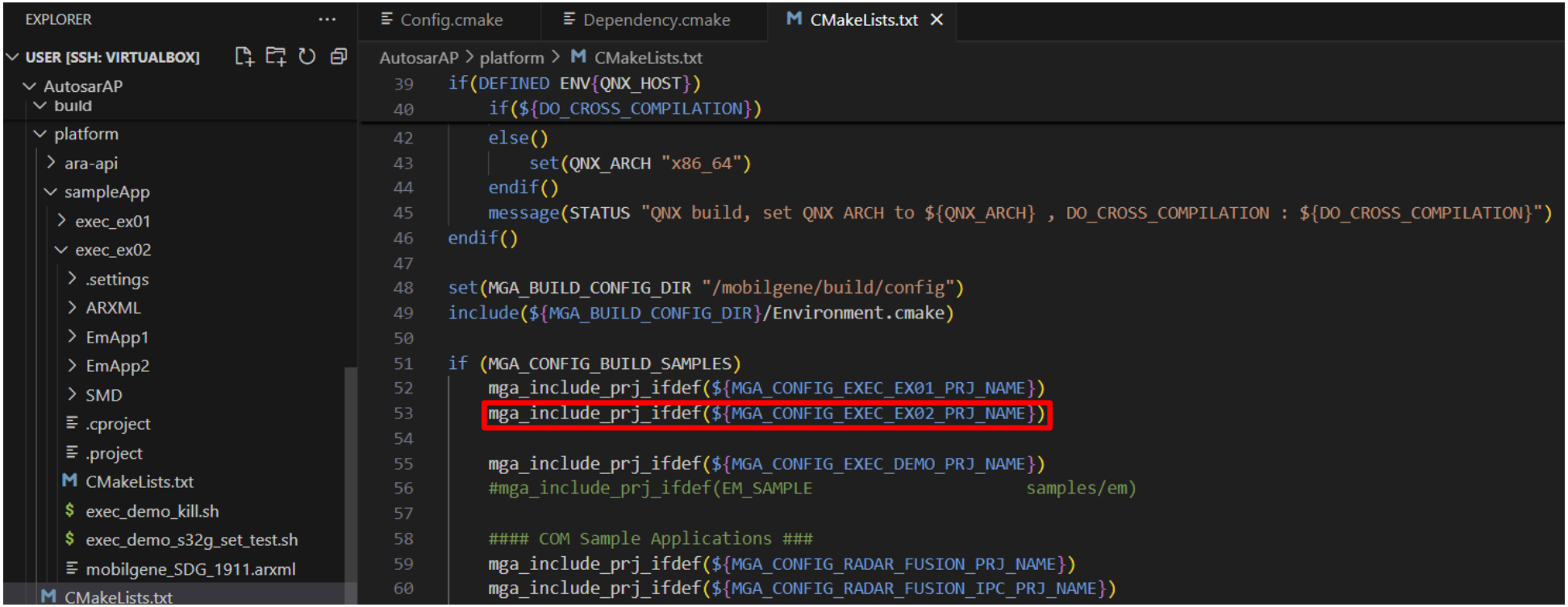
EXEC: Build

- **Adaptive Application 관련 의존성 설정 추가**
  - ✓ 'build' – 'config' - 'Dependency.cmake'에 추가하고자 하는 Adaptive Application 관련 의존성 설정 추가



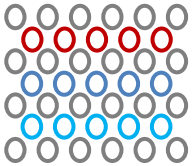
EXEC: Build

- **Adaptive Application을 Build 목록에 추가**
  - ✓ Adaptive Application을 'platform' - 'CMakeLists.txt'에 추가하여 Build 목록에 추가



Q & A

Thank you for your attention



Architecture and
Compiler
for Embedded Systems Lab.

School of Electronics Engineering, KNU

ACE Lab (junho7513@knu.ac.kr)