Unit Testing Plan
for Network Printing System

- **Test Plan**
- **Test Design Specification**
- **Test Cases Specification**

Project Team
Team T5

Date
2015-11-10

Team Information

201411316 정진호

201411294 이상혁

201411296 이선명

291411305 이찬규

Table of Contents

1 Introduction

1.1 Objectives

본 문서는 2015년 2학기 소프트웨어공학개론 수업의 Team T5가 개발한 Network Printing System (NPS)을 Unit Testing 하기 위한 계획을 다루고 있다. Team T5 가 정의한 Unit Testing을 수행하기 위하여 Testing Pass/Fail Criteria를 정의하고 이를 수행하기 위한 Test Design & Test Case 를 정의한다.

1.2 Background

2015년 2학기 소프트웨어공학개론 수업에서 개발하는 모든 NPS의 요소들은 SASD 기법을 이용하여 개발한다. 기능별로 나누어진 Unit 은 SRA와 SDS 문서에서 모두 정의되어 있다.

Network Printer System(이하 NPS)은 네트워크 프린터 시스템으로서, 사용자 또는 관리자가 네트워크 커맨드를 통해 데이터를 가상으로 출력하는 시스템이다.

Unit test는 시스템을 구성하는 최소 단위 모듈들을 대상으로 하는 test이며, 시스템의 성능을 좌우하는 요소들이 요구사항을 만족하는지를 확인할 수 있는 기본적인 Test approach이다.

1.3 Scope

이 계획 문서는 NPS의 unit test를 수행하기 위한 모든 것을 포함한다. NPS의 unit test를 수행하기 위한 지원과 절차, test approach와 technique과 필요로하는 환경 및 도구들을 정의한다. NPS의 unit test는 시스템을 구성하는 최소 단위의 모듈들을 대상으로 하며, 구현된 모듈이 요구사항을 만족하는지를 test 한다.

1.4 Project plan

1.5 Configuration management plan

1.6 References

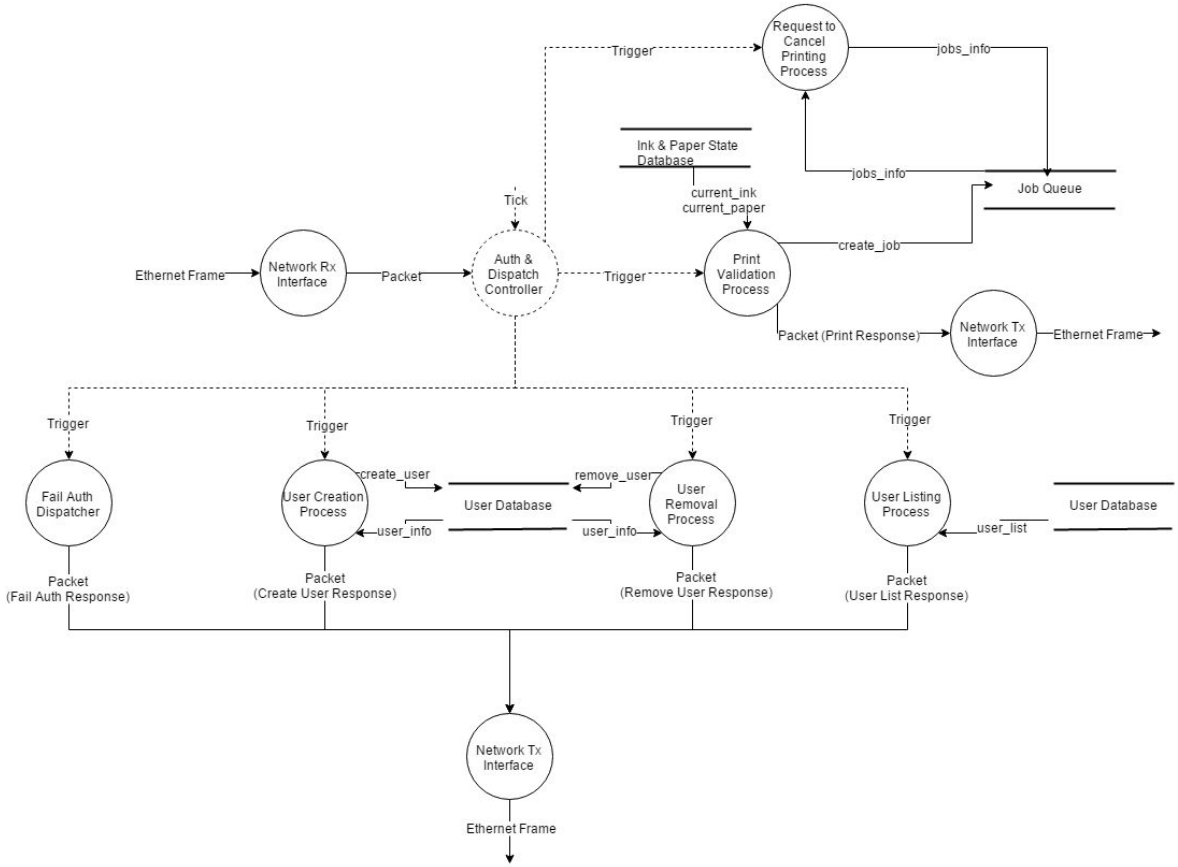
NPS SRS Ver 1.0

[2015SE_A][T5]SRA v2.0

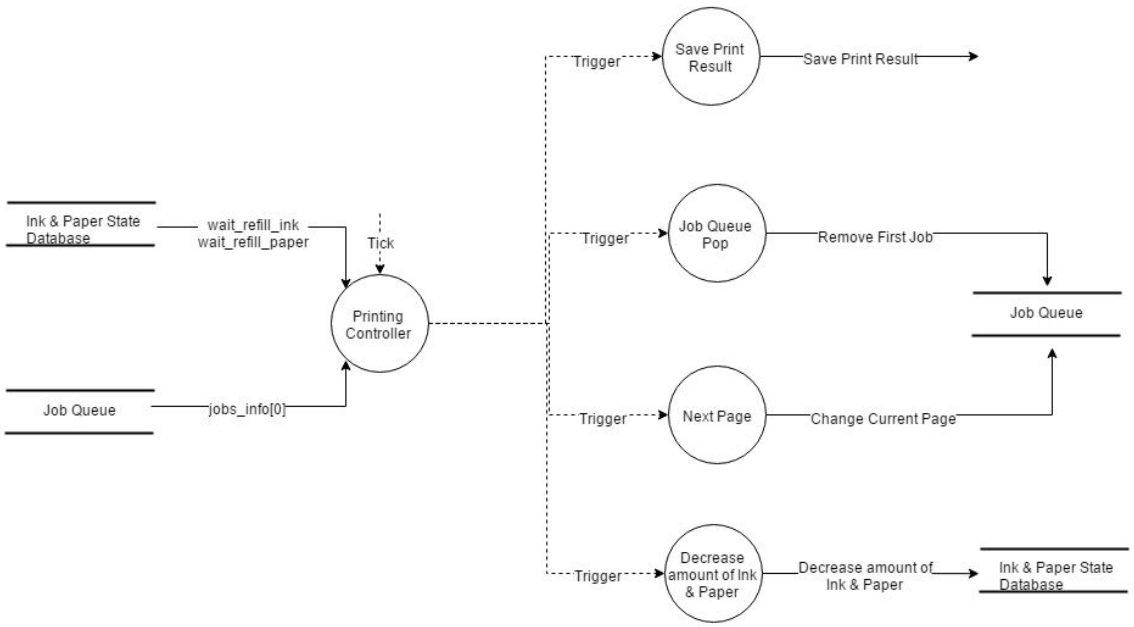
[2015SE_A][T5]SDS v1.0

2 Test items

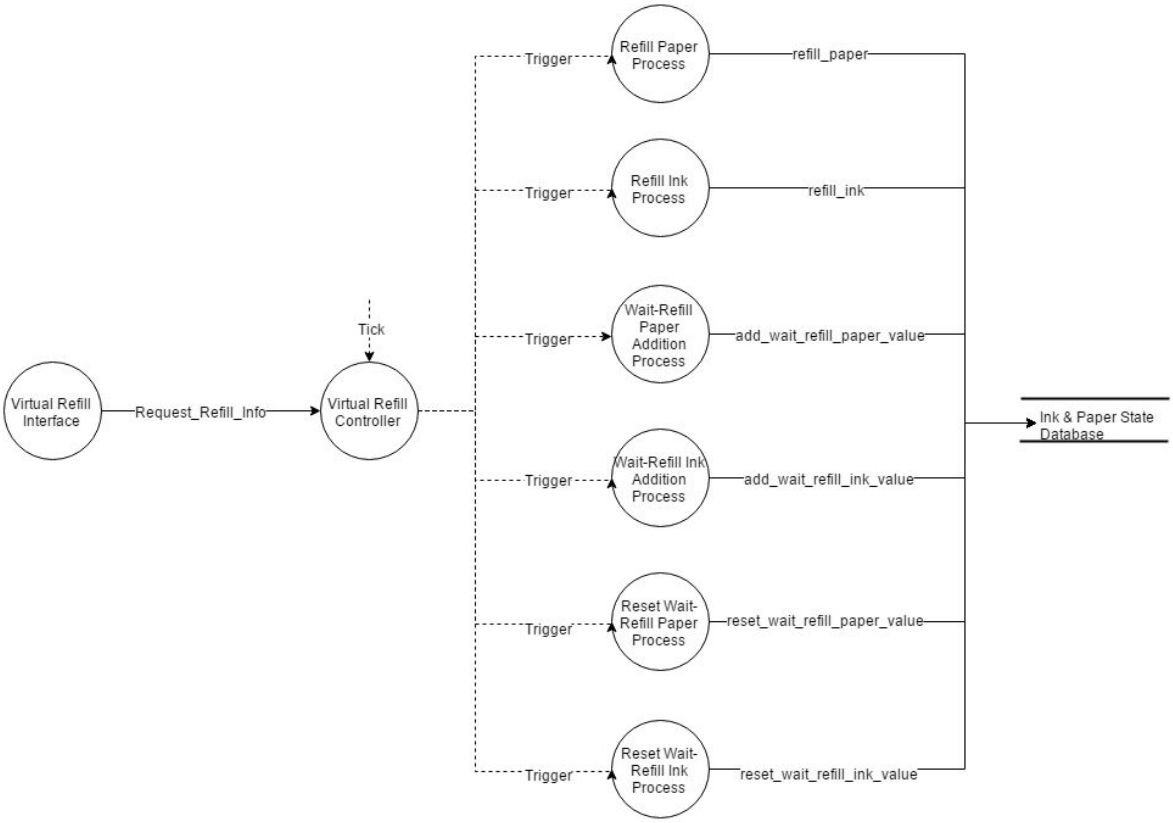
Team T5가 SASD 기법을 이용하여 개발한 NPS를 Testing한다. SA와 SD에서 분류한 각 Process/Module 별로 Testing을 수행한다. 각 그림을 참조하여 Unit을 지정하고, 지정한 Unit을 SRA에 명세된 내용과 일치하는 동작을 수행하는지 확인한다. Test item은 다음 자료들로부터 작성되었다.



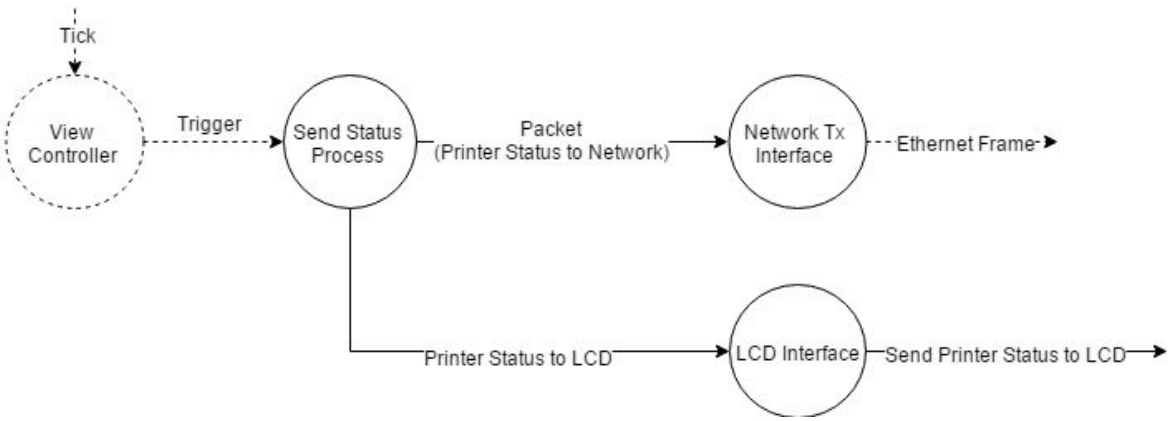
<Figure 1> DFD For Management Process



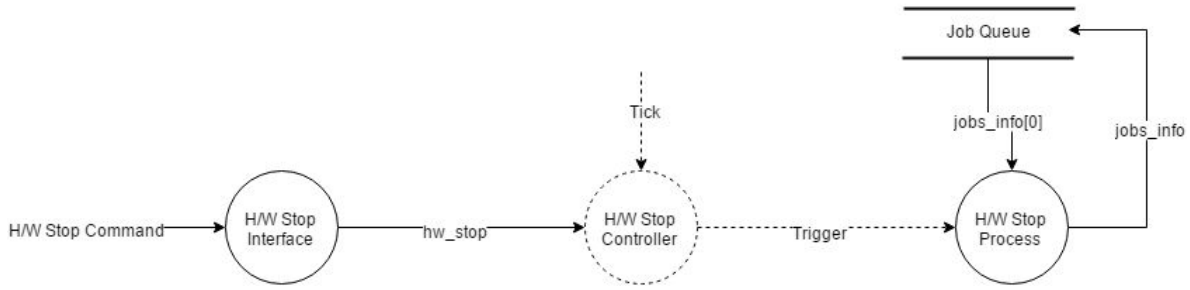
<Figure 2> DFD For Printing Process



<Figure 3> DFD for Refill Process



<Figure 4> DFD for View Process



<Figure 5> DFD for Stop Process

3 Features to be tested

Process/Module 별로 입력이 들어오면 그에 맞는 출력이 나오는지 Testing 한다.

< Table 1.1 Test 할 Process(DFD) 리스트 >

Name	**Description**
Print Validation Process	사용자가 인쇄하고싶은 데이터를 JobQueue에 등록하는 프로세스다. 만약 JobQueue가 MAX치 만큼 차있거나 잉크 또는 용지가 부족한 경우 요청을 거절한다.
User Creation Process	사용자 ID를 새로 User Database에 등록하는 프로세스다. 만약 이미 등록하려는 사용자 ID가 존재한다면 요청을 거절한다.
HW Stop Process	현재 인쇄중인 문서가 있다면 인쇄를 중지하고 삭제중으로 변경한다.
Wait Refill Ink Addition Process	잉크의 충전대기량(충전해야할 양)을 증가시키는 프로세스이다.잉크의 충전대기량과 잉크의 현 잔량의 합은 INK_MAX_VALUE = 3000을 초과할 수 없다. 충전대기량은 음수값을 받을 수 없다.
Wait Refill Paper Addition Process	용지의 충전대기량(충전해야할 양)을 증가시키는 프로세스이다. 용지의 충전대기량과 용지의 현 잔량의 합은 PAPER_MAX_VALUE = 100 을 초과할 수 없다. 충전대기량은 음수값을 받을 수 없다.
Reset Wait Refill Ink Process	충전을 다 끝내거나 잉크의 현 잔량의 합이 INK_MAX_VALUE = 3000이 되었을 때 잉크의 충전대기량을 초기화하는 프로세스이다.
Reset Wait Refill Paper Process	충전을 다 끝내거나 용지의 현 잔량의 합이 PAPER_MAX_VALUE = 100이 되었을 때 용지의 충전대기량을 초기화하는 프로세스이다.
Refill Ink Process	잉크의 충전대기량이 있을경우 Ink Paper State Database에 잉크의 잔량을 증가시키는 프로세스이다. 증가한 잉크의 잔량은 INK_MAX_VALUE = 3000을 초과할 수 없다. 프로세스 실행 한번에 증가시킬 수 있는 잉크의 최대값은 100이다.

Refill Paper Process	용지의 충전대기량이 있을경우 Ink Paper State Database에 용지의 잔량을 증가시키는 프로세스이다. 증가한 용지의 잔량은 PAPER_MAX_VALUE = 100을 초과할 수 없다. 프로세스 실행 한번에 증가시킬 수 있는 용지의 최대값은 10이다.
Job Queue Pop	JobQueue에서 인쇄중이었던 데이터의 인쇄가 모두 완료되고 난 뒤에 그 데이터를 JobQueue에서 삭제시키는 프로세스이다.
Next Page	JobQueue에서 인쇄중인 데이터의 현재 페이지(current_page)를 넘기는 역할을 하는 프로세스이다.
Decrease amount of Ink Paper	JobQueue에서 인쇄한 데이터가 사용한 잉크와 용지를 Ink Paper State Database에서 차감시키는 프로세스이다. 음수값을 차감시킬수 없다.
Send Status Process	JobQueue로 부터 JobQueue에 들어있는 인쇄해야할 데이터들(jobs_info) , Ink Paper State Database로 부터 현재 잉크 잔량, 잉크 충전 대기량, 현재 용지 잔량, 용지 충전 대기량(current_ink, wait_refill_ink, current_paper, wait_refill_paper) 을 받아서 LCD_Interface로 보낸다.
Network Rx Interface	네트워크로 데이터를 받아서 인증 모듈로 전달하는 프로세스이다.

4 Features not to be tested

이번 테스트는 Unit별로 진행하므로 전체 System과 관련된 사항은 Testing 하지 않는다. 또한 단순히 데이터를 전달하고 예외사항이 존재하지않는 프로세스들은 진행하지 않는다.

Name	**Description**
LCD Interface	View Process 에서 받은 데이터를 LCD로 전송한다.
View Controller	NPS의 데이터를 Client에게 보낼수 있도록 Send Status Process를 trigger 시키는 프로세스이다.

5 Approach

Network Printing System의 Test Code 는 Cygwin + gcc 환경에서 Team T5가 직접 작성한 Unit Test Framework 로 작성되어 실행된다. Program Source Code 와 Test Code 의 변경 및 수정 사항은 지속적으로 통합하고 Test 한다. 각 프로세스 별로 (process_name)_unit_test 내에 유닛 테스트를 위한 코드들을 Test Case에 따라 작성한다.

6 Item pass/fail criteria

Functional Test Pass/Fail Criteria : 각 Process/Module 은 요구사항을 만족해야 한다.

7 Unit test design specification

7.1 Test design specification identifier

Controller 와 그에 종속된 Process (Structured Chart 참조) 별로 구분한다.

7.1.1 Refill Process 소속

NPS.UTC.1000

7.1.2 Stop Process 소속

NPS.UTC.2000

7.1.3 Management Process 소속

NPS.UTC.3000

7.1.4 View Process 소속

NPS.UTC.4000

7.1.5 Printing Process 소속

NPS.UTC.5000

7.2 Features to be tested

7.3 Approach refinements

7.4 Test identification

Identifier	Feature	Valid/Invalid Value	Expected Output
NPS UTC 1000_000	Wait Refill Ink Addition Process	Valid. current_ink_value = 0, wait_refill_ink_value = 0 process->trigger(0)	db->wait_refill_ink_value == 0
NPS UTC 1000_001	Wait Refill Ink Addition Process	Invalid. current_ink_value = 0, wait_refill_ink_value = 0 process->trigger(-10)	db->wait_refill_ink_value == 0
NPS UTC 1000_002	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = 0 process->trigger(MAX)	db->wait_refill_ink_value == MAX
NPS UTC 1000_003	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = 0 process->trigger(5000)	db->wait_refill_ink_value == MAX
NPS UTC 1000_004	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = 0 process->trigger(10)	db->wait_refill_ink_value == 10

NPS UTC 1000_005	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = MAX / 2 process->trigger(0)	db->wait_refill_ink_value == MAX/2
NPS UTC 1000_006	Wait Refill Ink Addition Process	Invalid current_ink_value = 0, wait_refill_ink_value = MAX / 2 process->trigger(-10)	db->wait_refill_ink_value == MAX/2
NPS UTC 1000_007	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = MAX / 2 process->trigger(MAX)	db->wait_refill_ink_value == MAX
NPS UTC 1000_008	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = MAX / 2 process->trigger(5000)	db->wait_refill_ink_value == MAX
NPS UTC 1000_009	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = MAX / 2 process->trigger(10)	db->wait_refill_ink_value == MAX/2 + 10
NPS UTC 1000_010	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = MAX process->trigger(0)	db->wait_refill_ink_value == MAX
NPS UTC 1000_011	Wait Refill Ink Addition Process	Invalid current_ink_value = 0, wait_refill_ink_value = MAX process->trigger(-10)	db->wait_refill_ink_value == MAX
NPS UTC 1000_012	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = MAX process->trigger(MAX)	db->wait_refill_ink_value == MAX
NPS UTC 1000_013	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = MAX process->trigger(5000)	db->wait_refill_ink_value == MAX
NPS UTC 1000_014	Wait Refill Ink Addition Process	Valid current_ink_value = 0, wait_refill_ink_value = MAX process->trigger(10)	db->wait_refill_ink_value == MAX
NPS UTC 1000_015	Wait Refill Ink Addition Process	Valid. current_ink_value = MAX / 2, wait_refill_ink_value = 0 process->trigger(0)	db->wait_refill_ink_value == 0
NPS UTC	Wait Refill Ink Addition	Invalid.	db->wait_refill_ink_

1000_016	Process	current_ink_value = MAX / 2, wait_refill_ink_value = 0 process->trigger(-10)	value == 0
NPS UTC 1000_017	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2 , wait_refill_ink_value = 0 process->trigger(MAX)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_018	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2, wait_refill_ink_value = 0 process->trigger(5000)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_019	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2, wait_refill_ink_value = 0 process->trigger(10)	db->wait_refill_ink_value == 10
NPS UTC 1000_020	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2, wait_refill_ink_value = MAX / 2 process->trigger(0)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_021	Wait Refill Ink Addition Process	Invalid current_ink_value = MAX / 2, wait_refill_ink_value = MAX / 2 process->trigger(-10)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_022	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2, wait_refill_ink_value = MAX / 2 process->trigger(MAX)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_023	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2, wait_refill_ink_value = MAX / 2 process->trigger(5000)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_024	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2, wait_refill_ink_value = MAX / 2 process->trigger(10)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_025	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2, wait_refill_ink_value = MAX process->trigger(0)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_026	Wait Refill Ink Addition Process	Invalid current_ink_value = MAX / 2, wait_refill_ink_value = MAX process->trigger(-10)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_027	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2,	db->wait_refill_ink_value == MAX / 2

		wait_refill_ink_value = MAX process->trigger(MAX)	
NPS UTC 1000_028	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2, wait_refill_ink_value = MAX process->trigger(5000)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_029	Wait Refill Ink Addition Process	Valid current_ink_value = MAX / 2, wait_refill_ink_value = MAX process->trigger(10)	db->wait_refill_ink_value == MAX / 2
NPS UTC 1000_030	Wait Refill Ink Addition Process	Valid. current_ink_value = MAX, wait_refill_ink_value = 0 process->trigger(0)	db->wait_refill_ink_value == 0
NPS UTC 1000_031	Wait Refill Ink Addition Process	Invalid. current_ink_value = MAX, wait_refill_ink_value = 0 process->trigger(-10)	db->wait_refill_ink_value == 0
NPS UTC 1000_032	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = 0 process->trigger(MAX)	db->wait_refill_ink_value == 0
NPS UTC 1000_033	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = 0 process->trigger(5000)	db->wait_refill_ink_value == 0
NPS UTC 1000_034	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = 0 process->trigger(10)	db->wait_refill_ink_value == 0
NPS UTC 1000_035	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = MAX / 2 process->trigger(0)	db->wait_refill_ink_value == 0
NPS UTC 1000_036	Wait Refill Ink Addition Process	Invalid current_ink_value = MAX, wait_refill_ink_value = MAX / 2 process->trigger(-10)	db->wait_refill_ink_value == 0
NPS UTC 1000_037	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = MAX / 2 process->trigger(MAX);	db->wait_refill_ink_value == 0
NPS UTC 1000_038	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = MAX / 2	db->wait_refill_ink_value == 0

		process->trigger(5000)	
NPS UTC 1000_039	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = MAX / 2 process->trigger(10)	db->wait_refill_ink_value == 0
NPS UTC 1000_040	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = MAX process->trigger(0)	db->wait_refill_ink_value == 0
NPS UTC 1000_041	Wait Refill Ink Addition Process	Invalid current_ink_value = MAX, wait_refill_ink_value = MAX process->trigger(-10)	db->wait_refill_ink_value == 0
NPS UTC 1000_042	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = MAX process->trigger(MAX)	db->wait_refill_ink_value == 0
NPS UTC 1000_043	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = MAX process->trigger(5000)	db->wait_refill_ink_value == 0
NPS UTC 1000_044	Wait Refill Ink Addition Process	Valid current_ink_value = MAX, wait_refill_ink_value = MAX process->trigger(10)	db->wait_refill_ink_value == 0
NPS UTC 1000_045	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = 0 process->trigger(0)	db->wait_refill_paper_value == 0
NPS UTC 1000_046	Wait Refill Paper Addition Process	InValid current_paper_value = 0, wait_refill_paper_value = 0 process->trigger(-10)	db->wait_refill_paper_value == 0
NPS UTC 1000_047	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = 0 process->trigger(MAX)	db->wait_refill_paper_value == MAX
NPS UTC 1000_048	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = 0 process->trigger(5000)	db->wait_refill_paper_value == MAX
NPS UTC 1000_049	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = 0 process->trigger(10)	db->wait_refill_paper_value == 10

NPS UTC 1000_050	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = MAX/2 process->trigger(0)	db->wait_refill_paper_value == MAX / 2
NPS UTC 1000_051	Wait Refill Paper Addition Process	InValid current_paper_value = 0, wait_refill_paper_value = MAX/2 process->trigger(-10)	db->wait_refill_paper_value == MAX / 2
NPS UTC 1000_052	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = MAX/2 process->trigger(MAX)	db->wait_refill_paper_value == MAX
NPS UTC 1000_053	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = MAX/2 process->trigger(5000)	db->wait_refill_paper_value == MAX
NPS UTC 1000_054	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = MAX/2 process->trigger(10)	db->wait_refill_paper_value == MAX
NPS UTC 1000_055	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = MAX process->trigger(0)	db->wait_refill_paper_value == MAX
NPS UTC 1000_056	Wait Refill Paper Addition Process	InValid current_paper_value = 0, wait_refill_paper_value = MAX process->trigger(-10)	db->wait_refill_paper_value == MAX
NPS UTC 1000_057	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = MAX process->trigger(MAX)	db->wait_refill_paper_value == MAX
NPS UTC 1000_058	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = MAX process->trigger(5000)	db->wait_refill_paper_value == MAX
NPS UTC 1000_059	Wait Refill Paper Addition Process	Valid current_paper_value = 0, wait_refill_paper_value = MAX process->trigger(10)	db->wait_refill_paper_value == MAX
NPS UTC 1000_060	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = 0 process->trigger(0)	db->wait_refill_paper_value == 0
NPS UTC	Wait Refill Paper	InValid	db->wait_refill_pap

1000_061	Addition Process	current_paper_value = MAX/2, wait_refill_paper_value = 0 process->trigger(-10)	er_value == 0
NPS UTC 1000_062	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = 0 process->trigger(MAX)	db->wait_refill_pap er_value == MAX / 2
NPS UTC 1000_063	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = 0 process->trigger(5000)	db->wait_refill_pap er_value == MAX / 2
NPS UTC 1000_064	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = 0 process->trigger(10)	db->wait_refill_pap er_value == 10
NPS UTC 1000_065	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = MAX/2 process->trigger(0)	db->wait_refill_pap er_value == MAX / 2
NPS UTC 1000_066	Wait Refill Paper Addition Process	InValid current_paper_value = MAX/2, wait_refill_paper_value = MAX/2 process->trigger(-10)	db->wait_refill_pap er_value == MAX /2
NPS UTC 1000_067	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = MAX/2 process->trigger(MAX)	db->wait_refill_pap er_value == MAX / 2
NPS UTC 1000_068	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = MAX/2 process->trigger(5000)	db->wait_refill_pap er_value == MAX / 2
NPS UTC 1000_069	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = MAX/2 process->trigger(10)	db->wait_refill_pap er_value == MAX / 2
NPS UTC 1000_070	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = MAX process->trigger(0)	db->wait_refill_pap er_value == MAX /2
NPS UTC 1000_071	Wait Refill Paper Addition Process	InValid current_paper_value = MAX/2, wait_refill_paper_value = MAX process->trigger(-10)	db->wait_refill_pap er_value == MAX / 2
NPS UTC 1000_072	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2,	db->wait_refill_pap er_value == MAX /

		wait_refill_paper_value = MAX process->trigger(MAX)	2
NPS UTC 1000_073	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = MAX process->trigger(5000)	db->wait_refill_paper_value == MAX / 2
NPS UTC 1000_074	Wait Refill Paper Addition Process	Valid current_paper_value = MAX/2, wait_refill_paper_value = MAX process->trigger(10)	db->wait_refill_paper_value == MAX / 2
NPS UTC 1000_075	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = 0 process->trigger(0)	db->wait_refill_paper_value == 0
NPS UTC 1000_076	Wait Refill Paper Addition Process	InValid current_paper_value = MAX, wait_refill_paper_value = 0 process->trigger(-10)	db->wait_refill_paper_value == 0
NPS UTC 1000_077	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = 0 process->trigger(MAX)	db->wait_refill_paper_value == 0
NPS UTC 1000_078	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = 0 process->trigger(5000)	db->wait_refill_paper_value == 0
NPS UTC 1000_079	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = 0 process->trigger(10)	db->wait_refill_paper_value == 0
NPS UTC 1000_080	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = MAX/2 process->trigger(0)	db->wait_refill_paper_value == 0
NPS UTC 1000_081	Wait Refill Paper Addition Process	InValid current_paper_value = MAX, wait_refill_paper_value = MAX/2 process->trigger(-10)	db->wait_refill_paper_value == 0
NPS UTC 1000_082	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = MAX/2 process->trigger(MAX)	db->wait_refill_paper_value == 0
NPS UTC 1000_083	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = MAX/2	db->wait_refill_paper_value == 0

		process->trigger(5000)	
NPS UTC 1000_084	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = MAX/2 process->trigger(10)	db->wait_refill_paper_value == 0
NPS UTC 1000_085	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = MAX process->trigger(0)	db->wait_refill_paper_value == 0
NPS UTC 1000_086	Wait Refill Paper Addition Process	InValid current_paper_value = MAX, wait_refill_paper_value = MAX process->trigger(-10)	db->wait_refill_paper_value == 0
NPS UTC 1000_087	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = MAX process->trigger(MAX)	db->wait_refill_paper_value == 0
NPS UTC 1000_088	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = MAX process->trigger(5000)	db->wait_refill_paper_value == 0
NPS UTC 1000_089	Wait Refill Paper Addition Process	Valid current_paper_value = MAX, wait_refill_paper_value = MAX process->trigger(10)	db->wait_refill_paper_value == 0
NPS UTC 1000_090	Reset Wait Refill Ink Process	Valid wait_refill_ink_value = 100 process->trigger()	db->wait_refill_ink_value == 0
NPS UTC 1000_091	Reset Wait Refill Ink Process	Valid wait_refill_ink_value = 0 process->trigger()	db->wait_refill_ink_value == 0
NPS UTC 1000_092	Reset Wait Refill Paper Process	Valid wait_refill_paper_value = 100 process->trigger()	db->wait_refill_paper_value == 0
NPS UTC 1000_093	Reset Wait Refill Paper Process	Valid wait_refill_paper_value = 0 process->trigger()	db->wait_refill_paper_value == 0
NPS UTC 1000_094	Refill Ink Process	Valid current_ink_value = 0 wait_refill_ink_value = 0 process->trigger()	db->current_ink_value == 0
NPS UTC 1000_095	Refill Ink Process	Invalid current_ink_value = 0 wait_refill_ink_value = -50	db->current_ink_value == 0

		process->trigger()	
NPS UTC 1000_096	Refill Ink Process	Valid current_ink_value = 0 wait_refill_ink_value = 100 process->trigger()	db->current_ink_value == 100
NPS UTC 1000_097	Refill Ink Process	Valid current_ink_value = 0 wait_refill_ink_value = 1000 process->trigger()	db->current_ink_value == 100
NPS UTC 1000_098	Refill Ink Process	Valid current_ink_value = 0 wait_refill_ink_value = 50 process->trigger()	db->current_ink_value == 50
NPS UTC 1000_099	Refill Ink Process	Valid current_ink_value = MAX/2 wait_refill_ink_value = 0 process->trigger()	db->current_ink_value == MAX/2
NPS UTC 1000_100	Refill Ink Process	Invalid current_ink_value = MAX/2 wait_refill_ink_value = -50 process->trigger()	db->current_ink_value == MAX/2
NPS UTC 1000_101	Refill Ink Process	Valid current_ink_value = MAX/2 wait_refill_ink_value = 100 process->trigger()	db->current_ink_value == MAX/2 + 100
NPS UTC 1000_102	Refill Ink Process	Valid current_ink_value = MAX/2 wait_refill_ink_value = 1000 process->trigger()	db->current_ink_value == MAX/2 + 100
NPS UTC 1000_103	Refill Ink Process	Valid current_ink_value = MAX/2 wait_refill_ink_value = 50 process->trigger()	db->current_ink_value == MAX/2 + 50
NPS UTC 1000_104	Refill Ink Process	Valid current_ink_value = MAX wait_refill_ink_value = 0 process->trigger()	db->current_ink_value == MAX
NPS UTC 1000_105	Refill Ink Process	Invalid current_ink_value = MAX wait_refill_ink_value = -50 process->trigger()	db->current_ink_value == MAX
NPS UTC 1000_106	Refill Ink Process	Valid current_ink_value = MAX wait_refill_ink_value = 100 process->trigger()	db->current_ink_value == MAX

NPS UTC 1000_107	Refill Ink Process	Valid current_ink_value = MAX wait_refill_ink_value = 1000 process->trigger()	db->current_ink_value == MAX
NPS UTC 1000_108	Refill Ink Process	Valid current_ink_value = MAX wait_refill_ink_value = 50 process->trigger()	db->current_ink_value == MAX
NPS UTC 1000_109	Refill Paper Process	Valid current_paper_value = 0 wait_refill_paper_value = 0 process->trigger()	db->current_paper_value == 0
NPS UTC 1000_110	Refill Paper Process	Invalid current_paper_value = 0 wait_refill_paper_value = -5 process->trigger()	db->current_paper_value == 0
NPS UTC 1000_111	Refill Paper Process	Valid current_paper_value = 0 wait_refill_paper_value = 10 process->trigger()	db->current_paper_value == 10
NPS UTC 1000_112	Refill Paper Process	Valid current_paper_value = 0 wait_refill_paper_value = 1000 process->trigger()	db->current_paper_value == 10
NPS UTC 1000_113	Refill Paper Process	Valid current_paper_value = 0 wait_refill_paper_value = 5 process->trigger()	db->current_paper_value == 5
NPS UTC 1000_114	Refill Paper Process	Valid current_paper_value = MAX/2 wait_refill_paper_value = 0 process->trigger()	db->current_paper_value == MAX/2
NPS UTC 1000_115	Refill Paper Process	Invalid current_paper_value = MAX/2 wait_refill_paper_value = -5 process->trigger()	db->current_paper_value == MAX/2
NPS UTC 1000_116	Refill Paper Process	Valid current_paper_value = MAX/2 wait_refill_paper_value = 10 process->trigger()	db->current_paper_value == MAX/2 + 10
NPS UTC 1000_117	Refill Paper Process	Valid current_paper_value = MAX/2 wait_refill_paper_value = 1000 process->trigger()	db->current_paper_value == MAX / 2 + 10
NPS UTC	Refill Paper Process	Valid	db->current_paper

1000_118		current_paper_value = MAX/2 wait_refill_paper_value = 5 process->trigger()	_value == MAX / 2 + 5
NPS UT C 1000_119	Refill Paper Process	Valid current_paper_value = MAX wait_refill_paper_value = 0 process->trigger()	db->current_paper _value == MAX
NPS UTC 1000_120	Refill Paper Process	Invalid current_paper_value = MAX wait_refill_paper_value = -5 process->trigger()	db->current_paper _value == MAX
NPS UTC 1000_121	Refill Paper Process	Valid current_paper_value = MAX wait_refill_paper_value = 10 process->trigger()	db->current_paper _value == MAX
NPS UTC 1000_122	Refill Paper Process	Valid current_paper_value = MAX wait_refill_paper_value = 1000 process->trigger()	db->current_paper _value == MAX
NPS UTC 1000_123	Refill Paper Process	Valid current_paper_value = MAX wait_refill_paper_value = 5 process->trigger()	db->current_paper _value == MAX

NPS UTC 1000_124	Virtual Refill Interface	/*정상 작동 테스트*/
		Valid
		struct RequestRefillInfo info;
		int client;
		test_buf[] = {0,0,0,0,10};
		
		struct sockaddr_in server;
		bzero(&server, sizeof(server));
		server.sin_addr.s_addr = inet_addr("127.0.0.1");
		s erver.sin_family = AF_INET;
		server.sin_port = htons(6500);
		
		client = create_test_client();
		
		sendto(client, test_buf, 8, 0, (struct sockaddr *)&server, sizeof(server));
		
		info = interface->get_request_info();
			info.type == 0 && info.value == 10

NPS UTC 1000_125	Virtual Refill Controller	//IDLE => Wait_Refill_Ink Valid current_ink_value = 0 wait_ink_value = 0	*(db->wait_refill_ink _value) == 2570

		char test_buf[] = {0,0,0,10,10};
		sendto(client, test_buf, 8, 0,
		(struct sockaddr *)&server,
		sizeof(server));
		controller->tick();	

NPS UTC
1000_126
	Virtual Refill Controller	//Wait_Refill_Ink => Reill_Ink =>
		Wait_refill_Ink
		Valid
		
		for(i = 0; i < 21; i++)
		{
		controller->tick();
		}
			*(db->wait_refill_ink
			_value) == 2370
			&&
			*(db->current_ink_v
			alue) == 200

NPS UTC
1000_127
	Virtual Refill Controller	//Wait_Refill_Ink => ... => IDLE
		Valid
		
		while(*(db->wait_refill_ink_value))
		{
		controller->tick();
		}
			*(db->wait_refill_ink
			_value) == 0 &&
			*(db->current_ink_v
			alue) == 2570

NPS UTC
1000_128
	Virtual Refill Controller	//IDLE => Wait_Refill_Paper
		Valid
		
		current_paper_value = 0
		wait_paper_value = 0
		
		char test_buf[] = {1,0,0,0,60};
		sendto(client, test_buf, 8, 0,
		(struct sockaddr *)&server,
		sizeof(server));
		controller->tick();
			*(db->wait_refill_pa
			per_value) == 60)

NPS UTC
1000_129
	Virtual Refill Controller	//Wait_Refill_Ink => Reill_Ink =>
		Wait_refill_Ink
		Valid
		
		for(int i = 0; i < 21; i++)
		{
		controller->tick();
		}
			*(db->wait_refill_pa
			per_value) == 40
			&&
			*(db->current_pape
			r_value) == 20

NPS UTC
1000_130
	Virtual Refill Controller	//Wait_Refill_Paper => ... =>
		IDLE
		Valid
		
		while(*(db->wait_refill_paper_val
		ue))
		{
		controller->tick();
		}
			*(db->wait_refill_pa
			per_value) == 0 &&
			*(db->current_pape
			r_value) == 60

NPS UTC 2000_000	H/W Stop Process	*JobQueue가 비어있는 경우 */ Valid	// 프로세스가 작동하지 않아야

		/ JobQueue->jobs_info[0].job_id = -1 // job_id = -1일 경우 Job_info가 비어있음 JobQueue->jobs_info[0].state = 0 process->trigger()	한다. test_jobs_info.state == 0

NPS UTC 2000_001	H/W Stop Process	/*인쇄중이거나 대기중인경우*/
		Valid
		JobQueue->jobs_info[0].job_id = 1;
		JobQueue->jobs_info[0].state = 0;
		process->trigger()
			//정상 작동
			test_jobs_info.state == 1

NPS UTC 2000_002	H/W Stop Process	/*현재 삭제중인 경우*/ Valid JobQueue->jobs_info[0].job_id = 1 JobQueue->jobs_info[0].state = 1 process->trigger()	test_jobs_info.state == 1
NPS UTC 2000_003	H/W Stop Interface	//hw_stop 함수 테스트 Valid test_buf[0] = 1 sendto(csock, test_buf, 1, 0, (struct sockaddr *)&sa, sizeof(sa)) output = interface->hw_stop()	output == 1
NPS UTC 2000_004	H/W stop Controller	Valid test_buf[1] = {1} sendto(csock, test_buf, 1, 0, (struct sockaddr *)&server, sizeof(server)); process->tick()	test_jobs_info.state == 1

NPS UTC 3000_000	auth_and_dispatch_controller	// 현재 상태가 대기이며 인쇄 요청 패킷이옴 (권한 실패)
		// 패킷 데이터는 프로토콜 참고
		current_state = ready
		packet->length = ~;
		packet->data = /* 0, 0, test, test, content:test로 테스트 */
		
		current_state == fail_auth ?	

NPS UTC 3000_001	auth_and_dispatch_controller	// 관리자 명령이 온 경우
		// 그러나 일반사용자가 보냄
		test_packet_data[0] = 2;
		user_id = “user0”
		user_pw = “pass0”
			execute FailAuthDispatcher trigger

NPS UTC 3001_002	auth_and_dispatch_controller	test_packet_data[0] = 0;
		user_id = “admin”
		user_pw= “admin”
		// 이하 패킷 요청 유저는 동일
			execute PrintValidation Process

NPS UTC 3000_003	auth_and_dispatch_controller	test_packet_data[0] = 1;	execute RequestToCancelPrintingProcess trigger
NPS UTC 3000_004	auth_and_dispatch_controller	test_packet_data[0] = 2;	execute UserCreationProcess trigger
NPS UTC 3000_005	auth_and_dispatch_controller	test_packet_data[0] = 3;	execute UserRemovalProcess trigger
NPS UTC 3000_006	auth_and_dispatch_controller	test_packet_data[0] = 4;	execute UserListingProcess trigger
NPS UTC 3000_007	fail_auth_dispatcher	// send_packet을 제대로 호출하는가? execute trigger	trigger sends packet
NPS UTC 3000_008	print_validation_process	test_jobs_info[JOB_MAX_SIZE-2].job_id = -1 _get_empty_job_index() == JOB_MAX_SIZE - 2	
NPS UTC 3000_009	print_validation_process	// test for get_require_ink char *buf = "Cowabunga Cowabunga Cowabunga \n";	_get_require_ink(buf, 27) == 27
NPS UTC 3000_010	print_validation_process	// test for get_require_paper char *buf = "a\na\na\na\na\na\na\na\na\na\na\na\na\na\na\na\na\na\na\na\n";	_get_require_paper(buf, 40) == 2

NPS UTC 3000_011	print_validation_process	// Test for get_job_queue_require_ink
		res = get_job_queue_require_ink
			res == 0

NPS UTC 3000_012	print_validation_process	// Test for get_job_queue_require_paper
		res = get_job_queue_require_paper
			res == 0

NPS UTC 3000_013	print_validation_process	// Test for trigger
		trigger();
			res_type == 0

NPS UTC 3000_014	print_validation_process	// Test for trigger
		// 잉크 부족
		current_ink_value = 0;
		trigger();
			res_type == 2

NPS UTC 3000_015	print_validation_process	// Test for trigger
		// 용지 부족
		current_paper_value = 0;
		trigger();
			res_type == 3

NPS UTC 3000_016	rquest_to_cancel_printing_process	// 첫번째 작업취소
		test_packet.data = "\0\0\0\0\0user0\0\0\0\0\0\0\0pass0\0\0\0\0\0\0\0\0\0\0\1",
		process->trigger(&test_packet, 0);
			test_jobs_info[0].state == REMOVING

NPS UTC 3000_017	rquest_to_cancel_printing_process	// 두번째 작업 취소
		test_packet.data = "\0\0\0\0\0user1\0\0\0\0\0\0\0pass0\0\0\0\0\0\0\0\0\0\0\2",
		process->trigger(&test_packet, 0);
			test_jobs_info[0].job_id == 1 &&
			
			test_jobs_info[1].job_id == 3 &&
			
			test_jobs_info[2].job_id == 4 &&
			
			test_jobs_info[3].job_id == 5 &&
			
			test_jobs_info[4].job_id == -1

NPS UTC 3000_018	rquest_to_cancel_printing_process	// 세번째 작업 취소
		test_packet.data = "\0\0\0\0\0user2\0\0\0\0\0\0\0pass2\0\0\0\0\0\0\0\0\0\0\3",
		process->trigger(&test_packet, 0);
			test_jobs_info[0].job_id == 1 &&
			
			test_jobs_info[1].job_id == 2 &&
			
			test_jobs_info[2].job_id == 4 &&
			
			test_jobs_info[3].job_id == 5 &&
			
			test_jobs_info[4].job_id == -1

NPS UTC 3000_019	rquest_to_cancel_printing_process	// 네번째 작업 취소
		test_packet.data = "\0\0\0\0\0user3\0\0\0\0\0\0\0pass3\0\0\0\0\0\0\0\0\0\0\4",
		process->trigger(&test_packet, 0);
			test_jobs_info[0].job_id == 1 &&
			
			test_jobs_info[1].job_id == 2 &&
			
			test_jobs_info[2].job_id == 3 &&

			test_jobs_info[3].job_id == 5 && test_jobs_info[4].job_id == -1

NPS UTC 3000_020	rquest_to_cancel_printing_process	// 다섯번째 작업 취소
		test_packet.data = "\0\0\0\0\0user4\0\0\0\0\0\0\0pass4\0\0\0\0\0\0\0\0\0\0\5",
		process->trigger(&test_packet, 0);
			test_jobs_info[0].job_id == 1 &&
			
			test_jobs_info[1].job_id == 2 &&
			
			test_jobs_info[2].job_id == 3 &&
			
			test_jobs_info[3].job_id == 4 &&
			
			test_jobs_info[4].job_id == -1

NPS UTC 3000_021	user_creation_process	// _exist_user 함수가 유저를 정상적으로 찾아내는가
		Valid
		
		test_user.user_id = "test\0\0\0\0\0\0\0\0"
		
		v_header.nextInfo = &test_user;
		test_user.prevInfo = &v_header;
		test_user.nextInfo = &v_tailer;
		v_tailer.prevInfo = &test_user;
		
		test_user_packet = {
		.data = "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0test\0\0\0\0\0\0\0\0scott\0\0\0\0\0\0\0",
		.length = TYPE_LENGTH + REQ_ID_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH,
		.sock = 0
		};
		
		process->trigger(&test_user_packet);
		
		output = process->_exist_user("test\0\0\0\
			output == 1

		0\0\0\0\0")	

NPS UTC 3000_022	user_creation_process	//DB에 유저 정보가 없을경우 _exist_user 함수가 유저가 없다고 제대로 체크하는가
		Valid
		
		v_header.prevInfo = NULL;
		v_header.nextInfo = &v_tailer;
		v_tailer.prevInfo = &v_header;
		v_tailer.nextInfo = NULL;
		
		test_user_packet = {
		.data = "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0test\0\0\0\0\0\0\0\0scott\0\0\0\0\0\0\0",
		.length = TYPE_LENGTH + REQ_ID_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH,
		.sock = 0
		};
		
		process->trigger(&test_user_packet);
		
		output = process->_exist_user("test\0\0\0\0\0\0\0\0")
			output != 1

NPS UTC 3000_023	user_creation_process	//DB에 유저 정보가 있을경우 잘 찾는지(1)
		Valid
		
		test_user.user_id = "test"
		test_user2.user_id = "a"
		test_user3.user_id = "b"
		
		v_header.nextInfo = &test_user;
		test_user.prevInfo = &v_header;
		test_user.nextInfo = &test_user2;
		test_user2.prevInfo = &test_user;
		test_user2.nextInfo = &test_user3
		test_user3.prevInfo = &test_user2
		test_user3.nextInfo = &v_tailer;
		v_tailer.prevInfo = &test_user3;
		
		test_user_packet = {
		.data = "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0test\0\0\0\0\0\0\0scott\0\0\0\0\0\0\0",
		.length = TYPE_LENGTH + REQ_ID_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH,
		.sock = 0
		};
		
		process->trigger(&test_user_packet);
		
		output = process->_exist_user("test\0\0\0\0\0\0\0\0")
			output == 1

NPS UTC 3000_024	user_creation_process	//DB에 유저 정보가 있을경우 잘 찾는지(2)
		Valid
		
		test_user.user_id = "test"
		test_user2.user_id = "a"
		test_user3.user_id = "b"
		
		v_header.nextInfo = &test_user;
		test_user2.prevInfo = &v_header;
		test_user2.nextInfo = &test_user;
		test_user.prevInfo = &test_user2;
		test_user.nextInfo = &test_user3;
		test_user3.prevInfo = &test_user;
		test_user3.nextInfo = &v_tailer;
		v_tailer.prevInfo = &test_user3;
			output == 1

		test_user_packet = {
		.data =
		"\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0test\0\0\0\0\0\0\0\0scott\0\0\0\0\0\0\0",
		.length = TYPE_LENGTH + REQ_ID_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH,
		.sock = 0
		};
		
		process->trigger(&test_user_packet);
		
		output = process->_exist_user("test\0\0\0\0\0\0\0\0")	

NPS UTC 3000_025	user_creation_process	//DB에 유저 정보가 있을경우 잘 찾는지(3)
		Valid
		
		test_user.user_id = "test"
		test_user2.user_id = "a"
		test_user3.user_id = "b"
		
		v_header.nextInfo = &test_user2;
		test_user2.prevInfo = &v_header;
		test_user2.nextInfo = &test_user3;
		test_user3.prevInfo = &test_user2;
		test_user3.nextInfo = &test_user;
		test_user.prevInfo = &test_user3;
		test_user.nextInfo = &v_tailer;
		v_tailer.prevInfo = &test_user3;
		
		test_user_packet = {
		.data =
		"\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0test\0\0\0\0\0\0\0\0scott\0\0\0\0\0\0\0",
		.length = TYPE_LENGTH + REQ_ID_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH,
		.sock = 0
		};
		
		process->trigger(&test_user_packet);
			output == 1

		output=_exist_user("test\0\0\0\0\0\0\0\0\0")) == 1	

NPS UTC 3000_026	user_creation_process	// DB에 요청받은 유저id가 없는경우
		Valid
		
		v_header.prevInfo = NULL;
		v_header.nextInfo = &v_tailer;
		v_tailer.prevInfo = &v_header;
		v_tailer.nextInfo = NULL;
		
		test_user_packet = {
		.data = "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0test\0\0\0\0\0\0\0\0scott\0\0\0\0\0\0\0",
		.length = TYPE_LENGTH + REQ_ID_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH,
		.sock = 0
		};
		
		output = _exist_user("test\0\0\0\0\0\0\0\0\0")
			output == 1

NPS UTC 3000_027	user_creation_process	// DB에 요청받은 유저id가 이미 있는경우
		test_user = {
		.user_id = "test\0\0\0\0\0\0\0\0\0"
		};
		
		v_header.prevInfo = NULL;
		v_header.nextInfo = &test_user;
		test_user.prevInfo = &v_header;
		test_user.nextInfo = &v_tailer;
		v_tailer.prevInfo = &test_user;
		v_tailer.nextInfo = NULL;
		test_user_packet = {
		.data = "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0test\0\0\0\0\0\0\0\0scott\0\0\0\0\0\0\0",
		.length = TYPE_LENGTH + REQ_ID_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH,
		.sock = 0
		};
			call NetworkTxInterface->send_packet ?

		process->trigger(&test_user_packet);	

NPS UTC 3000_028	user_listing_process	/* 정상 작동 테스트 */
		Valid
		v_header.nextInfo = &v_tailer;
		v_tailer.prevInfo = &v_header;
		// 테스트 유저 추가
		for (i = 0 ; i < 10 ; i++) {
		// 구조체 초기화 (공간 할당)
		tmp_usr[i] = (struct UserInfo *)malloc(sizeof(struct UserInfo));
		memset(tmp_usr[i], 0, sizeof(struct UserInfo));
		// ID
		memset(buf, 0, 12);
		sprintf(buf, "user%d", i);
		strcpy(tmp_usr[i]->user_id, buf);
		// PW
		memset(buf, 0, 12);
		sprintf(buf, "pass%d", i);
		strcpy(tmp_usr[i]->user_pw, buf);
		
		// 연결
		tmp_lnk = test_user_database.tailer->prevInfo;
		
		tmp_lnk->nextInfo = tmp_usr[i];
		tmp_usr[i]->prevInfo = tmp_lnk;
		tmp_usr[i]->nextInfo = test_user_database.tailer;
		
		test_user_database.tailer->prevInfo = tmp_usr[i];
		}
		test_user_packet = {
		.data = "\0\0\0\0\0",
		.length = TYPE_LENGTH + REQ_ID_LENGTH,
		.sock = 0
		};
		
		process->trigger(&test_user_packet);
			call NetworkTxInterface->send_packet ?

NPS UTC 3000_029	user_removal_process	// 존재하지 않는 유저의 삭제
		Valid
		
		// 리스트 초기화 (연결 생성)
		v_header.prevInfo = NULL
			res == 1

		v_header.nextInfo = &v_tailer
		v_tailer.prevInfo = &v_header
		v_tailer.nextInfo = NULL
		
		// 테스트 유저 생성
		test_user = (struct UserInfo *)malloc(sizeof(struct UserInfo))
		strcpy(test_user->user_id, "bill_gates_")
		strcpy(test_user->user_pw, "bill_gates_")
		v_header.nextInfo = test_user
		test_user->prevInfo = &v_header
		test_user->nextInfo = &v_tailer
		v_tailer.prevInfo = test_user;
		
		res = 1;
		res &= !process->_remove_exist_user("steve_jobs_"); // 유저 이름이 존재하지 않는다.
		res &= process->_remove_user_id_cmp("steve_jobs_", test_user->user_id); // 테스트 유저와 비교해봐서 없는가?	

NPS UTC 3000_030	user_removal_process	// 존재하는 유저를 삭제
		Valid
		start = TYPE_LENGTH + REQ_ID_LENGTH + USER_ID_MAX_LENGTH + USER_PW_MAX_LENGTH;
		
		p.data = (char*)malloc(start + 12);
		strcpy((char*)(p.data + start), "bill_gates_");
		p.length = start+12;
		process->trigger(&p);
			((v_header.nextInfo == &v_tailer) && (v_tailer.prevInfo == &v_header)) == 1

NPS UTC 4000_000	send_status_process	Valid job_info[0].state = 0 state = 0 (no wait_refill) char *msg = process->_transform();	*msg = “current_state: printing”
NPS UTC 4000_001	send_status_process	Valid job_info[0].state = 0 state = 1 (wait_refill) char *msg = process->_transform();	*msg = “current_state: waiting”
NPS UTC 4000_002	send_status_process	Valid job_info[0].state = 1 char *msg =	*msg = “currnt_state: removing”

		process->_transform();	
NPS UTC 4000_003	send_status_process	Valid test_ink_value = 140 char *msg = process->_transform();	*msg = “ink: 140”
NPS UTC 4000_004	send_status_process	Valid test_paper_value = 5 char *msg = process->_transform();	*msg = “paper: 5”

NPS UTC 5000_000	decrease_amount_of_ink_paper	Valid
		db->current_ink_value = 150;
		db->current_paper_value = 2;
		trigger(100);
			db->current_ink_value==50 &&
			db->current_paper_value == 1;

NPS UTC 5000_001	decrease_amount_of_ink_paper	Invalid db->current_ink_value = 150 db->current_paper_value = 0 trigger(151)	db->current_ink_value == 0 db->current_paper_value == 0
NPS UTC 5000_002	decrease_amount_of_ink_paper	Invalid db->current_ink_value = 0 trigger(-1)	db->current_ink_value == 0

NPS UTC 5000_003	job_queue_pop	//JobQueue에 데이터가 0개
		Valid
		
		for(i = 0; i < JOB_MAX_SIZE;
		i++){
		test_jobs_info[i].job_id = -1;
		test_jobs_info[i].fd = -1;
		}
		
		process->trigger();
		
		boolean = 1;
		for(i = 0; i < JOB_MAX_SIZE;
		i++){
		boolean = boolean &&
		(test_jobs_info[i].job_id == -1);
		}
			boolean == 1

NPS UTC 5000_004	job_queue_pop	//JobQueue에 데이터가 1개
		Valid
		for(i = 0; i < 1; i++){
		test_jobs_info[i].job_id = i;
		test_jobs_info[i].fd = -1;
		}
		for(i = 1; i < JOB_MAX_SIZE;
		i++){
		test_jobs_info[i].job_id = -1;
		test_jobs_info[i].fd = -1;
		}
			boolean == 1

		process->trigger();
		
		boolean = 1;
		for(i = 0; i < JOB_MAX_SIZE; i++){
		boolean &= (test_jobs_info[i].job_id == -1);
		}	

NPS UTC 5000_005	job_queue_pop	//JobQueue에 데이터가 2개 Valid
		for(i = 0; i < 2; i++){
		test_jobs_info[i].job_id = i;
		test_jobs_info[i].fd = -1;
		}
		for(i = 2; i < JOB_MAX_SIZE; i++){
		test_jobs_info[i].job_id = -1;
		test_jobs_info[i].fd = -1;
		}
		
		process->trigger();
		
		boolean = 1;
		for(i = 0; i < 1; i++){
		boolean &= (test_jobs_info[i].job_id == i+1);
		}
		for(i = 1; i < JOB_MAX_SIZE; i++){
		boolean &= (test_jobs_info[i].job_id == -1);
		}
			boolean == 1

NPS UTC 5000_006	job_queue_pop	//JobQueue에 데이터가 3개 Valid
		for(i = 0; i < 3; i++){
		test_jobs_info[i].job_id = i;
		test_jobs_info[i].fd = -1;
		}
		for(i = 3; i < JOB_MAX_SIZE; i++){
		test_jobs_info[i].job_id = -1;
		test_jobs_info[i].fd = -1;
		}
		
		process->trigger();
		
		boolean = 1;
		for(i = 0; i < 2; i++){
		boolean &= (test_jobs_info[i].job_id == i+1);
			boolean == 1

		}
		for(i = 2; i < JOB_MAX_SIZE;
		i++){
		boolean &=
		(test_jobs_info[i].job_id == -1);
		}	

NPS UTC
5000_007
	job_queue_pop	//JobQueue에 데이터가 4개
		for(i = 0; i < 4; i++){
		test_jobs_info[i].job_id = i;
		test_jobs_info[i].fd = -1;
		}
		for(i = 4; i < JOB_MAX_SIZE;
		i++){
		test_jobs_info[i].job_id = -1;
		test_jobs_info[i].fd = -1;
		}
		
		process->trigger();
		
		boolean = 1;
		for(i = 0; i < 3; i++){
		boolean &=
		(test_jobs_info[i].job_id == i+1);
		}
		for(i = 3; i < JOB_MAX_SIZE;
		i++){
		boolean &=
		(test_jobs_info[i].job_id == -1);
		}
			boolean == 1

NPS UTC
5000_008
	job_queue_pop	//JobQueue에 데이터가 5개
		for(i = 0; i < 5; i++){
		test_jobs_info[i].job_id = i;
		test_jobs_info[i].fd = -1;
		}
		
		process->trigger();
		
		boolean = 1;
		for(i = 0; i < 4; i++){
		boolean &=
		(test_jobs_info[i].job_id == i+1);
		}
		for(i = 4; i < JOB_MAX_SIZE;
		i++){
		boolean &=
		(test_jobs_info[i].job_id == -1);
		}
			boolean == 1

NPS UTC 5000_009	save_print_result	print_test(“test, “test”, 0);	buf == output
NPS UTC 5000_010	save_print_result	str = “testtest…..” // 32자 print_test(str, str[30자], 0);	buf == output

NPS UTC 5000_011	save_print_result	str = “a\na\na….” // 12줄 print_test(str, str[10줄], 0);	buf == output
NPS UTC 5000_012	next_page	Valid test_jobs_info[0].current_page = 0 trigger()	test_jobs_info[0].current_page == 1

NPS UTC 5000_013	printing_controller	//IDLE there is no job
		//don't call any function ??
		Valid
		test_jobs_info[0].job_id = -1;
		
		controller->tick();
		for(i = 0; i < 20; i++)
		controller->tick();
		controller->tick();
			!(call any function) ??

NPS UTC 5000_014	printing_controller	//IDLE -> job_ready -> Printing -> job_ready
		//call Decrease amount, Save Print result, Next Page ??
		Valid
		test_jobs_info[0].job_id = 0;
		controller->tick();
		for(i = 0; i < 20; i++)
		controller->tick();
		controller->tick();
			call decrease_amount_ink_paper, save_print_result, next_page ??

NPS UTC 5000_015	printing_controller	//job_ready -> System_Wait -> job_ready
		//!(call Decrease amount, Save Print result, Next Page) ??
		
		Valid
		
		test_wait_ink_value = 1
		controller->tick();
			!(call any function) ??

NPS UTC 5000_016	printing_controller	//job_ready -> end_job -> IDLE
		//call job_queue_pop ??
		
		Valid
		
		test_wait_ink_value = 0
		test_jobs_info[0].current_page = test_jobs_info[0].total_page
		
		controller->tick();
		controller->tick();
			call job_queue_pop

NPS UTC	printing_controller	//IDLE -> job_ready -> end_job ->	

5000_017		IDLE Valid test_wait_ink_value = 0 cont	

7.5 Feature pass/fail criteria
NPS의 각 모듈은 SRA에 정의되어 있는 요구사항(입출력, 동작)을 모두 만족해야 한다. 각 프로세스의 입출력 및 동작은 SRA의 Process description 항목 및 State Transition Diagram을 참조한다.

8 Unit test case specification

8.1 Test case specification identifier
Test identification에 같이 명시

8.2 Test items
Test identification에 같이 명시

8.3 Input specifications
Test identification에 같이 명시

8.4 Output specifications
Test identification에 같이 명시

9 Testing tasks

Task	Predecessor Tasks	Special Skills	Effort	Finish Date
(1) Unit Test Plan 작성	SRA 작성 SDS 작성 DWS 구현		4	
(2) Test Design Specification	Task 1	NPS에 대한 이해		
(3) Test Case Specification	Task 2	NPS에 대한 이해		
(4) Test Execution	Task 3	Test Code 작성 Test Framework 에 대한 이해		
(5) Test Result Report	Task 4		1	
(6) 개발팀에 Report 전달	Task 5		1	

10 Environmental needs
C Language Complier/Linker : GCC

IDE : Cloud9

11 Unit Test deliverables

12 Schedules
Testing tasks 참조