

[100] PDF V2

Created by 김동규 dkd.kim, last modified by 조인성 insung.jo about an hour ago

Manual 목차
[01] Aurora2
[01][01] 배포용 zip으로 시작
[01][02] Aurora Template으로 시작
[02] 장비생성
[03] 화면 기본구성 및 구현
[03][01] 전지UI의 화면기본 구성 및 구현
[04]Option & Logger
[05] Log-In/Permission 설정
[06] Setting
[07] Recipe
[08] Model/Ingredient
[09] Event
[10] Devices
[11] Sequence
[12] UI-View
[13] BackUp
[14] Multi-Language
[15] Alarm
[16] MessageUtil
[17] DataBase
[17][01] MongoDB/Compass 설치 및 사용법
[17][02] Native DataHub(MongoDB) Driver

[01] Aurora2

Overview

Aurora 2.0은 LG 생산기술원 장비를 위한 운영SW 플랫폼이다.

운영SW에 필요한 공통 기능을 높은 품질로 제공하여 장비 적용시 운영SW 개발자는 장비 특화된 부분에 보다 집중할 수 있도록 하고, 다른 자사 플랫폼들과 연계성을 강화하여 통합된 솔루션으로써 효율적인 운영SW 개발을 목표로 한다.

Aurora 플랫폼의 계층 구조는 아래 그림1과 같다.

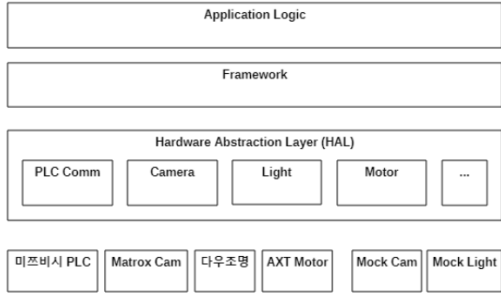


그림1. Hardware Abstraction Layer(HAL)를 통한 계층 분리

하드웨어 추상화 계층 (Hardware Abstract Layer)

Aurora 플랫폼은 장치(HW) 의존적인 부분을 HAL(Hardware Abstraction Layer)로 분리하여 독립적인 응용 계층 개발을 가능하도록 지원한다.

이를 위해 제어 모듈들은 플랫폼에서 정의된 인터페이스를 따라 구현하는 것을 권장하며, 기존에 플랫폼에서 확보된 제어 모듈은 이 인터페이스에 따라 구현하여 제공함으로써 재사용할 수 있게 한다. 또한, 시뮬레이션을 통해 HAL 상위의 응용 계층 개발이 가능하도록 Mock 구현을 제공한다.

공통 기능

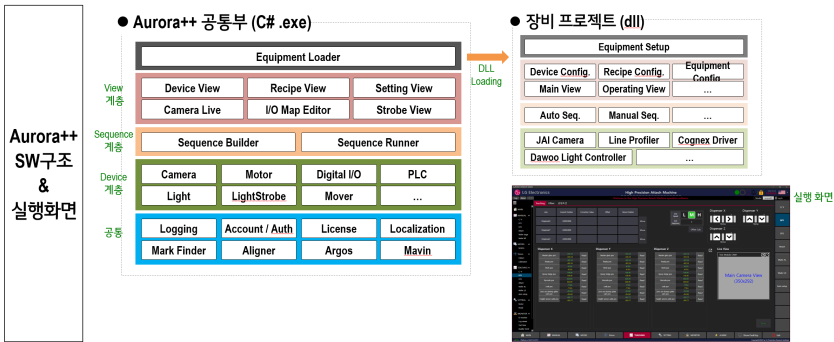
Aurora 플랫폼은 운영SW의 공통된 상위 기능을 Framework으로 제공한다. 장비 설정을 위한 **Settings**, 생산 모델에 대한 설정을 위한 **Recipe**, 자동공정 실행을 위한 **Sequence**, 실행이력 및 디버깅을 위한 Logging, 계정 컨트롤을 위한 Auth 등이 제공된다.

고정부와 변동부

Aurora 플랫폼은 그림2과 같이 공통 기능을 비롯한 재사용 가능한 모듈을 제공하는 플랫폼 공통부와 장비 특화된 부분을 담당하는 장비프로젝트(dll)로 구성되어 운영SW를 구성하게 된다.

플랫폼 고정부에는 Settings, Recipe, Sequence, Logging 등의 공통 기능, UI 컴포넌트, 자주 사용하는 PLC I/O, 카메라, 조명, 모터 등의 Device, 그리고 Mavin 등과 같이 자사 플랫폼들과 연동할 수 있는 모듈들을 제공한다.

장비Project에서는 플랫폼에서 제공하는 기능에 더하여 장비 특화된 제어 모듈과 UI들을 추가하여 요구사항에 맞는 운영 시나리오 로직을 구현하게 된다.



```
Aurora
├─ _config // Aurora.exe.config와 hislog.config 파일을
  보관할 빈 폴더
├─ bin
  └─ Aurora.exe // Aurora 플랫폼의 C# 실행파일.
고정부
├─ include //C/C++ Native 장치 제어모듈 구
  현을 위한 헤더파일들
├─ lib // C/C++ Navtive 장치 제어모듈을
  위한 라이브러리들
├─ csharp_lib // Aurora 플랫폼을 사용하기 위
  해 필요한 라이브러리들
├─ csharp_runtime // Aurora 플랫폼에서 사용되는
  runtime 라이브러리들
```

- 대표적인 폴더와 파일의 내용은 다음과 같다.
 - Aurora/bin/Aurora.exe
 - Aurora 플랫폼의 실행을 제공하는 파일이다. 중요한 역할은 실행하면서 변동부 기능을 제공하는 dll을 찾아 로드하는 것이다. 그리고 변동부에게 플랫폼의 공통기능을 제공한다.
 - Aurora/include
 - Aurora 플랫폼은 C/C++ 언어로 장치 제어모듈을 구현하여 사용할 수 있는 방법을 제공한다. 이 제어모듈을 Aurora 플랫폼으로 adapt하기 위한 인터페이스들이 정의된 헤더파일들이 여기에 위치한다.
 - Aurora/lib
 - C/C++ Native를 위한 Logging 등의 utility와 장치 제어모듈을 Aurora 플랫폼으로 adapt하기 위한 기본 라이브러리들이 위치한다. 여기에 포함된 dll들은 실행시에 Aurora.exe와 함께 위치하여야 한다.
 - Aurora/csharp_lib
 - Aurora 플랫폼의 실행을 제공하는 파일이다. 중요한 역할은 실행하면서 변동부 기능을 제공하는 dll을 찾아 로드하는 것이다. 그리고 변동부에게 플랫폼의 공통기능을 제공한다.
 - Aurora/csharp_runtime
 - Aurora 플랫폼에서 실행 중(runtime)에 사용되는 라이브러리들이 위치한다.

[01][01] 배포용 zip으로 시작

Set-Up

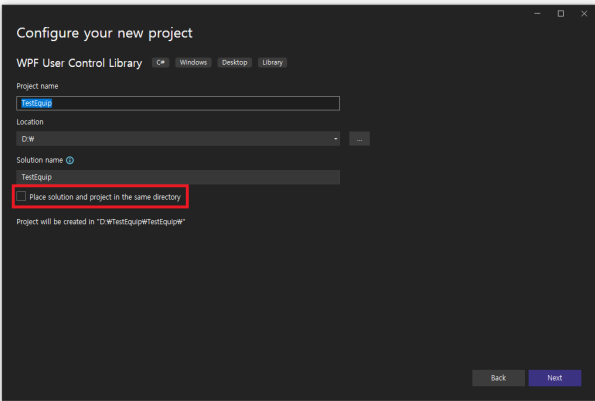
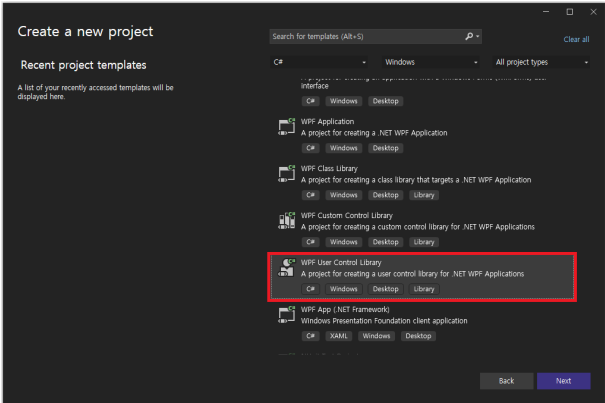
| 배포용 zip 으로 Project 생성 / 빌드 / 실행

1. 변동부의 구현은 배포된 Aurora.zip을 사용하여 새로운 Equipment project를 생성하여 구현하도록 하고 있음.
2. TemplateProject로 시작할 경우 아래 내용은 스킵할 것. (TemplateProject는 [\[01\]\[02\]Aurora Template으로 시작](#) 참고)

Set-Up 과정

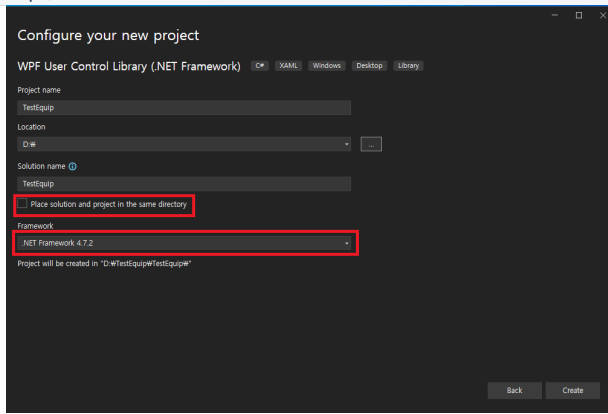
1) Project 생성

- Visual Studio 실행 > "Create a new project" > "WPF User Control Library(.NET Framework)" 선택 > Next 버튼
> Project name, Location 설정 > "Place solution and project in the same directory" 옵션 OFF > ".Net Framework 4.7.2" 설정 > "Create" 버튼



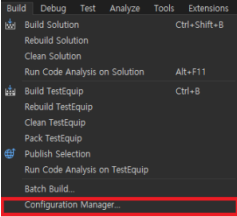
* 위의 빨간색 박스 안, 솔루션 및 프로젝트를 같은 디렉토리에 배치하는 체크 해제

Set-Up 과정

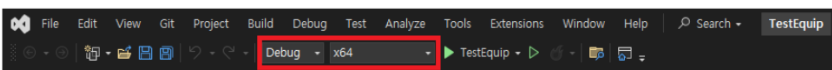
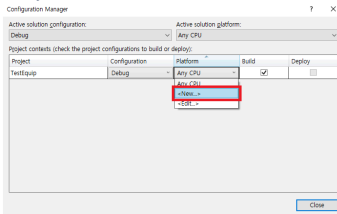


2) Build Configuration 수정

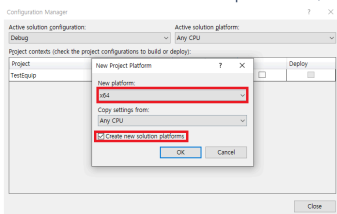
→ Build > Configuration Manager... 실행



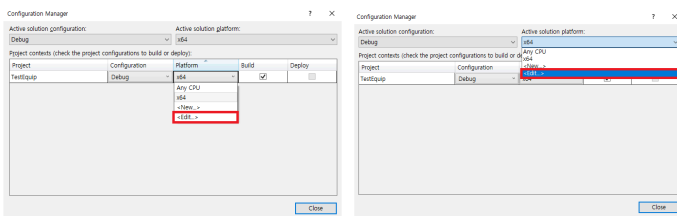
→ Project Platform과 Active Solution Platform을 모두 x64로 변경해준다.



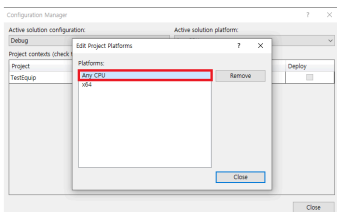
→ x64를 만들고 Create new solution platform을 체크한다.



→ Edit를 통해 지원하지 않는 Platform은 삭제한다. (Any CPU)



→ Any CPU Remove



→ 아래와 같이, 툴바에 x64가 활성화 되어있는지 확인

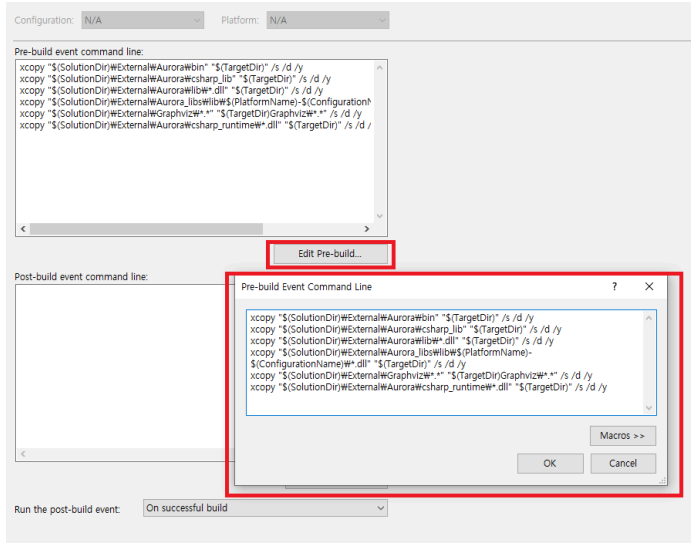
Set-Up 과정

3) 배포된 Aurora.zip 사용 설정

(1) Solution Directory 하위에, V2 메뉴얼에 따라 배포된 압축해제된 External 폴더를 넣음.



(2) 해당 Project 마우스 오른쪽 click > Properties > Build Events 선택 > "Pre-build event command line" 설정
(참고 : build 시작 전, 해당 file 들을 TargetDir 로 copy 실행)

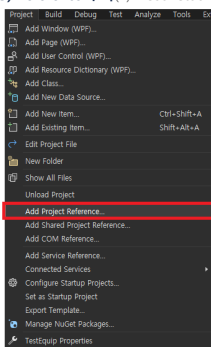


// 자신의 폴더 환경에 맞게 적절하게 변경 가능

```
xcopy "$(SolutionDir)External\Aurora\bin" "$(TargetDir)" /s /d /y
xcopy "$(SolutionDir)External\Aurora\wsharp_lib" "$(TargetDir)" /s /d /y
xcopy "$(SolutionDir)External\Aurora\lib*.dll" "$(TargetDir)" /s /d /y
xcopy "$(SolutionDir)External\Aurora\lib\$(PlatformName)\$(ConfigurationName)\$(ConfigurationName)\wsharp_runtime*.dll" "$(TargetDir)" /s /d /y
xcopy "$(SolutionDir)External\Aurora\lib\$(PlatformName)\$(ConfigurationName)\wsharp_runtime*.dll" "$(TargetDir)" /s /d /y
xcopy "$(SolutionDir)External\Aurora\lib\$(PlatformName)\$(ConfigurationName)\wsharp_runtime*.dll" "$(TargetDir)" /s /d /y
```

(3) 해당 Project 마우스 오른쪽 click > "Build" 선택 빌드 시 성공여부 확인

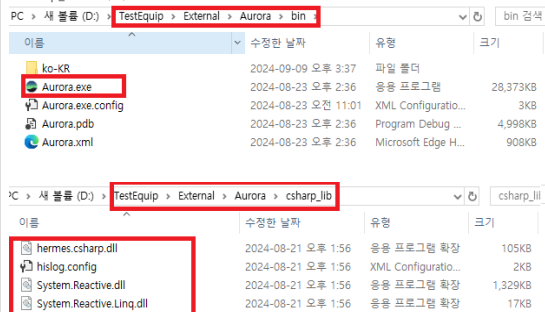
5) Reference 추가(1) Visual Studio > "Project" bar click > "Add Reference ..." 선택 > Browse



(2) 아래 "Browse..." 버튼을 통해 해당 파일들을 add 해 준다.

- Aurora.dll

- csharp_lib 폴더의 dll 들



6) Start 프로그램 지정 (Aurora.exe)

- 빌드하고자 하는 Equipment 에 Start 프로그램 지정

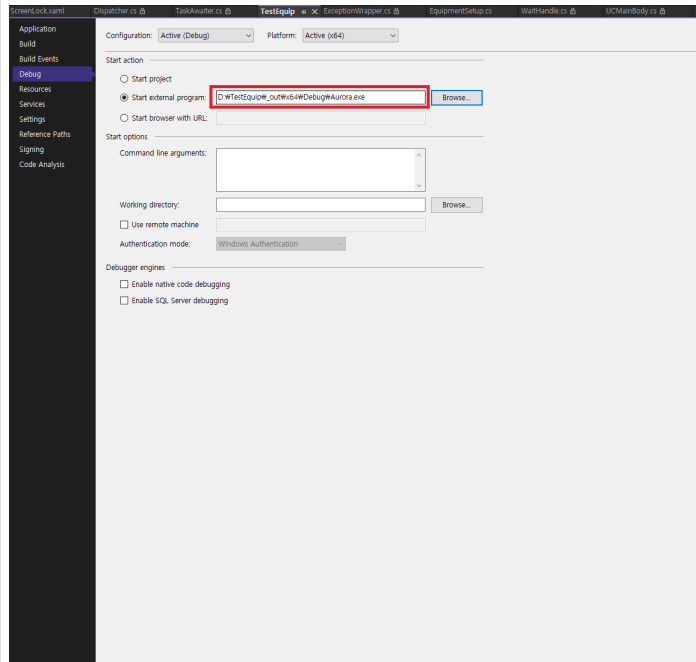
(Visual Studio > 해당 project > 마우스 오른쪽 click > Properties > Debug > "Start external program" path 지정)

Set-Up 과정

: project target 폴더W**Aurora.exe**

(ex. D:WAuroraMachineWbinWx64-DebugW**Aurora.exe**

(주의 : External 폴더의 Aurora.exe 가 아님 (※ Build 후에 Post script로 binWx64WDebug에 복사된 Aurora.exe를 실행하는 것임)



7) 빌드 확인 (현 단계에서는 빌드만 가능)

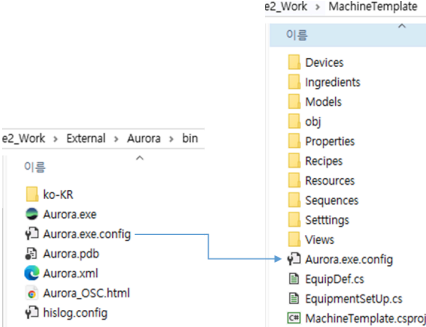
참고) UserControl1.xml은 자동생성된 파일이며 사용하지 않을 경우 지워도 무방하다

| Config 파일 설정

- 플랫폼 업그레이드 시 ExternalWAurora 폴더를 덮어쓸 경우 config 파일이 함께 덮어써지는 문제가 있다.
이를 방지하기 위한 Aurora.exe.config 환경 세팅에 대해 알아본다.
- TemplateProject로 시작할 경우 아래 내용은 스킵할 것. (TemplateProject는 [\[01\]\[02\]Aurora Template으로 시작](#) 참고)

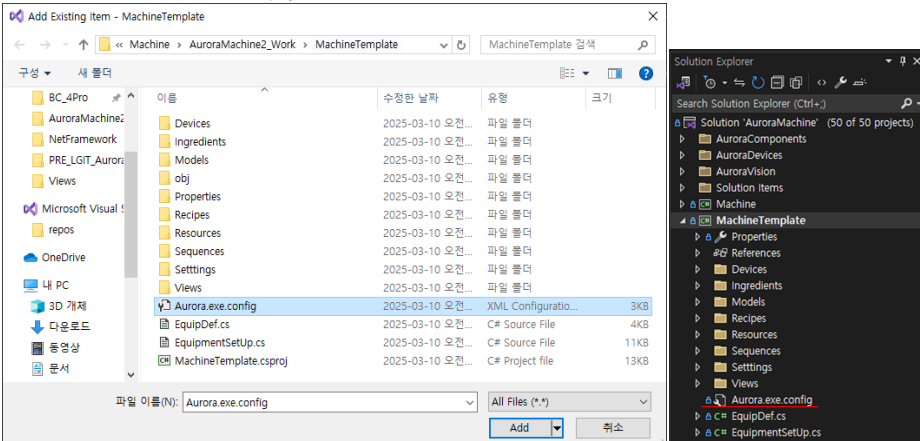
1) 파일 복사

- External - Aurora - bin 안에 들어있는 Aurora.exe.config 파일을 project 폴더에 복사한다.



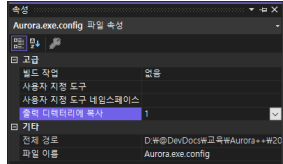
2) 프로젝트에 추가

- Visual Studio의 솔루션 탐색기에서 장비 project - 추가 - 기존항목을 선택하고 복사한 두 파일을 선택한다.



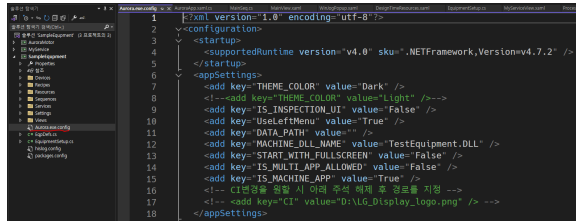
3) 속성변경

- Visual Studio의 솔루션 탐색기에서 해당 file을 우클릭 - 속성으로 진입하여 빌드 시 복사되도록 설정한다.



4) config 수정

- 이제 해당 config를 Visual Studio에서 수정하고 빌드 시 바로 적용된다.

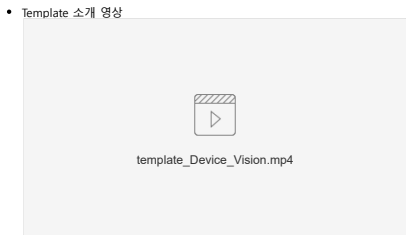


[01][02] Aurora Template으로 시작

Overview

| 개요

- Template 설명
 - 배경
 - 장비 개발자의 초반 장비 프로그램 개발의 용이성을 위해, 최소한의 샘플코드와 플랫폼에서 제공하는 화면들을 모두 탑재한 Template Project를 제공
 - 참고
 - 플랫폼에서 제공하는 화면과 주요 기능으로 위주로 Template 프로그램 구성
 - Machine은 개발에 참고할 샘플 프로젝트로 다양한 샘플코드로 구성
 - Template Project는 초반 개발 시작 단계에서 Base 코드로 사용
 - Template는 Machine v 2.1.1.1 이후로 제공



| Template 구성

- 화면 기본 구성 및 고정부 뷰 등록
 - 기본 화면 등록
 - 플랫폼 제공 공통 뷰 등
 - 커스텀 뷰 등록 예제
 - 고정 화면 Component 세팅
- Recipe/Model/Ingredient 사용
 - Reipe/Model/Ingredient 등록
 - 커스텀 Recipe 뷰 등록
- Device 사용
 - 공용화 Device 사용 예제 (Aurora PLC 사용)
 - 커스텀 Device 뷰 등록
- Seuquence 사용
 - Sequence 등록
 - Sequence 모니터링 뷰 등록
- Setting 사용
 - 사용자 정의 Setting 뷰 등록
- 기타
 - 로그 설정
 - Permission 설정
 - Event Publish 사용 예제

Set-up

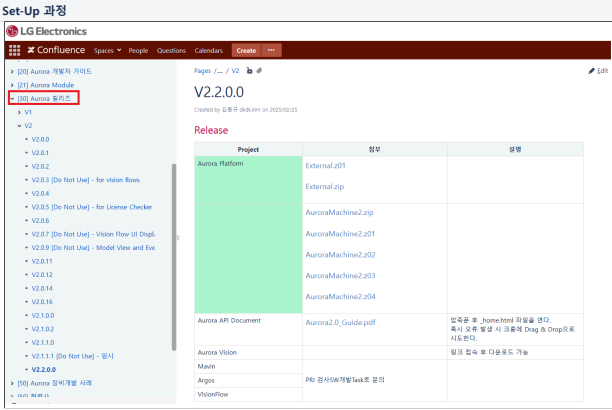
| Aurora Template Setting

1. Template 프로젝트는 Machine 프로젝트안에 생성되어 있음

Set-Up 과정

1) Template Project 다운

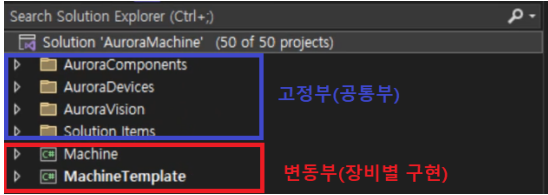
- 해당 Aurora Template은 Machine 소스 코드에 포함되어 있으며, 해당 Machine 소스는 아래와 같이 배포 페이지에서 다운로드 받을 수 있음 (배포페이지 링크 - [V2](#))



- 해당 페이지의 AuroraMachine2의 압축파일을 받고 해제를 진행

2) Template Project 소스 구조 설명

- 다운로드 받은 솔루션은 고정부(공통부)와 변동부(장비별 구현) 프로젝트로 구성됨

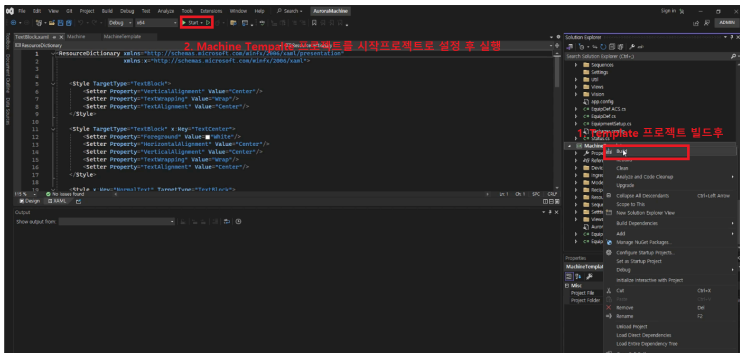


- 폴더 및 프로젝트 설명

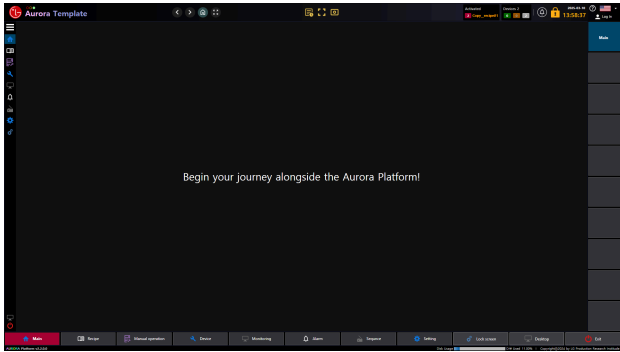
- AuroraDevices : PLC, Laser 등 장치와 인터페이스인 Device 라이브러리와 테스트뷰로 구성되며 약 100여개의 미리 구현된 디바이스를 제공
- UIComponent : 각종화면을 구현하기 위한 컨트롤, 색상, UI 도구로 구성
- AuroraVision : VisionAlign을 위한 라이브러리
- Machine: 개발에 참고할 샘플 프로젝트로 다양한 샘플코드로 구성

3) 빌드 및 실행

- MachineTemplate 프로젝트 빌드 후 실행



- 실행을 하게되면, 아래와 같이 Machine Template 초기화면이 나온다



[02] 장비생성

0. Overview

- Aurora 장비 DLL 은 하나의 장비를 가질 수 있으며, 장비 DLL 이 여러개인 경우 이중 먼저 검색되는 파일로 로딩됩니다.
- Aurora.exe 는 초기화 시 같은 폴더의 C# DLL 들을 검색하여, Aurora.Api.EquipmentSetup 인터페이스를 상속하여 구현한 첫번째 DLL 을 찾아 Load 하게 됩니다. 즉, 장비를 생성하기 위해서는 Aurora.Api.EquipmentSetup 을 상속받아 구현되어야 합니다.
- 여기서는 IEquipmentSetup 인터페이스 에 대해서 설명합니다.
- IEquipmentSetup가 있어지만 장비가 실행 됩니다.
- 장비와 장치에 대해서 혼동하지 않으시길 바랍니다. 장치는 개별 Device 입니다.

1. IEquipmentSetup

- Aurora 장비 Project를 위한 필수 구현 인터페이스

1.1 IEquipmentSetup Interface 정의

- Aurora 장비 Project(Dll)에서는 반드시 Aurora.Api.IEquipment Interface를 구현한 class가 제공되어야 합니다.
- 이 class는 Aurora.exe에서 DLL 로드시 인스턴스화하여, 필요한 멤버 메서드를 호출합니다.
- 변동부에서는 이 호출에 따라 필요한 정보를 제공하여 초기화를 하게 됩니다.

```
public interface IEquipmentSetup
{
    IAuroraEnvironment AuroraEnv { get; set; }
    string GetEquipmentName();
    void RegisterSettings(SettingRegister register);
    void RegisterIngredients(IngredientsRegister register);
    void RegisterModels(ModelRegister register);
    void RegisterRecipes(RecipeRegister register);
    void RegisterServices(IServiceRegistry register);
    void RegisterDevices(IDeviceRegistry registry);
    void RegisterSequences(SequenceRegister register);
    void RegisterViews(ViewManager manager);
    void RegisterShortcuts(ShortcutRegister register);
    void BuildMainStatusView(MainStatusBuilder builder);
    void RegisterLanguageResource(LangResourceRegister register);
    bool CanShutdown();
    void OnStartup();
    void OnShutdown();
    void OnSetAlarm(int code, bool isHeavy);
    void OnResetAlarm(List<int> codeList);
}
```

1.2 AuroraEnv

- 플랫폼에서 제공하는 서비스와 매니저의 집합체로 사용자가 등록한 각종 정보에 접근하는 기능이 있고 다음 과 같은 Property로 구성됩니다.
(여기서는 간략히 다루고 실제로 사용되는 부분에서 자세히 다룹니다.)

Type	Property	설명
IEventPublisher	EventPublisher	Event를 발생시킬 때 사용
IEventObservable	EventObservable	Event를 받고 심을 때 사용
ISettingEntry	SettingEntry	Setting을 가져오는 등의 API 제공
IModelManager	ModelManager	Model을 가져오는 등의 API 제공
IRecipeManager	RecipeManager	Recipe를 가져오는 등의 API 제공
IIngredientsManager	IngredientsManager	Ingredients를 가져오는 등의 API 제공
ISquenceEntry	SequenceEntry	시퀀스를 가져오는 등의 API 제공
IServiceManager	ServiceManager	서비스를 가져오는 등의 API 제공
IStreamManager	StreamManager	Stream 관련 API 제공
IDeviceManager	DeviceManager	디바이스를 가져오는 등의 API 제공
IAlarmService	AlarmService	Alarm을 발생시키는 등의 API 제공
IViewControlManager	ViewManager	Aurora 플랫폼 View를 제어하는 API 제공 (TitleBar의 아이콘 상태변경 등)

1.3 Interface 정의된 Method

- Interface에 정의된 메서드들에 대해서 간략하게 설명합니다.

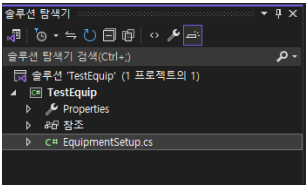
정의된 Method	설명
BuildMainStatusView	운영SW의 다국어 추가와 장치/서비스의 활성화 상태 표시 등의 설정을 한다.
CanShutdown	사용자가 운영SW의 닫기 버튼을 눌렀을 때 종료 가능 여부를 확인하기 위해 호출된다. 변동부에서 종료 가능 상태 여부를 판단하여 true/false로 반환한다.
GetEquipmentName	장비의 이름을 반환하도록 한다. 타입은 String이다. 이는 주후 장비의 정보를 나타내기 위해 사용할 수 있다.
OnShutdown	Aurora 플랫폼에서 실행 종료시 메서드를 호출한다. 따라서 변동부에서 운영SW 종료시 실행하고자 하는 로직을 여기에 넣을 수 있다.
OnStartup	Aurora 플랫폼에서 초기화 완료후 이 메서드를 호출한다. 따라서 변동부에서 운영SW 시작시 실행하고자 하는 로직을 여기에 넣을 수 있다.
RegisterDevices	장비를 제어하기 위한 제어모듈을 Device로 등록할 수 있도록 한다. Device 목록은 플랫폼에서 관리되고 변동부에서 조회 및 사용이 가능하도록 제공한다. 디지털 I/O, 모터, 조정 등의 유형별로 정의된 인터페이스가 있으며, 이 이러한 인터페이스를 구현하여 등록된 제어모듈은 플랫폼에서 제공하는 Device 설정 UI를 제공받을 수 있다. 만약 플랫폼에서 제공하는 UI가 적용상황에 적합하지 않을 경우 커스텀 UI를 사용할 수 있다.
RegisterLanguageResource	다국어 지원에 필요한 각 언어별 토큰이 정의된 리소스를 등록한다.
RegisterModel	장비운영에 필요한 모델정보를 정의하여 등록할 수 있다. 모델과 레시피는 구별되며, 보통 레시피가 없고 모델만 있는 경우에는 모델을 레시피로 취급한다. 모델은 정의된 레시피에서 사용하는 이름으로 된 정보로 존재할 수 없고, 레시피에 의존적이다.
RegisterRecipes	장비 운영에 필요한 레시피들을 정의하여 등록할 수 있도록 한다. 여기서 레시피란 함은 장비가 다루는 생산제품에 따른 설정을 의미한다. 등록된 레시피는 플랫폼에서 자동으로 조회 및 편집에 필요한 UI와 저장기능을 제공한다. 만약 플랫폼에서 자동 생성된 UI가 적용상황에 적합하지 않을 경우 커스텀 UI를 사용할 수 있다.
RegisterIngredients	장비 운영에 필요한 Ingredients를 정의하고 등록할 수 있다. Ingredients는 일반 공정/가공레시피외에 sub recipe 또는 parts recipe를 정의할때 사용한다.
RegisterSequences	장비의 자동공정을 정의한 Sequence를 등록할 수 있도록 한다. Sequence는 변동부에서 SequenceBuilder를 사용하여 정의하며, 등록된 Sequence는 변동부의 요정에 따라 고정부에서 실행, 중지 서비스를 제공한다.
RegisterService	Device와 유사하지만 실물이 없는 경우(ex FTP, Disk Cleaner, DB etc)는 Service로 등록한다.
RegisterSettings	장비 운영에 필요한 설정들을 정의하여 등록할 수 있도록 한다. 등록된 설정은 플랫폼에서 자동으로 조회 및 편집에 필요한 UI와 저장기능을 제공한다. 만약 플랫폼에서 자동 생성된 UI가 적용상황에 적합하지 않을 경우 커스텀 UI를 사용할 수 있다.
RegisterViews	운영SW의 화면에 보여질 View들을 등록한다. Aurora 플랫폼에서는 기본적으로 Device View, Setting View, Recipe View 등을 제공하고, 적용상황에 필요한 커스텀 UI의 View를 등록할 수 있다. 등록된 View들은 메뉴 항목으로 구분되어 보이며, 이 View들은 사용자 권한에 따른 접근 설정이 가능하다.
OnSetAlarm	알람이 발생했을 때 호출된다. 따라서 알람 발생 직후 실행하고자 하는 로직을 여기에 넣을 수 있다. 동시다발적으로 알람이 발생하는 경우 해당 횟수만큼 호출되며 알람이 파라미터로 전달된다.
OnResetAlarm	알람이 해제됐을 때 호출된다. 따라서 알람 해제 직후 실행하고자 하는 로직을 여기에 넣을 수 있다. 동시다발적으로 알람이 해제되며 해제된 알람이 리스트형태의 파라미터로 전달된다.
RegisterShortcuts	단축키를 등록하여 수행할 수 있다. id, key 조합을 등록하면 key가 눌렀을 때 해당 id를 event로 받을 수 있다.

2. 장비 Class 생성

- TemplateProject로 시작할 경우 아래 내용은 스킵할 것. (TemplateProject는 [01][02]Aurora Template으로 시작 참고)

| 2.1 Class 파일 생성

- 1장에서 Aurora++ 고정부의 배포 version zip과 함께 생성한 Project 를 Visual Studio 로 Open 합니다.
- 생성된 Project (장비) > Add > "EquipmentSetup.cs" 추가



- IEquipmentSetup 을 상속받은 장비 Class 생성

| 2.2 IEquipmentSetup 상속

IEquipmentSetup 을 상속받은 장비 Class 생성

- Interface 를 확인하면 아래와 같이 구현해야 하는 여러 Method 를 볼 수 있음.

```
public class EquipmentSetup : IEquipmentSetup
{
    public IAuroraEnvironment AuroraEnv { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

    public void BuildMainStatusView(MainStatusBuilder builder)
    {
        throw new NotImplementedException();
    }

    public bool CanShutDown()
    {
        throw new NotImplementedException();
    }

    public string GetEquipmentName()
    {
        throw new NotImplementedException();
    }

    public void OnResetAlarm(List<int> codeList)
    {
        throw new NotImplementedException();
    }

    public void OnSetAlarm(int code, bool isHeavy)
    {
        throw new NotImplementedException();
    }

    public void OnShutDown()
    {
        throw new NotImplementedException();
    }

    public void OnStartUp()
    {
        throw new NotImplementedException();
    }

    public void RegisterDevices(IDeviceRegistry registry)
    {
        throw new NotImplementedException();
    }

    public void RegisterIngredients(IngredientsRegister register)
    {
        throw new NotImplementedException();
    }

    public void RegisterLanguageResource(LangResourceRegister register)
    {
        throw new NotImplementedException();
    }

    public void RegisterModels(ModelRegister register)
    {
        throw new NotImplementedException();
    }

    public void RegisterRecipes(RecipeRegister register)
    {
        throw new NotImplementedException();
    }

    public void RegisterSequences(SequenceRegister register)
    {
        throw new NotImplementedException();
    }

    public void RegisterServices(IServiceRegistry register)
    {
        throw new NotImplementedException();
    }

    public void RegisterSettings(SettingRegister register)
    {
        throw new NotImplementedException();
    }

    public void RegisterViews(IViewManager manager)
    {
        throw new NotImplementedException();
    }
}
```

```

void RegisterShortcuts(ShortcutRegister register)
{
    throw new NotImplementedException();
}
}

```

| 2.3 장비 기본 수정

AuroraEnv 는 Aurora 환경변수로 고정부에서 자동으로 Set 해주는 변수 임.

이 변수를 Getter/Setter 로 변경 (throw new NotImplementedException() 로 Exception 발생하는 부분을 삭제)

```

public IAuroraEnvironment AuroraEnv { get; set; }

```

MainStatusView 환경을 Setting 해 줌.

- **AddLanguage**: 사용할 언어종류추가

- **AddDeviceActiveStatus**: 사용될 Device의 ID를 등록하여 활성/비활성 상태표시 (참고 : Device Id 의 경우, EquipDef.cs 파일에 정의하여 관리 함.)

- **AddDeviceGroupActiveStatus**: Device가 너무 많을 경우 여러개의 Device ID를 그룹화 할 수 있다.

```

public void BuildMainStatusView(MainStatusBuilder builder)
{
    builder.AddLanguage("ko-KR", "vi-VN", "en-US")
        .AddDeviceActiveStatus(DeviceIds.Scanner)
        .AddDeviceActiveStatus(DeviceIds.PLCDeivce)
        .AddDeviceGroupActiveStatus(
            DeviceIds.MotorDP,
            new List<string>() {
                DeviceIds.MotorDpX,
                DeviceIds.MotorDpY,
                DeviceIds.MotorDpZ});
}

```

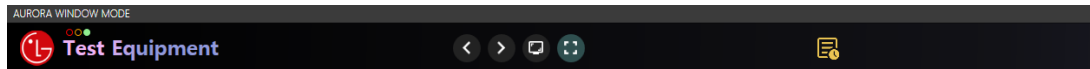
장비 이름을 설정해 줌.

- 설정된 이름은 Main Window 상단에 표시된다.

```

0 references
public string GetEquipmentName()
{
    return "Test Equipment";
}

```



CanShutDown 함수는 return 값이 필요하므로, 아래처럼 일단 바꿔줌. (실제, 종료 가능 상태 여부를 판단하는 로직이 들어 가는 부분으로, true/false 로 반환된다.)

```

public bool CanShutDown() {
    return true;
}

```

나머지 함수들의 "throw exception" (throw new NotImplementedException()) 을 지워서 빌드가 되는지 확인

- example)

```

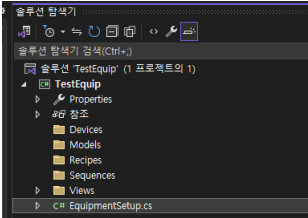
public void OnShutDown() {
    //throw new NotImplementedException();
}

```

장비 실행 후 화면에 잘 나오는지 확인



*참고사항
아래와 같은 폴더구조를 권장한다.

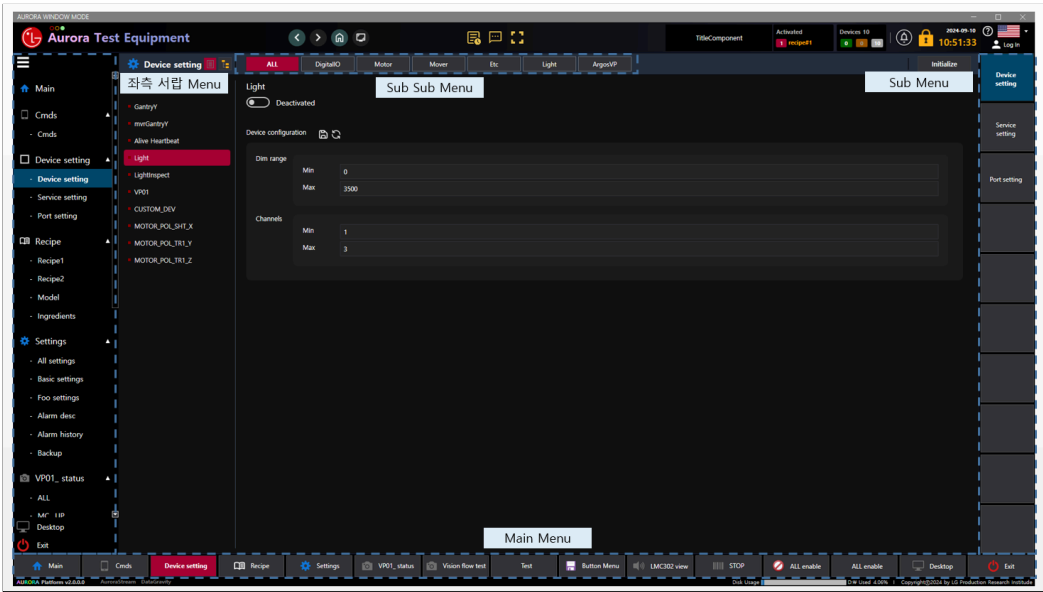


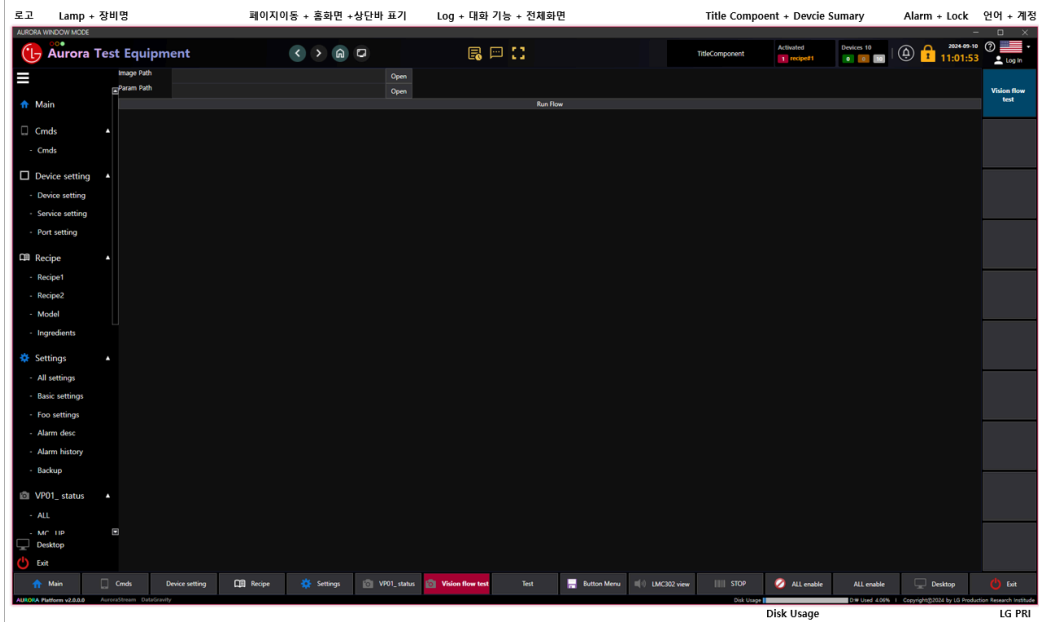
[03] 화면 기본구성 및 구현

0. Overview

- Aurora 프로그램의 기본 화면 구성 및 화면 구현 예시를 설명합니다.

1. 화면 기본 구성





2. View 구성 및 등록

- View `IEquipmentSetup`의 `RegisterViews(IViewManager manager)` 함수를 통해 xaml로 만든 화면과 함께 등록한다.
- `IViewManager`는 다음 기능을 가진다.

```
public interface IViewManager
{
    ExecutionResult RegisterMainMenu(MainMenu viewGroup);
    ExecutionResult RegisterTitleComponent(ContentControl control);
    ExecutionResult RegisterFixedComponent(ContentControl control);
    ExecutionResult RegisterStatusComponent(ContentControl control);
    ExecutionResult AddComponent(ContentControl control);
    ExecutionResult RegisterScreenInterlock(ContentControl control);
    ExecutionResult RegisterViewGroup(string groupName, ImageSource groupIcon = null, PermissionLevel permissionLevel = PermissionLevel.NONE);
    ExecutionResult RegisterView(string viewName, FrameworkElement view, PermissionLevel permissionLevel = PermissionLevel.NONE, ViewPlace place = ViewPlace.Dialog,
        string groupName = null, ImageSource icon = null, ImageSource icon = null, double iconMargin = double.NaN)
}
```

| 2.1 View Menu 등록

- `IViewManager`의 `RegisterMainMenu`를 통해 View와 함께 등록한다.
- 참고로 View는 Aurora 플랫폼에서 제공되는 공통 화면과 장비 DLL에서 만든 장비 특화 화면(`UserControl`)이 있다.
- 화면과 Menu를 등록하기 위해서는 `MainMenu`를 만들어야 하고 `MainMenu`의 생성자는 다음과 같다.

```
public MainMenu(string mainName, ImageSource mainIcon = null, PermissionLevel permission = PermissionLevel.NONE, bool showSubMenuBar = true)
```

main Name - 하단 `MainMenu`에 표시되는 버튼 text

mainIcon - `MainMenu`의 icon

permission - 해당 화면에 permission을 주고자 할 경우 사용

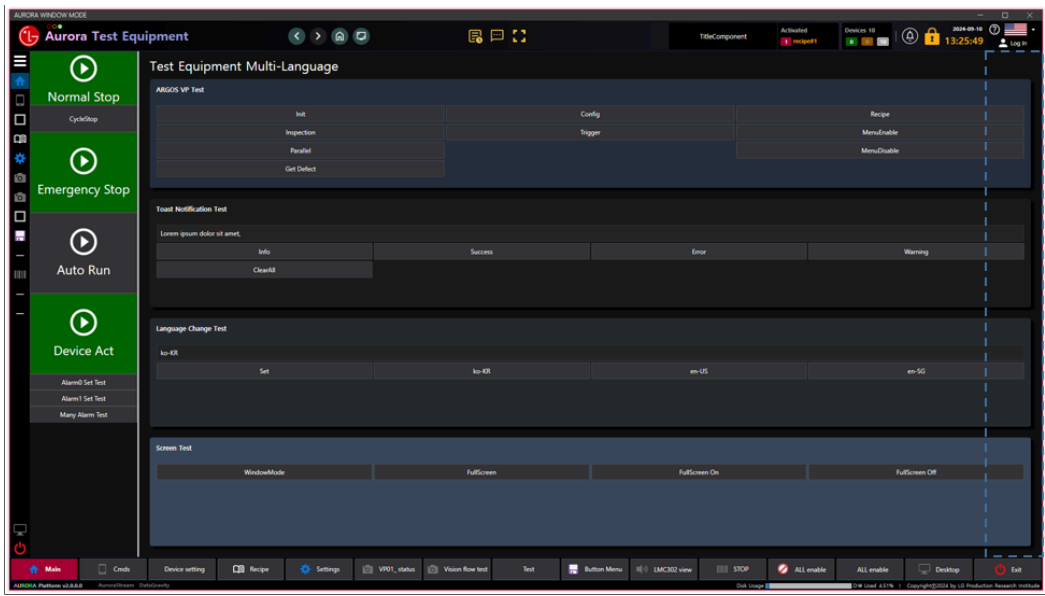
showSubMenuBar - `MainMenu`버튼을 누를 시 `SubMenu`영역을 숨기고 싶을 경우 `false`로 설정

- `MainMenu`는 `SubMenu`를 등록하기 위해 `AddSubMenu`를 제공한다.

```
public MainMenu AddSubMenu(string subName, FrameworkElement view)
```

subName - 우측 `SubMenu`에 표시되는 버튼 text

view - `SubMenu`버튼을 클릭 시 화면 중앙에 표현하고싶은 View
`UserControl`로 만들어서 추가할 수 있다.



<showSubMenuBar를 false로 할 경우 위와 같이 우측에 파란색 영역의 SubMenu가 사라지는 것을 볼 수 있다.>

- SubMenu까지 등록하였다면 해당 MainMenu Instance를 RegistMainMenu함수를 통해 등록한다.

```
var home = new MainMenu("Home", MenuIcon.GetMenuIcon(eMenuIcon.HOME), PermissionLevel.NONE);
home.AddSubMenu("HomeSub", new HomeView());
manager.RegisterMainMenu(home);
```

- 미리 정의된 Icon을 사용할 수 있다.

```
MenuIcon.GetMenuIcon(eMenuIcon.HOME)
```

```
public enum eMenuIcon
{
    HOME,
    SETTING,
    RECIPE,
    DEVICE,
    DESKTOP,
    EXIT,
    MIC,
    CAMERA,
    PRINTER,
    ROUTER,
    SCANNER,
    TABLET,
    TV,
    VIDEO,
    VOLUME,
    BARCODE,
    SAVE,
    GEAR,
    BOOK,
    LIBRARY_BOOK,
    ALARM,
    MAX
}
```

- 실제 UI의 화면 등록 코드(예시)

```
public void RegisterViews(IViewManager manager)
{
    InformPanel = new UCInformPanel(AuroraEnv);
    _Status = new UCStatus();
    //Icons
    ResourceDictionary IconDictionary = new ResourceDictionary();
    IconDictionary.Source = new Uri("pack://application:,,,/AuroraUIComponent;component/Resources/IconsDic.xaml");

    // Main
    var main = new MainMenu(ViewIds.Main, MenuIcon.GetMenuIcon(eMenuIcon.HOME), PermissionLevel.NONE, false);
    main.AddSubMenu(ViewIds.Main, new UCMainView() { DataContext = new UCMainViewModel(AuroraEnv) });
    manager.RegisterMainMenu(main);

    var recipe = new MainMenu(ViewIds.Recipe, MenuIcon.GetMenuIcon(eMenuIcon.RECIPE));
    recipe.AddSubMenu(ViewIds.RecipeSelect, new UCRecipe());
    recipe.AddSubMenu(ViewIds.RecipeAll, new RecipeSettingView());
    recipe.AddSubMenu("Model", new UCModel() { DataContext = new UCModelViewModel(AuroraEnv, null) });

    manager.RegisterMainMenu(recipe);

    // Vision
    //var visionView = new UCVision(AuroraEnv);
    //var vision = new MainMenu(ViewIds.Vision, MenuIcon.GetMenuIcon(eMenuIcon.CAMERA));
    //vision.AddSubMenu(ViewIds.LaserAlign, visionView);
    //manager.RegisterMainMenu(vision);
```

```

// Teaching
var teaching = new MainMenu(ViewIds.Teaching,
    new BitmapImage(new Uri("pack://application:,,,/CellRepair;component/Resources/Icons/Design_32x32.png")));
teaching.AddSubMenu(ViewIds.Teaching, new UCTeaching());
manager.RegisterMainMenu(teaching);

// Manual Operation
var uc = new UCManual(AuroraEnv);
var manual = new MainMenu(ViewIds.Manual, (DrawingImage)IconDictionary["Maint"], PermissionLevel.LEVEL3);
manual.AddSubMenu(ViewIds.Scanner, uc);
manual.AddSubMenu(ViewIds.LaserMaint, uc);
manual.AddSubMenu(ViewIds.SlitMaint, uc);
manual.AddSubMenu(ViewIds.AF, uc);
manual.AddSubMenu(ViewIds.Group_PG_LC_OCR, uc);
manual.AddSubMenu(ViewIds.ScannerCalibration, uc, PermissionLevel.LEVEL3);
manual.AddSubMenu(ViewIds.ScannerBlob, uc, PermissionLevel.LEVEL3);
manual.AddSubMenu(ViewIds.LaserMeasure, uc);
manager.RegisterMainMenu(manual);

// Settings
var setting = new MainMenu(ViewIds.Setting, MenuIcon.GetMenuIcon(eMenuIcon.SETTING), PermissionLevel.LEVEL3);
setting.AddSubMenu(ViewIds.DeviceSettings, new UCDeviceSettings() { DataContext = new UCDeviceSettingsViewModel(AuroraEnv) });
setting.AddSubMenu(ViewIds.LensSetting, new UCLensSetting()
{ DataContext = new UCLensSettingViewModel(AuroraEnv, SettingIds.SlitSetting) });
setting.AddSubMenu(ViewIds.WorkingCondition, new UCWCSetting(AuroraEnv) { });
manager.RegisterMainMenu(setting);

// monitoring
var monitoring = new MainMenu(ViewIds.Monitoring, (DrawingImage)IconDictionary["Monitoring"], PermissionLevel.LEVEL5);
monitoring.AddSubMenu(ViewIds.PLCBitMonitoring, new UCPLCBit()
{ DataContext = new UCPLCBitViewModel(AuroraEnv, DeviceIds.PLCDevice, PLCNetType.MelsecNet) });
//monitoring.AddSubMenu(ViewIds.PLCWordMonitoring, new UCPLCWordView(AuroraEnv));
monitoring.AddSubMenu(ViewIds.PLCWordMonitoring, new UCPLCWord()
{ DataContext = new UCPLCWordViewModel(AuroraEnv, DeviceIds.PLCDevice, PLCNetType.MelsecNet) });
manager.RegisterMainMenu(monitoring);

// Alarm
MainMenu Alarm = new MainMenu("Alarm", (DrawingImage)IconDictionary["AlarmBell"], PermissionLevel.NONE);
Alarm.AddSubMenu("History", new AlarmHistoryView());
Alarm.AddSubMenu("Description", new AlarmDescView());
// Application.Current.AuroraEnv().AlarmService.SetAlarm(code);
// Log.E($"Alarm {bufferNo}, {code}, {Application.Current.AuroraEnv().AlarmService.GetAlarmDescription(code).Name}");
manager.RegisterMainMenu(Alarm);

// door interlock
var btnMenu = new MainMenu("Door Interlock", (DrawingImage)IconDictionary["DoorLock"], PermissionLevel.NONE, false);
btnMenu.AddButtonType(OnDoorInterlockClick);
manager.RegisterMainMenu(btnMenu);

var device = new MainMenu(ViewIds.Device, MenuIcon.GetMenuIcon(eMenuIcon.DEVICE), PermissionLevel.LEVEL3);
device.AddSubMenu(ViewIds.Device, new Aurora.View.DeviceSettingView());
manager.RegisterMainMenu(device);

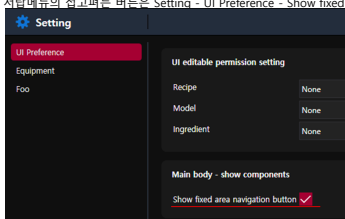
// System Settings
//icon = MenuIcon.GetSvgIcon("pack://application:,,,/Scriber;component/Resources/Icons/Save.svg");
//var systemSetting = new MainMenu(ViewIds.SystemSettings, icon, PermissionLevel.LEVEL3);
var systemSetting = new MainMenu(ViewIds.SystemSettings, MenuIcon.GetMenuIcon(eMenuIcon.SETTING), PermissionLevel.LEVEL3);
systemSetting.AddSubMenu(ViewIds.SystemSettings, new Aurora.View.EquipSettingView());
systemSetting.AddSubMenu(ViewIds.Backup, new AutoBackupView2() { DataContext = new AutoBackupViewModel() });
manager.RegisterMainMenu(systemSetting);
}

```

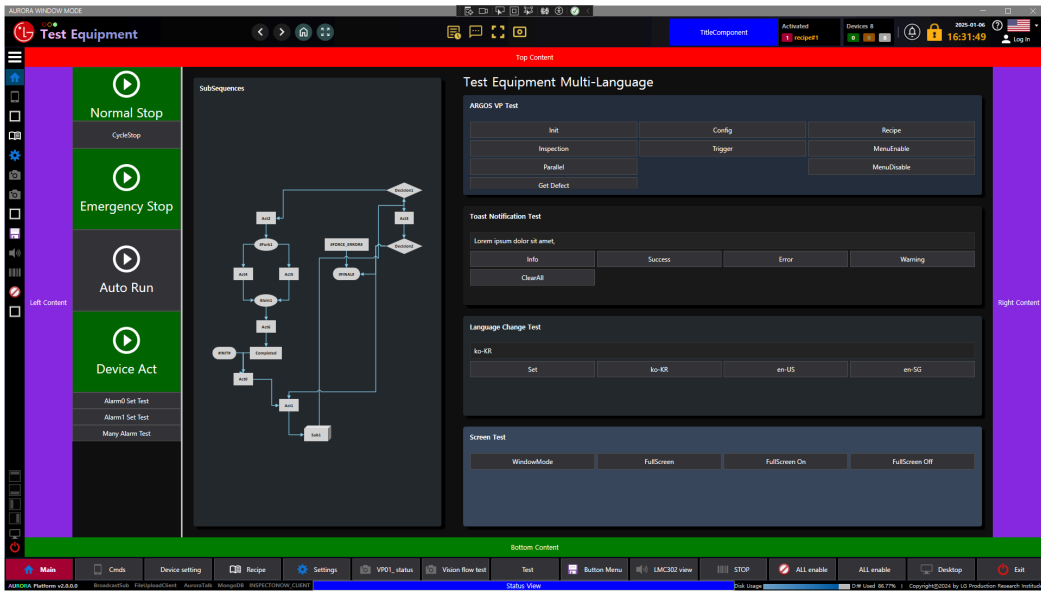
2.2 고정 영역 Component

- 앞서 살펴본 View와 같이 Menu버튼을 클릭 시 화면이 변경되는 영역이 있는가 하면 View 변경 시에도 변하지 않는 고정영역이 있다.

- Title Component - Titlebar의 Recipe 패널 좌측.
- Fixed Component - Top/Bottom/Left/Right에 배치. 좌하단 서랍메뉴의 버튼으로 집고 펴는게 가능.
 - 서랍메뉴의 집고펴는 버튼은 Setting - UI Preference - Show fixed area navigation button 을 enable해야 표시됨.



- Fixed Component는 마지막 접힘 상태가 저장됨.
- StatusComponent - 하단의 LG PRI표기 영역 왼쪽.



- IViewManager의 RegisterTitleComponent(ContentControl control) 함수를 통해 추가할 수 있다.

```
public void RegisterViews(IViewManager manager)
{
    manager.RegisterTitleComponent(new TitleContent());

    manager.RegisterFixedComponents(new UCFixedTopContent(),
                                    new UCFixedBottomContent(),
                                    new UCFixedLeftContent(),
                                    new UCFixedRightContent());

    manager.RegisterStatusComponent(new StatusView());
}
```

2.3 Handler 등록

- View가 변경될 때 변경된 View의 Main ID와 Sub ID를 받는 Handler를 등록
- Titlebar의 Title component 우측 recipe 패널과 device 패널 클릭 시 호출되는 Handler를 등록

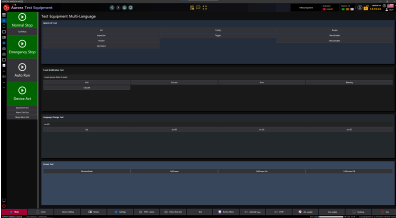
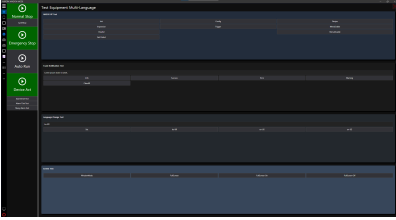
```
manager.ViewChangedHandler((string main, string sub) => {
    Logger.I($"View changed. {main}-{sub}");

    return true;
});
//manager.DeviceStatusMouseDownHandler(() => { }); // DeviceStatusMouseDownHandler는 등록하면 default device init popup이 안 뜨므로 주의할 것.
manager.RecipeStatusMouseDownHandler(() => { }); // RecipeStatus Panel을 누를 경우 발생된다.
```

2.4 기타 플랫폼 고정영역 기능별 설명

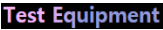
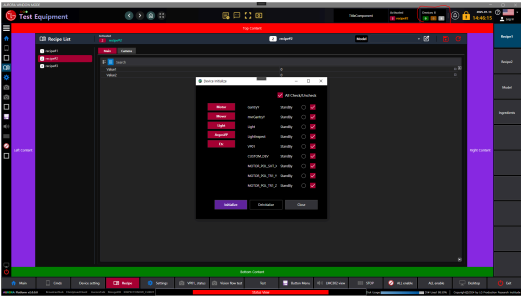
- 장비DLL에서 Aurora 플랫폼에 변경 요청 가능한 기능 및 UI
- 해당 기능은 IAuroraEnvironment.ViewManager에서 API로 제공됩니다.

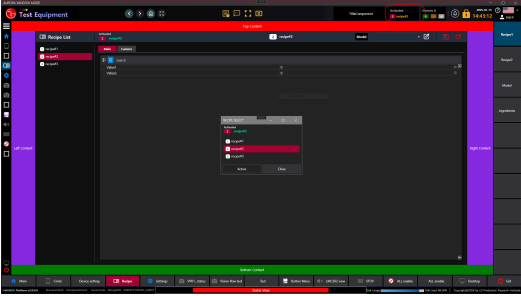
화면	기능	구현방법
	스크린 인터락	1. 앞장에서 언급한 RegisterViews함수에서 View Menu와 각종 고정영역 Component등록과 함께 ScreenLock에 사용할 화면도 등록 가능 <pre>RegisterViews (IViewManager manager) { manager.RegisterScreenInterlock(new ScreenLock ()); }</pre> 2. 장비DLL에서 스크린 인터락을 활성화 시키는 API <pre>IAuroraEnvironment.ViewManager.SetScreenInterlock(true);</pre> 3. ScreenLock() 화면에서 SetScreenInterlock(false)를 실행시킬 수 있도록 구현 c) 화면의 UI가 Lock UI으로 완전히 덮여져야 함.
	Door Lock	Event-Publisher로 변경 ViewManager에서 API로 제공 <pre>//EventPublisher.Publish(// new EventMessage("", AuroraSysEvent.SYS_REQ_MAIN_LOCK, false)); IAuroraEnvironment.ViewManager.DoorLock(false);</pre>
		Event-Publisher로 변경 ViewManager에서 API로 제공 <pre>//EventPublisher.Publish(// new EventMessage("", AuroraSysEvent.SYS_REQ_MAIN_LAMP, eTowerLamp.GREEN)); IAuroraEnvironment.ViewManager.SetTowerLamp(eTowerLamp.GREEN);</pre>

화면	기능	구현방법
		<pre>public enum eTowerLamp { ALL_OFF = 0, GREEN = 1, ORANGE = 2, RED = 3 }</pre>
	메뉴 변경	메뉴버튼을 클릭하지 않고 코드 상에서 변경하기 위한 API parameter로 이동하려는 MainMenu의 ID를 넘겨준다. <pre>IAuroraEnvironment.ViewManager.ChangeMenu("MainMenu ID");</pre>
	메뉴 Enable/Disable	메뉴버튼을 Disable시키는 API 시퀀스 동작 시 특정 메뉴 진입을 막는 용도 등으로 사용 1. enable/disable여부와 해당조건을 수행할 화면 id 복수개, 해당 id를 제외한 나머지는 설정한 값과 반대로 동작한다. MainMenuEnable(bool isEnabled, params string[] idList); 2. 일괄 enable/disable MainMenuAllEnable(bool isEnabled); <pre>IAuroraEnvironment.ViewManager.MainMenuEnable(true, "Main", "Monitoring"); IAuroraEnvironment.ViewManager.MainMenuAllEnable(true);</pre>
	로그 / AuroraTalk / Window Mode	1. 로그창, AuroraTalk 띄우기 2. 윈도우 모드 변경 - 전체화면, 창 화면 <pre>IAuroraEnvironment.ViewManager.WindowMode(true); // true : 전체화면, false : 창 화면</pre>
 	View 전체화면	Title영역, 메뉴버튼 영역까지 View영역으로 사용하도록 설정하는 API <pre>IAuroraEnvironment.ViewManager.ViewFullScreen(true);</pre> 우측상단의 X버튼이나 'ESC' 클릭으로 원래대로 복귀할 수 있다. (혹은 ViewFullScreen(false)로도 가능)

화면	기능	구현방법
	언어	1. IEquipmentSetup의 Addlanguage으로 설정 <pre>void IEquipmentSetup.BuildMainStatusView(MainStatusBuilder builder) { builder .AddDeviceActiveStatus(DeviceIds.MOCK_DIGITAL_IO, DeviceIds.MOCK_LIGHT, DeviceIds.MTR_GANTRY_Y) .AddLanguage("en-US", "ko-KR", "vi-VN");//.AddComponent(GetButton()); }</pre> 2. 코드상에서 다음 API로도 변경 가능 <pre>IAuroraEnvironment.ViewManager.ChangeLanguage("ko-KR");</pre>

• 기타 UI

화면	기능	구현방법
	CI	Aurora.exe.config에 아래와 같이 경로를 변경하여 장비 DLL에서 바꿀 수 있음 <pre><appSettings> <add key="CI" value="res/newCI.png" /> </appSettings></pre>
	장비명	IEquipmentSetup의 GetEquipmentName()으로 설정 <pre>public string GetEquipmentName() { return "Equipment implemented as Aurora"; }</pre>
	알람 여부	AuroraSysEvent.SYS_ALARM_SET_ALARM / SYS_ALARM_RESET_ALARM / SYS_ALARM_RESET_ALL_ALARM <pre>EventPublisher.Publish(new EventMessage("", AuroraSysEvent.SYS_ALARM_SET_ALARM, false));</pre> true or false
	시계	디폴트 표시
	Device Status	장비 활성화 상태 보기. IEquipmentSetup의 AddDeviceActivateStatus로 설정 <pre>void IEquipmentSetup.BuildMainStatusView(MainStatusBuilder builder) { builder .AddDeviceActiveStatus(DeviceIds.MOCK_DIGITAL_IO, DeviceIds.MOCK_LIGHT, DeviceIds.MTR_GANTRY_Y) }</pre> 클릭 시 아래와 같이 default Device Init View 표시  변동부에서 별도의 handler를 등록하여 자신만의 Popup을 만들어 띄우고자 한다면 아래와 같이 등록한다.

화면	기능	구현방법
		<pre>public void RegisterViews(IViewManager manager) { manager.DeviceStatusMouseDownHandler(() => { }); }</pre> 주의 : handler를 등록할 경우 default popup은 발생되지 않는다.
Activated 1 recipe#1	Active Recipe	활성화된 Recipe 이름 클릭 시 아래와 같이 default Recipe Select Popup이 표시  변동부에서 별도의 handler를 등록하여 자신만의 Popup을 만들어 띄우고자 한다면 아래와 같이 등록한다. <pre>public void RegisterViews(IViewManager manager) { manager.RecipeStatusMouseDownHandler(() => { }); }</pre> 주의 : handler를 등록할 경우 default popup은 발생되지 않는다.
Log in	Log in	N/A, 로그인 버튼

[03][01] 전지UI의 화면기본 구성 및 구현

기존 검사 Aurora에서 사용하던 전지UI가 Aurora2.0에서 통합됨에 따라 관련 메뉴얼을 추가합니다.
기본적인 사용방법은 AuroraPlus에서와 동일하며, 기존의 검사Aurora에서 사용하던 API도 사용가능합니다.

화면 구성

스타일

Aurora 플랫폼은 장비운영 SW UI 일관성 및 구현 편의를 위해, UI 스타일을 기본 제공한다.
메인 화면에 모든 View가 Tab으로 나열되는 기본 UI(TabView Style)를 제공하며, <그림>과 같은 전지UI 공통UI를 제공한다. 화면은 상단부터 MainStatus 영역, Body 영역, StatusBar 영역으로 구분된다.

View 구성시 View Group이 별도 정의되지 않으면 TabView Style(기본 UI)이 적용되고, 그룹 정의가 2개 이상 정의되면 추가 조건에 따라 GroupView Style(전지 검사 표준 UI) 또는 MenuView Style(모듈/공정 장비 UI)이 선택 적용된다.



그림 아이콘이 지정되면 MenuView Style이 적용되고,
그림 아이콘이 지정되지 않으면 GroupView Style이 적용된다.



MenuView Style은 추후 제공될 예정이다.

GroupView Style (전지 검사 표준 UI)

IEquipmentSetup.RegisterViews 에 호출된 순서대로 ViewGroup과 View 가 추가된다.

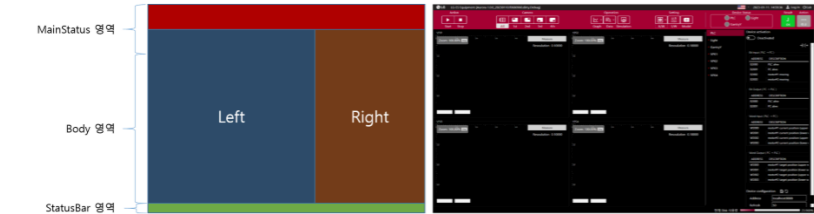
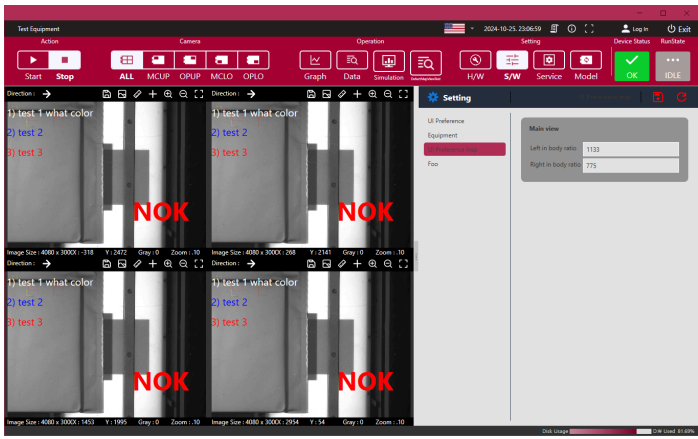


그림. GroupView Style Layout 과 예제

- Left와 Right 영역의 넓이는 App 종료 시 Setting에 저장되며 아래와 같이 직접 입력할 수도 있다.
 - Setting → UI Preference Inap → Main View → Left/Right in body radio
 - 설정값은 재부팅 시 적용된다.



• View 와 RegisterView



• 예제코드

```
manager.RegisterStatusComponent(new Aurora.View.DiskUsage.DiskUsageView());

// Action
MainMenu Action = new MainMenu("Action");
Action.AddSubMenu("Sequence", new Aurora.View.SequenceButtonView(SequenceIds.ProcessSeq),
    place: ViewPlace.ReplaceAtIcon);
manager.RegisterMainMenu(Action);

// Action
List<FrameworkElement> liveviews = new List<FrameworkElement>
{
    new LiveviewUsingArgosVPView(DeviceIds.VP1, CameraIds.CAM1, AuroraEnv.EventObservable),
    new LiveviewUsingArgosVPView(DeviceIds.VP1, CameraIds.CAM2, AuroraEnv.EventObservable),
    new LiveviewUsingArgosVPView(DeviceIds.VP1, CameraIds.CAM3, AuroraEnv.EventObservable),
    new LiveviewUsingArgosVPView(DeviceIds.VP1, CameraIds.CAM4, AuroraEnv.EventObservable)
};

MainMenu Camera = new MainMenu("Camera");
Camera.AddSubMenu("CameraIds.ALL", new Aurora.View.GroupViewLayout.CombineView(liveviews)
{
    Columns = 2,
    Rows = 2,
},
place: ViewPlace.FullInBody,
icon: new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_full.png")));
Camera.AddSubMenu(
    CameraIds.CAM1, liveviews[0], PermissionLevel.LEVEL5, ViewPlace.LeftInBody,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_1.png"))
);
Camera.AddSubMenu(
    CameraIds.CAM2, liveviews[1], PermissionLevel.LEVEL5, ViewPlace.LeftInBody,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_2.png")));
Camera.AddSubMenu(
    CameraIds.CAM3, liveviews[2], PermissionLevel.LEVEL5, ViewPlace.LeftInBody,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_3.png")));
Camera.AddSubMenu(
    CameraIds.CAM4, liveviews[3], PermissionLevel.LEVEL5, ViewPlace.LeftInBody,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_4.png")));
manager.RegisterMainMenu(Camera);

// "Operation"
```

```
MainMenu Operation = new MainMenu("Operation");
Operation.AddSubMenu(
    "Graph", new GraphView(), PermissionLevel.LEVEL5, ViewPlace.Dialog,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_graph.png")),
    5
);
Operation.AddSubMenu(
    "Data", new Aurora.View.DataView(), PermissionLevel.LEVEL5, ViewPlace.Dialog,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_data.png")),
    5
);
Operation.AddSubMenu(
    "Simulation", new SimulationView(), PermissionLevel.LEVEL5, ViewPlace.Dialog,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_simulation.png")),
    5
);
Operation.AddSubMenu(
    "DefectMapViewTest",
    new Aurora.Equipment.View.DefectMapViewTest(),
    icon: new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_data.png")),
    iconMargin: 5);
manager.RegisterMainMenu(Operation);

// "Setting"
MainMenu Setting = new MainMenu("Setting");
var deviceSettingView = new Aurora.View.DeviceSettingView();
Setting.AddSubMenu(
    "H/W", deviceSettingView, PermissionLevel.LEVEL5, ViewPlace.RightInBody,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_hw.png")),
    5
);
Setting.AddSubMenu(
    "S/W", new Aurora.View.EquipSettingView(), PermissionLevel.LEVEL5, ViewPlace.RightInBody,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_sw.png")),
    5
);
var recipeSettingView = new Aurora.View.RecipeSettingView();
Setting.AddSubMenu(
    "Model", recipeSettingView, PermissionLevel.LEVEL5, ViewPlace.RightInBody,
    new BitmapImage(new System.Uri("pack://application:,,,/TestEquipment;component/res/PNG/icon_model.png")),
    5
);
manager.RegisterMainMenu(Setting);

// ""
MainMenu verticalLine = new MainMenu("");
manager.RegisterMainMenu(verticalLine);

// "Device Status"
MainMenu DeviceStatus = new MainMenu("Device Status").AddSubMenu(
    "Judge",
    new Aurora.View.ResultView() { Width = 80 },
    place: ViewPlace.ReplaceAtIcon);
manager.RegisterMainMenu(DeviceStatus);

// "RunState"
MainMenu RunState = new MainMenu("RunState").AddSubMenu(
    "Running",
    new Aurora.View.RunningView() { Width = 80 },
    place: ViewPlace.ReplaceAtIcon);
manager.RegisterMainMenu(RunState);
```

코드. IEquipmentSetup.RegisterViews 구현 예제

MainStatus(GroupView) 개요

GroupView는 전지 검사 표준 UI Base로 구성되어 있다. 그 중에서 MainStatus는 아래와 같이 구성되어 있다.

<그림>과 같이 Main Status 내 각 그룹 별 기능에 대해 설명하고, 해당 기능을 수행할 수 있는 방법에 대해 설명하였다.



그림. 전지 검사 표준 UI

Device Status

- 디바이스의 Activate 여부를 표시한다

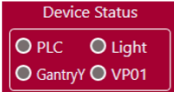


그림. Device Status 아이콘

- 아래 테이블과 같이 설정할 수 있다.

Parameter	예시	설명
그룹 이름	"Device Status"	상단에 보여지는 이름
디바이스 리스트	"PLC", "Light", "GantryY", "VP01"	디바이스 상태를 표기 하고 싶은 디바이스 지정한다
Rows	2	디바이스를 나타낼 수 있는 최대 3줄까지 가능하다
Width	160	디바이스 상태 뷰의 너비
EllipseSize	16.0	아이콘의 크기
EllipseThickness	2.0	선 굵기
지정 위치	ViewPlace.ReplaceAtIcon	해당 뷰 표시하는 위치를 나타내며 기본으로 ReplaceAtIcon 이다

Result

- 검사 결과를 OK/NG 아이콘으로 표시한다.



그림. OK / NG 아이콘

- 결과 는 최소 1개부터 최대 9개까지 동시에 보여줄 수 있다
- 테스트 결과 개수 정하는 방법은 <리스트>와 같다

결과 개수	결과 이미지
1	
2~4	
5~9	

리스트. 테스트 결과 개수 방법

```
Application.Current.AuroraEnv().EventPublisher.Publish(  
    new EventMessage(  
        "TestTotalNo",  
        AuroraSysEvent.REQ_RESULT_TOTAL_NUMBER,  
        4 // 개수 작성 (integer)  
    )  
);
```

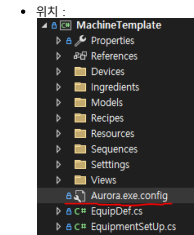
코드. 테스트 결과 개수 표시 예제

```
Application.Current.AuroraEnv().EventPublisher.Publish(  
    new EventMessage(  
        "ReportInspect",  
        AuroraSysEvent.REQ_UPDATE_RESULT_VVIEW,  
        new ResultViewUpdateParam(  
            (param.Param as ActionResult).resultNo,  
            (param.Param as ActionResult).result  
        )));
```

코드. 테스트 결과(OK/NG) 표시 예제

Aurora.exe.config

- Aurora 프로젝트의 옵션을 설정한다.



- 주요 옵션에 대해 알아보자.

```
<startup>
  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
</startup>
<appSettings>
  <add key="THEME_COLOR" value="Dark.Navy" />
  <add key="UseLeftMenu" value="True" />
  <add key="DATA_PATH" value="" />
  <add key="MACHINE_DLL_NAME" value="TestEquipment.DLL" />
  <add key="START_WITH_FULLSCREEN" value="True" />
  <add key="IS_MULTI_APP_ALLOWED" value="False" />
  <add key="IS_MACHINE_APP" value="True" />
  <!-- CI변경을 원할 시 아래 주석 해제 후 경로를 지정 -->
  <!-- <add key="CI" value="D:\LG_Display_logo.png" /> -->
</appSettings>
```

- THEME_COLOR : 네 가지 테마를 지원한다. Dark.Navy, Light.Navy, Dark.LGColor, Light.LGColor.
- IS_INSPECTION_UI : 검사/전지 UI 테마를 사용할지 Default Aurora UI를 사용할 지 결정
- UseLeftMenu : 좌측 서랍메뉴 사용유무
- DATA_PATH : 레시피등의 파일을 저장할 경로를 설정한다.
- MACHINE_DLL_NAME : Aurora는 기본적으로 IEquipmentSetup을 구현한 DLL을 Search한다. Search할 시간을 줄이기 위해 DLL이름을 직접 지정하여 바로 로딩하게 할 수 있다.
- START_WITH_FULLSCREEN : 시작시 전창으로 실행할지의 여부
- IS_MULTI_APP_ALLOWED : Aurora는 기본적으로 두개의 앱이 실행되지 못하게 막는데 이를 해제할 수 있다.
- IS_MACHINE_APP : 비장비 Application으로 제작 시 Recipe관련 등 불필요한 UI요소를 제거할 수 있다.
- CI : 화면 좌측에 표시할 CI의 이미지 경로를 작성한다.



→ 꼭 CI가 아니더라도 이미지를 배치할 수 있다.

Aurora 환경변수

- Aurora의 글로벌 환경변수에 대해 설명한다.

환경변수의 종류

```
namespace Aurora
{
    public static class ApplicationExt
    {
        internal static Mutex AuroraMutex;
        internal static AuroraCore AuroraCore { get; set; }
        internal static Func<string, bool, string, string> Tokens { get; set; }
        public static string DataPath { get; internal set; }
        public static bool UsePermission { get; internal set; }
        public static string CurrUserID { get; internal set; }
        public static ushort PermissionLevel { get; internal set; }
        public static string CurLang { get; internal set; }
        public static bool IsSimulation { get; internal set; }
        public static string DllName { get; internal set; }
        public static bool IsInspUI { get; internal set; }
        public static bool IsServiceApp { get; set; } = false;
    }
}
```

- DataPath : 레시피 저장 Path
- CurrUserId : 현재 로그인된 아이디
- PermissionLevel : 현재 접근 권한 레벨
- CurLang : 현재 설정된 언어
- IsSimulation : 현재 장비가 시뮬레이션 모드인지 확인.
- DllName : 현재 로딩된 장비 DLL의 파일이름
- IsInspUI : 현재 UI 테마가 검사/전지 장비 or Aurora Default인지 확인
- IsServiceApp : 현재 비장비 모드인지 확인

- 간단 사용 예제
몇 가지 예제를 간단히 소개합니다.

- 현재 설정된 언어확인용

```
var lang = ApplicationExt.CurLang;
```

- 특정 기능에서 접근권한에 따른 처리

```
var curPermissionLevel = ApplicationExt.PermissionLevel;
if (curPermissionLevel > (ushort)Aurora.Api.PermissionLevel.LEVEL3) {    // 권한 부족

} else {

}
```

- 디바이스가 시뮬레이션인지 확인

```
public class PumpController : Device
{
}
```

```
public PumpController(ePumpController type)
{
    ...
}

protected override int DoActivate(IDeviceConfig config)
{
    _Config = config as PumpControllerConfig;
    IsSimul = ApplicationExt.IsSimulation | _Config.IsSimulation;
    ...
}
```

Logger

- Aurora 플랫폼은 운영SW 개발 및 장비 운영시 발생하는 사항을 기록하여 관리할 수 있는 Logging 기능을 제공한다.

1. Logger Requirement

- Logger 기능을 사용하기 위해서는 Aurora.exe가 있는 폴더에 hislog2.dll 파일이 존재해야 하며 이는 OOO에서 설정되어있다.

< 필요한 Aurora 고정부 dll 파일 >

PC > 새 볼륨 (D:) > TestEquip > External > Aurora > lib > x64-Debug	x64-Debu		
이름	수정된 날짜	유형	크기
GrabberF.pdb	2024-09-09 오후 2:14	Program Debug ...	19,028KB
GrabberFCLI.dll	2024-09-09 오후 2:14	응용 프로그램 확장	674KB
GrabberFCLI.pdb	2024-09-09 오후 2:14	Program Debug ...	2,228KB
hermes.lib	2024-09-09 오후 2:14	Object File Library	881KB
hermes.pdb	2024-09-09 오후 2:14	Program Debug ...	412KB
hislog2.dll	2024-09-09 오후 2:14	응용 프로그램 확장	5,839KB
hislog2.lib	2024-09-09 오후 2:14	Object File Library	4KB
hislog2.pdb	2024-09-09 오후 2:14	Program Debug ...	26,508KB
MockGrabber.dll	2024-09-09 오후 2:14	응용 프로그램 확장	606KB
MockGrabber.lib	2024-09-09 오후 2:14	Object File Library	2KB

2. Logger Viewer

- 아래 그림1 은 로그의 예를 보여준다.
- Aurora 플랫폼은 C#에서 생성한 로그와 Native 코드(C++)에서 생성한 로그를 통합하여 관리한다.
- Aurora 플랫폼에서는 실시간으로 로그를 조회할 수 있도록 UI를 제공한다. 기본적으로는 실행시 메인창의 제목표시줄에 버튼으로 제공되며, 이를 선택시 로그 창이 그림1과 같은 모양으로 생성된다.
 - 로그UI의 상단부에 로그를 원하는 조건으로 필터링 할 수 있는 컨트롤이 제공된다.

< Aurora 실행 화면: Log 버튼은 중앙 상단에 위치 >

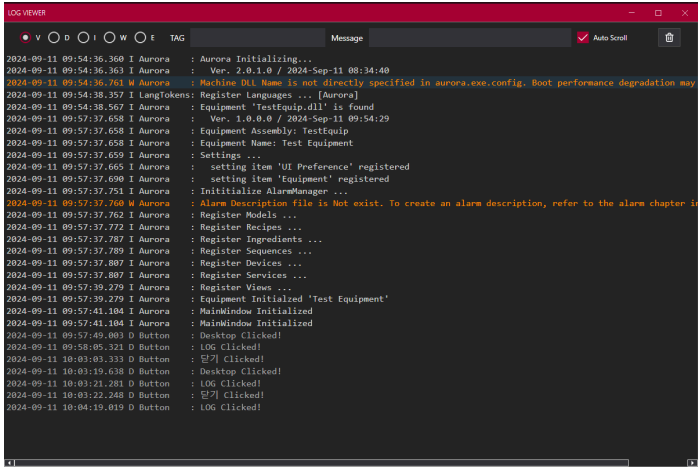


그림 1. Log 예시 : Aurora 플랫폼에서 제공하는 Log Viewer

3. Logger 포맷

- 로그 형식
 - Aurora의 로그는 시간, 로그레벨, 태그, 내용으로 구분된다.
- 시간
 - 로그가 발생한 시간. 자동으로 생성됨
- 로그 레벨
 - 로그의 경중을 나타낸다. 레벨은 로깅시 주어져야 한다. 무분별한 로깅은 추후 필요한 정보를 추출하여 추적하기 어려움을 유발할 수 있으므로 적절한 로그 레벨을 사용할 것을 권장한다. 레벨은 다음과 같으며, 아래로 갈수록 중요도가 높다.
 - [V] Verbose : 사소한 내용
 - [D] Debug : 디버깅을 위한 내용
 - [I] Information : 중요한 상태 변경, 실행 과정 등의 내용
 - [W] Warning : 에러를 유발할 수 있는 주의를 기울여야 하는 내용
 - [E] Error : 에러가 발생한 내용
- 태그
 - 로그 분석시 연관된 로그를 쉽게 찾기 위해 달아 놓는 표기표이다. 로깅시 주어져야 함. 실행시 로그 UI에서 태그로 필터링하여 추출할 수 있다.
- 내용
 - 로깅의 내용. 로깅시 주어져야 한다.

4. Logger API (C#)

- Aurora 플랫폼의 logging 기능을 C#에서 사용하는 예제는 아래 그림을 참고한다.
각 클래스에 로그 태그별로 Logger를 생성할 것을 권장한다.
중복된 로그 태그로도 Logger 생성이 가능하다.

```
using Aurora.Api;

class Foo
{
    // 로거를 생성한다. 선택시 사용할 로그 태그를 입력한다.
    static readonly Logger LOG = new Logger("MyLOG");

    // 로그레벨 별 API
    void method_1()
    {
        LOG.V("verbose log");
        LOG.D("debug log");
        LOG.I("info log");
        LOG.W("warning log");
        LOG.E("error log");
    }

    // 로그의 내용에 자동으로 클래스와 메서드 이름이 prefix로 붙는다.
    void method_2()
    {
        LOG.FuncV("verbose log");
        LOG.FuncD("debug log");
        LOG.FuncI("info log");
        LOG.FuncW("warning log");
        LOG.FuncE("error log");
    }
}
```

5. Logger API (C++)

- Aurora 플랫폼의 logging 기능을 C++에서 사용하는 예제는 아래 그림을 참고한다.
각 파일별로 로그 태그를 정의할 것을 권장한다.
중복된 로그 태그 정의 가능하다.

```
#include <hislog/hislog.h>

// 이 파일에 적용될 로그 태그를 정의한다.
#define LOG_TAG "MyLOG"

// 로그레벨 별 API. 위에 선언된 로그 태그를 사용하여 로깅
void func1()
{
    LOGV("verbose log: %d", 12); // formatting 지원
    LOGD("debug log: %s", "xx");
    LOGI("info log");
    LOGW("warning log");
    LOGE("error log");
}

// 로그의 내용에 자동으로 클래스와 메서드 이름이 prefix로 붙는다.
void func2()
{
    FUNC_LOGV("verbose log: %d", 12); // formatting 지원
    FUNC_LOGD("debug log: %s", "xx");
    FUNC_LOGI("info log");
    FUNC_LOGW("warning log");
    FUNC_LOGE("error log");
}
```

6. 로그 저장

- Aurora 플랫폼에서는 로그를 파일에 저장할 수 있는 기능을 제공한다. 이는 Aurora.exe가 위치한 경로의 hislog.config 파일을 통해서 설정 가능하다.
각각 항목의 설정은 리스트3의 아래 설명을 참고한다.

```
{
  "LogConfigs": [
    {
      "logger_name": "Aurora",
      "collector_mode": "MANUAL",
      "console_out": false,
      "file_out": true,
      "log_save_dir": "D:/LaserScriber/Log/Total",
      "log_save_level": 0,
      "log_file_prefix": "Aurora",
      "log_file_size": 5,
      "log_file_keepdays": 0,
      "socket_port_number_publish": 50109,
      "socket_port_number_logtransfer": 51109,
      "log_save_date": true,
      "log_save_date_size": 100
    }
  ],
}
```

리스트 3. hislog.config파일

- logger_name : 코드상에서 사용할 로거 이름("Aurora"는 지정하지 않아도 되는 디폴트)
- log_save_dir : 로그가 저장될 폴더위치
- log_save_level : 저장되는 로그의 level을 지정(파일에 저장되는 곳에만 적용되며 Log View에서의 Level에 영향을 주지는 않는다.
 - 참고 Log View의 초기 레벨 설정 방법 - 부팅 시점에 아래와 같이 설정한다.

```
HislogViewModel hislogVM = Application.Current.Resources["hislogVM"] as HislogViewModel;
if (hislogVM != null)
    hislogVM.FilterLevel = lge.pri.hislog.LogLevel.W;
```

- log_file_prefix : "Aurora" : Aurora_20230227.log <- 로그파일명 prefix
- log_save_date : 날짜별로 파일자르기 옵션
- log_save_date_size : 날짜별로 자르더라도 지정사이즈초과시 파일구분(1_2...)
- Code에서 사용시
Logger ButtonLogger = new Logger("Button", "System");
-> "Button" Tag명, "System" : logger_name
-> logger_name없을시(new Logger("Button")) 디폴트인 Aurora로거에 기록함.

7. LogSettingView 사용하기

- hislog.config를 Text편집기를 사용하여 수정 시 실수하여 json형식이 깨지거나 잘못된 값을 입력할 수 있다.

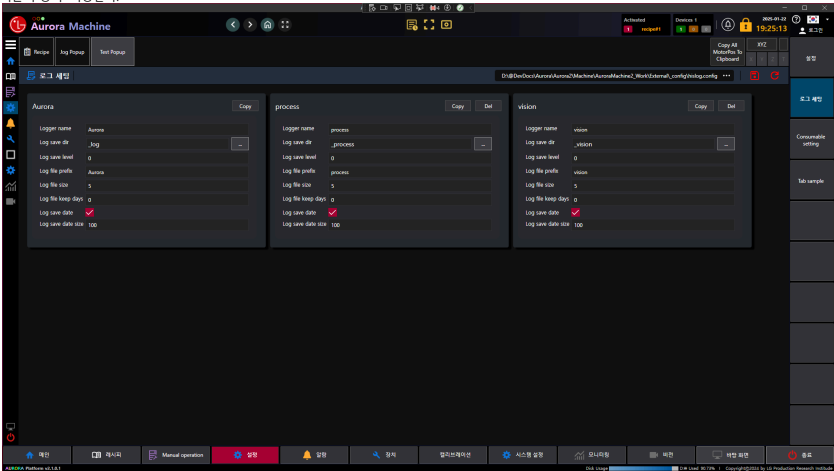
1. 화면추가방법

- a. EquipmentSetup의 RegisterViews() 함수 내에서 아래와 같이 추가한다.

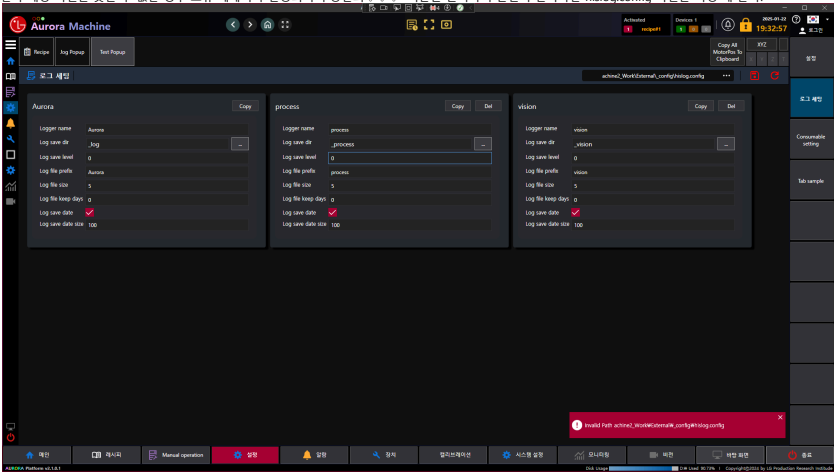
```
var setting = new MainMenu("Setting");
setting.AddSubMenu(ViewIds.LogSetting, new LogSettingView()); // LogSettingView화면을 SubMenu로 등록
manager.RegisterMainMenu(setting);
```

2. 화면 사용방법

- a. hislog.config 파일의 위치를 지정하고 Reload버튼 클릭 시 해당 파일을 읽어서 화면과 같이 표시 해 준다.
b. 이를 수정 후 저장한다.



- c. 만약 해당 파일을 찾을 수 없는 경우 오류 메시지가 발생되며 우상단의  버튼을 클릭하여 본인이 관리하는 hislog.config 파일을 지정 해 준다.



3. 주의사항

- a. 해당 뷰는 hislog.config를 수정하는 기능만 하므로 적용되게 하려면 해당 hislog.config파일을 Aurora.exe가 있는 폴더에 xcopy하는 event를 등록해야 한다.

8. Log Viewer 별도로 화면내에 붙이기

- [16][03] Logger UI를 화면에 붙이기 참고

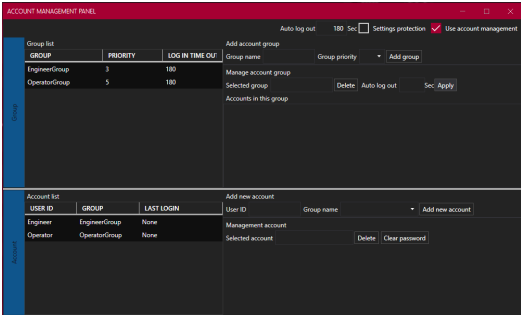
[05] Log-In/Permission 설정

0. Overview

- Log-In 계정에 따라 Permission을 설정하는 법에 대해 설명한다.

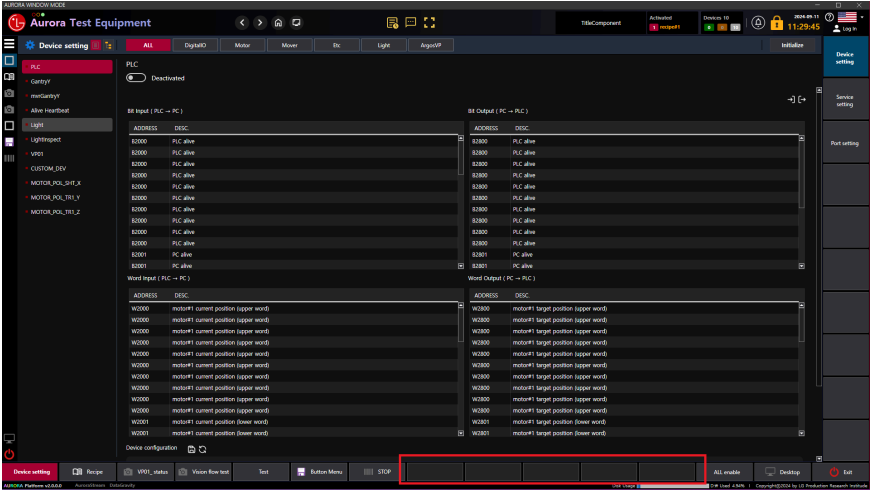
1. Settings - Account 화면

- Permission 설정을 적용여부는 Settings에 Apply permission check를 따른다.
- 디폴트는 uncheck이며, 따라서 전체가 별도의 Permission 없이 동작한다.



- Apply permission을 Check하고 저장후 껴다 커보자.

- 메뉴에 Permission Level에 따라 보이지 않는 화면이 생긴다. 따라서 Log-In을 수행해야한다.



- Log-In
 - Log-In Click 시 아래 그림 1과 같이 로그인 창이 나타난다.
 - 디폴트로 세 개의 계정을 제공한다
 - Admin, Engineer, Operator
 - 최초 세개의 계정은 비밀번호가 없는 상태로, Change Password를 통해 비밀번호를 설정해주어야 한다.

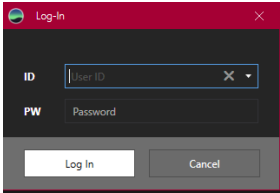


그림 1. Log-In 창

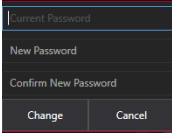
- Permission Level
- 050 ~ 5 까지의 Level을 갖고 있으며 각 화면별로 Level을 지정할 수 있다.

```
public enum PermissionLevel
{
    LEVEL0, // Admin Only
    LEVEL1,
    LEVEL2,
    LEVEL3,
    LEVEL4,
    LEVEL5,
}
```

- 각 기본 계정별 Permission Level

	Level	
Admin	0	가장 낮은 레벨로 모든 기능을 사용할 수 있다.
Engineer	3	Level3, 4, 5에 해당하는 기능을 사용할 수 있다.
Operator	5	Level5에 해당하는 기능만 사용할 수 있다.

- Admin계정
 - 최초 비밀번호가 없으므로 Change Password기능을 통해 암호를 지정한다. (Admin권한가진사람이 지정하도록 하자)



- Management Account
 - Management account메뉴를 누르면 Account와 Group 두가지 메뉴가 존재한다.
 - Admin계정은 Super계정으로 모든 Account에 대해 삭제 및 비밀번호초기화 기능을 갖는다. 또한 계정을 추가할 수 있다.

- 모든 그룹에 대해 삭제제한을 갖으며, Lv 1 ~ 5에 대해 지정되지 않은 레벨의 그룹을 추가할 수 있다.
 - Account ID는 최소 5자 이상 기록해야 "Add new account"버튼이 활성화 된다.

- Engineer 계정
 - 최초 비밀번호가 없으므로 마찬가지로 로그인 수행후 비밀번호를 변경한다.
 - 본인 Priority보다 낮은 그룹의 관리를 수행할 수 있다. 즉 Engineer는 Lv3인 Lv4와 Lv5에 대해 관리를 수행한다.

- Operator계정
 - 가장 낮은 Level 5의 계정으로 별도의 계정이나 그룹을 Manage할 수 없다.

- Permission Level에 따른 화면 표시
 - View를 등록할때 PermissionLevel을 추가지정할 수 있다.
 - Level에 따라 계정과 연동되며 해당 계정의 Level보다 낮은 Level을 갖는 뷰는 화면에 표시되지 않는다.

- 디폴트 (지정하지 않을시)는 Level5이다.

```
public void RegisterViews(IViewManager manager)
{
    var Main = new Aurora.Framework.Managers.MainMenu("Main", MenuIcon.GetMenuIcon(eMenuIcon.HOME), PermissionLevel.LEVEL5, false);
    Main.AddSubMenu("Main", new MainLayout());
    manager.RegisterMainMenu(Main);

    var DeviceSetting = new Aurora.Framework.Managers.MainMenu("DeviceSetting", null, PermissionLevel.LEVEL5);
    DeviceSetting.AddSubMenu("DeviceSetting", new DeviceSettingView());
    manager.RegisterMainMenu(DeviceSetting);

    var Recipe = new Aurora.Framework.Managers.MainMenu("Recipe", MenuIcon.GetMenuIcon(eMenuIcon.RECIPE), PermissionLevel.LEVEL3);
    Recipe.AddSubMenu("Recipe", new RecipeSettingView());
    manager.RegisterMainMenu(Recipe);

    var Settings = new Aurora.Framework.Managers.MainMenu("Settings", MenuIcon.GetMenuIcon(eMenuIcon.SETTING), PermissionLevel.LEVEL1);
    Settings.AddSubMenu("Settings", new EquipSettingView());
    manager.RegisterMainMenu(Settings);

    var LiveView = new Aurora.Framework.Managers.MainMenu("LiveView", MenuIcon.GetMenuIcon(eMenuIcon.CAMERA));
    LiveView.AddSubMenu("LiveView", new LiveViewTest());
    manager.RegisterMainMenu(LiveView);

    var VP01_Status = new Aurora.Framework.Managers.MainMenu("VP01_Status", MenuIcon.GetMenuIcon(eMenuIcon.SAVE));
    VP01_Status.AddSubMenu("VP01_Status", new VPStatusViewTest());
    manager.RegisterMainMenu(VP01_Status);

    var LMC302View = new Aurora.Framework.Managers.MainMenu("LMC302View", MenuIcon.GetMenuIcon(eMenuIcon.BARCODE));
    LMC302View.AddSubMenu("LMC302View", new DevLMC302ConfigView { DataContext = new DevLMC302ConfigViewModel() });
    manager.RegisterMainMenu(LMC302View);

    var LiveSettingView = new Aurora.Framework.Managers.MainMenu("LiveSettingView", MenuIcon.GetMenuIcon(eMenuIcon.SETTING));
    LiveSettingView.AddSubMenu("LiveSettingView", new LiveSettingView
    {
        AnalysisToolOption = true,
        DataContext = new LiveSettingViewModel(
            Application.Current.AuroraEnv().DeviceManager,
            new DeviceGroup()
            {
                Camera = DeviceIds.INSPECT_CAM1,
                Light = DeviceIds.LIGHT_INSPECT,
                LightChannelWhite = 1,
                LightChannelIR = 2,

                MotorX = DeviceIds.MOTOR_POL_SHT_X,
                MotorY = DeviceIds.MOTOR_POL_TR1_Y,
                MotorZ = DeviceIds.MOTOR_POL_TR1_Z,

                ReverseMotorX = false,
                ReverseMotorY = false,
                ReverseMotorZ = false,

                VP = DeviceIds.VP1
            },
            new HowToImageGrab()
            {
                CamPos = new Position()
                {
                    X = 0.0,
                    Y = 0.0,
                    Z = 0.0,
                    T = 0.0,
                },
                LightDimWhite = 1,
                LightDimIR = 2,
                CamExposure = 3,
                CamGain = 4,
            },
            null
        )
    });
    manager.RegisterMainMenu(LiveSettingView);
}
```

- Engineer로 Login : Settings가 보이지 않는다. (권한이 Level 3 보다 작은 경우 제한)
- Operator로 로그인 : Settings, Recipe가 보이지 않는다. (권한이 Level 5 보다 작은 경우 제한)

0. Overview

- 장비 설정을 구현하는 방법에 대해 알아봅니다.
- 각 장비에서 필요한 Setting 을 나눠서 구현할 수 있습니다.

1. Setting 구현

- Aurora.Api.SettingItem 을 상속 받은 Setting class 를 추가하고 Setting Property 를 선언, 이를 장비에 설정과 설정View 를 등록합니다.
- 참고로, 장비 Project 에 "Settings" 폴더를 만들어 관리 하는 방법을 추천합니다.

| 1.1 ISettingItem 상속 Class

- 아래와 같이, ISettingItem 을 상속받은 Class 를 생성하고, 설정에서 사용한 Settings Property 들을 선언해 줍니다.
- 참고로, "PropertyTools.DataAnnotations" 는 다양한 Property Data Annotation 을 위해서 사용합니다.

```
using Aurora.Api;
using PropertyTools.DataAnnotations;

namespace SampleEquipment.Settings {
    class TestSetting : ISettingItem {
        public bool IsSimulateMode { get; set; } = true;
        public int Value { get; set; } = 1000;
        public string SettingName { get; set; } = "KIM_PRI_sw";
        public string Path { get; set; } = "C:\\\\AAA.txt";
        public int FileVersion { get; set; } = 1;
    }
}
```

| 1.2 설정과 설정View 등록

- 장비에서 설정과 설정 View 를 등록해 줍니다.
- 참고 : Device Id 와 마찬가지로, View Id 또한, EquipDefcs 파일에 정의하여 관리하는 것을 추천

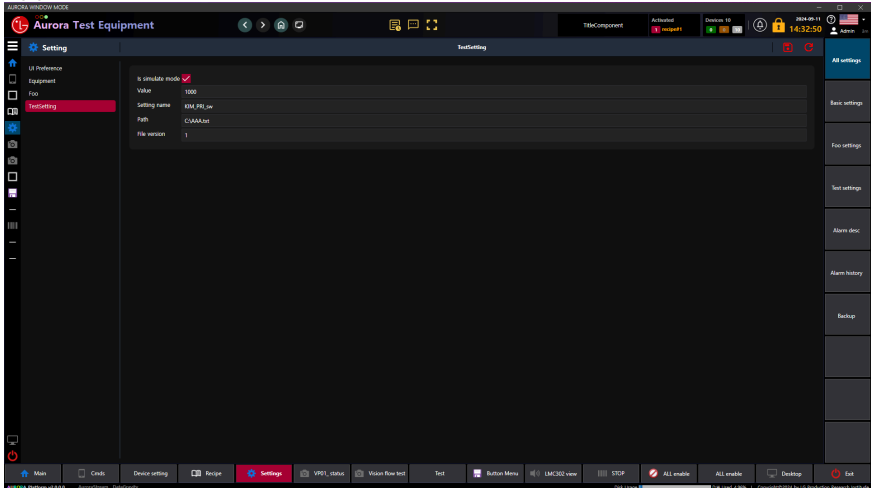
```
class EquipmentSetup : IEquipmentSetup {
    .....

    public void RegisterSettings(SettingRegister register) {
        register(SettingIds.TestSetting, new TestSetting());
        register(SettingIds.System, new SystemSetting());
        register(SettingIds.User, new UserSetting());
    }

    public void RegisterViews(IViewManager manager) {
        // Setting
        var setting = new MainMenu(ViewIds.Setting, MenuIcon.GetMenuIcon(eMenuIcon.SETTING), PermissionLevel.NONE, false);
        setting.AddSubMenu(ViewIds.Setting, new Aurora.View.EquipSettingView());
        manager.RegisterMainMenu(setting);
    }
}
```

| 1.3 실행결과와 화면

- 등록한 "TestSetting" 이 SubMenu 로 생깁니다. (또한, SystemSetting 과 UserSetting 도 등록해 뒀으므로, 생성됨을 확인할 수 있음)
- 왼쪽 목록에 생성한 Setting Class 의 이름이 표시되고, 오른쪽에 Property 가 표시됩니다.
- bool type 은 checkbox, number 나 string 은 editbox 가 자동으로 생성되며, Default 값이 표기됩니다.
또한, 저장 및 Refresh 버튼이 자동으로 생성됨을 확인할 수 있습니다.
- "Equipment" 와 "Account" 는 Aurora 에서 사용하는 기본 설정으로 자동으로 생성됩니다.



1.4 설정 저장 위치

- Setting 값은 "_config_setting" 폴더의 각 파일에서 생성, 저장, 불러오기 가 이루어집니다.
- 파일이 없으면 선언할 때의 Default 값으로 생성하고 Load
파일이 있으면 Load 하게 됩니다.
- 저장 위치는 배포된 Aurora 고정부 config 파일 "Aurora.exe.config" 의 "DATA_PATH" key 의 value 를 따르게 되며, 설정 파일은 json 파일 로 값이 저장됩니다.

```
<configuration>
<startup>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
</startup>
<appSettings>
<add key="THEME_COLOR" value="Light.Navy" />
<add key="DATA_PATH" value="D:\ForSampleEquipment" />
<add key="MACHINE_DLL_NAME" value="TestEquipment.DLL" />
<add key="CI" value="" />
</appSettings>
</configuration>
```

로컬 디스크 (D:) > ForSampleEquipment

ForSampleEquipment 검색

이름	수정된 날짜	유형	크기
Account.json	2023-02-23 오후 5:45	JSON File	1KB
Equipment.json	2023-02-23 오후 5:45	JSON File	1KB
IT_SystemSetting.json	2023-02-23 오후 5:45	JSON File	1KB
TestSetting.json	2023-02-23 오후 5:45	JSON File	1KB
UserSetting.json	2023-02-23 오후 5:45	JSON File	1KB

```
{
  "IsSimulateMode": true,
  "Value": 1000,
  "SettingName": "KIM_PRI_sw",
  "Path": "C:\\AAA.txt",
  "FileVersion": 1
}
```

2. Property Data Annotation

- 다양한 Annotation 이 있습니다.
- Object Property 에 대해서 알아 봅니다. Property 가 특정 Object 인 경우, 2단계까지 표시가 가능합니다.

2.1 Annotation 설명

- 다양한 Annotation 중, 중요한 Annotation 을 살펴봅니다.

```
using Aurora.Api;
using PropertyTools.DataAnnotations;

namespace SampleEquipment.Settings {
    class AnnotationSetting : ISettingItem {
        public bool IsSimulateMode { get; set; } = true;
        public int Value { get; set; } = 1000;
        public SubSettings SubSetting { get; set; } = new SubSettings();

        [DisplayName("Author")]
        public string SettingName { get; set; } = "KIM_PRI_sw";

        [Category("File")]
        [DirectoryPath]
        public string Path { get; set; } = "C:\\AAA.txt";
        [SelectorStyle(SelectorStyle.ComboBox)]
        public SaveImageFormat ImageFormat { get; set; } = SaveImageFormat.PNG;
        [ReadOnly(true)]
        public int FileVersion { get; set; } = 1;

        [Category("Account")]
        public bool UseLocalAccountServer { get; set; } = true;
        [VisibleBy("UseLocalAccountServer", true)]
        public int PortNumber { get; set; } = 12345;
        [VisibleBy("UseLocalAccountServer", false)]
        public string IpAddress { get; set; } = "127.0.0.1";

        [Category("Align")]
        [InputFilePath(".xml")]
        [FilterProperty("__XmlFilter")]
        public string AlignCalibrationFile { get; set; }
        [Browsable(false)]
        public string __XmlFilter { get; private set; } = "Xml files (*.xml)|*.xml|All files (*.*)|*.*";
    }

    public enum SaveImageFormat {
        PNG,
        JPG,
        BMP
    }

    class SubSettings {
        public double Item1 { get; set; }
        public double Item2 { get; set; }
    }
}
```

```
}
}
```

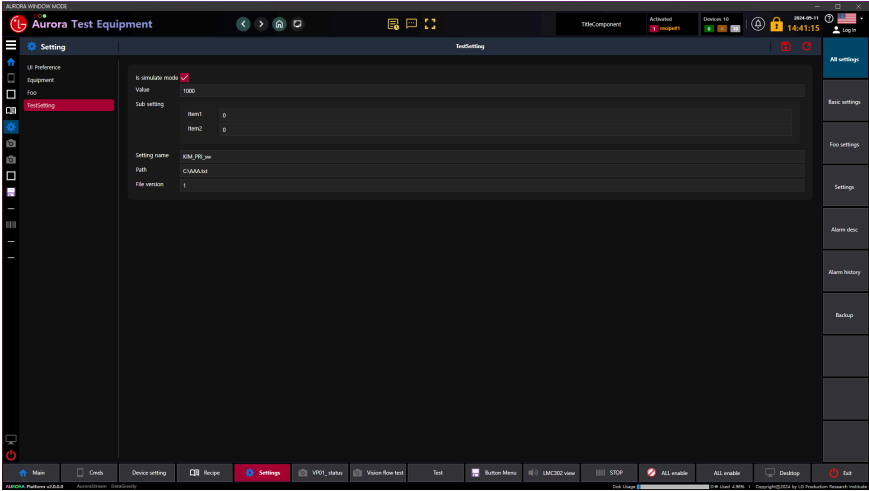
Annotation 종류	설명
DisplayName	Property를 화면에 어떤 이름으로 표시할지를 지정
ReadOnly	편집 불가
Category	화면을 Category화 하는데 사용
DirectoryPath	편집Edit에 버튼을 달아 폴더를 선택할 수 있도록 해준다.
SelectorStyle	enum등 여러항목중 선택할 수 있는 항목에 대해 어떤 Style로 표시할지 선택한다. (Radio, Combo, List)
VisibleBy	특정 Property 값에 따라 화면에 보일지 말지를 결정해준다.
InputFilePath	File Dialog를 열어준다. ".xml" 어떤 파일을 선택할지..
FilterProperty	File Filter를 지정해줄 Property를 지정한다.
Browsable	Property Grid에서 보여주지 않을 Property.

| 2.2 Object Property

Property 가 특정 Object 인 경우, 2단계까지 표시가 가능합니다.
(그 이상의 표현이 필요할 경우 Custom View를 만들어서 사용하는 것을 권장함)
위의 Code 와 실행 결과에서 "SubSetting" 의 이름을 갖는 Class 를 사용한 Property 를 확인 바랍니다.

```
namespace AuroraSample.Setting
{
    class TestSetting : ISettingItem
    {
        public bool IsSimulateMode { get; set; } = true;
        public int Value { get; set; } = 1000;
        public SubSetting SubSetting { get; set; } = new SubSetting();
        public string SettingName { get; set; } = "KIM_PRI_sw";
        public string Path { get; set; } = "C:\\\\AAA.txt";
        public int FileVersion { get; set; } = 1;
    }

    class SubSetting
    {
        public int Item1 { get; set; }
        public int Item2 { get; set; }
    }
}
```



3. Setting 항목 사용

- 위에서 설정된 Property 들은 AuroraEnv 의 SettingEntry 를 통해서 조회할 수 있습니다.
- 또한, 저장을 위해서는 명시적으로 Save 를 해주어야 합니다.

| 3.1 Setting 항목 조회

- SettingEntry 를 통한 조회 방법의 예시 입니다.
(주의 : SettingEntry에서 조회할 경우 Json파일로부터 읽어오기 때문에 시퀀스 내 반복 작업 등에서 읽지 않도록 함)

```
public void OnStartup()
{
    var setting = AuroraEnv.SettingEntry[SettingIds.TestSetting] as TestSetting;
```

```
Log.I($"Setting : {setting.Value}");
}
```

3.2 Setting 항목 Save

- 항목의 값을 저장 위해서는 명시적으로 Save 를 해주어야 합니다.
- SettingEntry 를 통한 Save 방법의 예시 입니다.

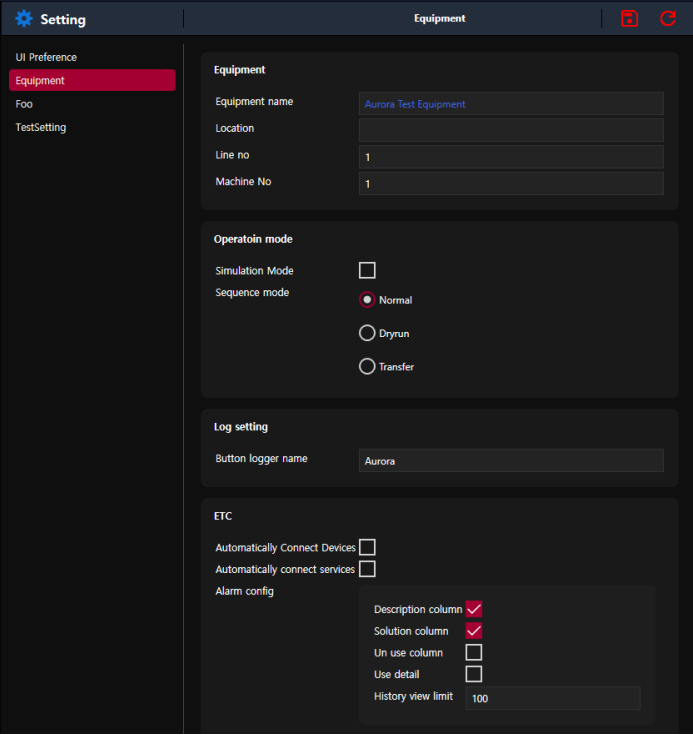
```
public void OnStartup()
{
    var setting = AuroraEnv.SettingEntry[SettingIds.TestSetting] as TestSetting;
    Log.I($"Setting : {setting.Value}");

    setting.SettingName = "Change Setting Name";

    AuroraEnv.SettingEntry.Save(SettingIds.TestSetting, setting, string.Empty);
}
```

4. Aurora System Settings

- Equipment : Setting Id (AuroraSettings.Equipment)

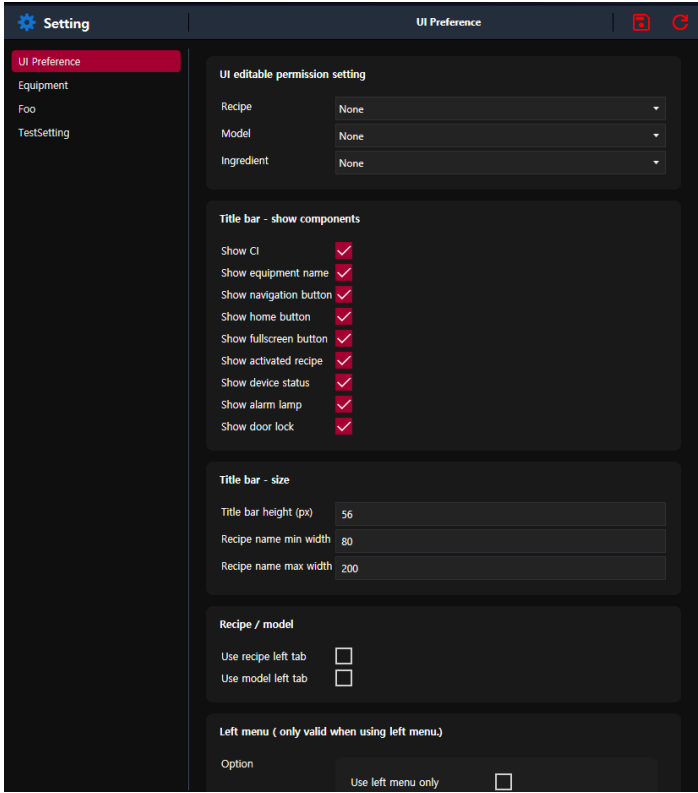


- 장비명, 위치, 장비번호
- 시뮬레이션 모드, 시퀀스 모드
- 로그 설정 : 버튼 자동 로깅할 로그 이름 설정, 사용하지 않을 경우 비워둔다.
- 디바이스 자동 연결 : 부팅후 자동 연결할지 설정 (실제 자동연결은 OnStartup에서 설정값보고 처리해주어야 한다.)
- Alarm Config : 알람 칼럼 사용/미사용 설정, 알람 히스토리 뷰어에서 보일 최대 갯수 설정

- Account : Account Mangement 상단바로 이동(Aurora Plus에서는 Setting에서 관리)
권한 : 권한적용시 로그인을 필요로 함.
Auto Logout : 권한적용된 상태에서 로그인후 UserAction이 없으면 자동로그아웃 되는 시간으로 '분'단위로 설정한다.



- UI Preference
Aurora 화면 설정을 할 수 있는 화면이다.



- 1) Use recipe left tab : 레시피뷰어에서 탭은 기본 탭에 배치되나 왼쪽에 배치하고 싶은 경우
2) Use model left tab : 모델 뷰어에서 탭은 기본 탭에 배치되나 왼쪽에 배치하고 싶은 경우
3) Left menu : 좌측 서랍 메뉴 설정
4) Toast notification : 토스트 알림 UI설정
5) Full screen UI : 특정 메뉴의 화면을 전채로 확대시에 대한 설정 (타이틀바 메뉴등이 사라지고 전체화시키는 기능)

[07] Recipe

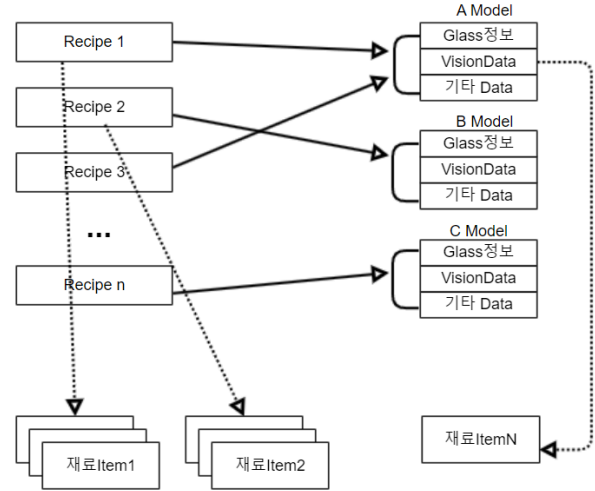
0. Overview

레시피(Recipe)와 모델(Model), 재료(Ingredients), 시스템설정(Settings)

- 시스템 설정
 - 보통, 동일PC내에서 하나의 값을 가지는 값, 또는 하나의 장비에 하나의 값을 가지는 값, 또는 리스트 형태의 값.
- 모델
 - 장비에 투입되는 자재를 말하며, 자재특성에 의해 영향을 받는 조건
 - 20인치 자재와 30인치 자재가 있는 경우 : 각 자재의 Size가 달라지면서 바뀌는 값, Mark위치가 달라져 변경되어야 하는 비전 데이터등
- 레시피
 - 각각 장치들의 설정값들을 공정조건을 변경하며 튜닝하는 값들
 - ex) 레이저 파워, 압력값, 속도, ...
 - 장비의 핵심 Data로 어떤 레시피로 가공하느냐에 따라 Output을 결정짓는다.
- Ingredient
 - "재료"라는 의미로, 레시피 또는 모델등에서 사용할 재료를 정의하는데, 특히 여러개의 목록으로 관리되어야 할 재료를 의미한다.
- 예) 삼계탕
 - 모델 : 요리할 할 대상을 의미 → 닭 (닭 7호 - 닭 크기 / 냄비 크기...)
 - 레시피 : 삼계탕을 끓이는 방법에 대한 정의로, 끓이는 방법은 다양하다 (물의 양, 끓이는 시간, 소금의 양, ...)
 - 재료 : 다양한 재료로 하나의 재료가 여러가지로 정의될때를 의미한다. (황기-중국산, 한국산, 영동황기,... 만능육수-닭으로만든육수,돼지고기로만든육수...)
- 예) 잉크젯 분사 장비
 - 모델 : 잉크젯을 분사할 대상 → Glass
 - 레시피 : 분사하는 방법에 대한 정의, 파워, 속도, 위치...
 - 재료 : 잉크의 종류, 노즐의 종류
 - 장비는 레시피로 가공하며, 하나의 레시피는 가공할 대상인 Glass(투입대상이며, 모델)에다가 어떤 잉크를 뿌릴지 (잉크의목록 : 재료), 어떤 노즐을 쓸지(노즐의목록:재료)를 선택한후 가공속도와 분사파워, 가공할 위치를 가질 수 있다.

레시피와 모델, 재료의 관계

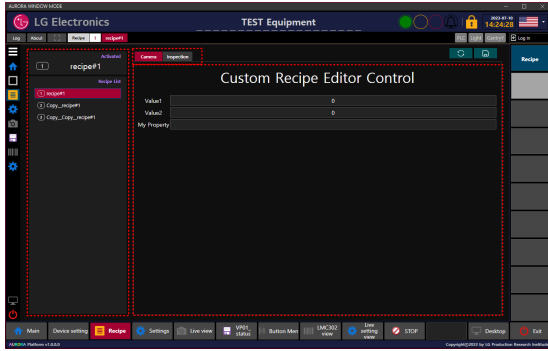
- 공정 레시피는 본인이 사용할 모델데이터를 선택하여 사용한다.



Recipe

- 공정 레시피(or 가공레시피)라고 부르며 이에 대해 설명한다.

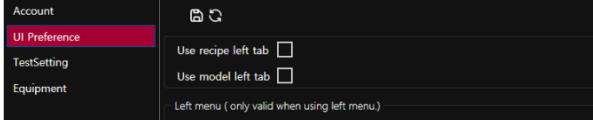
1. 기본 UI



- Recipe List : 생성된 레시피 목록
- Activated : 현재 활성화된 레시피아이디와 이름
- Tab : 레시피 항목 목록(Camera / Inspection ...)

- 레시피 항목이 많은 경우는 Tab을 좌측으로 배치할 수 있다.

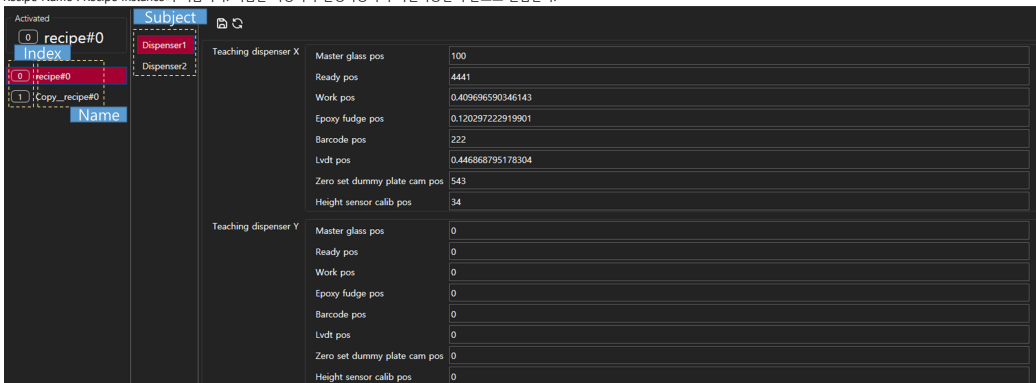
설정 > UI Preference > Use recipe left tab



2. 세부구성

1. Recipe 화면이 아닌 다른 화면에서 Recipe를 읽고 저장할 일이 있다.
2. Recipe는 Setting과 달리 Instance를 여러 개 만들 수 있으므로 헷갈릴 때가 있으니, 먼저 Recipe를 사용하면서 접하는 용어를 이해하자.
 - a. Recipe Entry : Recipe Instance 하나를 가리킨다. Recipe는 여러 개의 Instance로 관리된다. Entry는 id와 name이 있다.

- i. Recipe Id : Recipe Instance의 Index이다. 해당 Index는 MES등 상위시스템과 연동시 활용될 수 있다.
- ii. Recipe Name : Recipe Instance의 이름이다. 이름은 사용자가 변경가능하며 식별가능한 수준으로 편집한다.



- iii. subject : Recipe를 등록할 때 사용한 ID가 subject이다. (아래 코드의 "Dispenser1"과 "Dispenser2")

```
public void RegisterRecipes(RecipeRegister register)
{
    register("Dispenser1", new ModelDp());
    register("Dispenser2", new ModelDp());
}
```

3. RecipeManager에서 다음과 같이 처리한다.

- a. 현재 액티브된 Entry에서 원하는 Subject에 접근하기

```
ModelDp dp1 = Application.Current.AuroraEnv().RecipeManager["Dispenser1"] as ModelDp;
```

- b. Recipe의 값을 업데이트 하였을 때 이를 저장하기

현재 액티브된 Entry에 저장하는 것을 가정(그렇지 않는 경우는 없을 것으로 예상됨)

- i. 액티브된 Entry의 id, name을 알아낸다.

- ii. RecipeManager.SaveRecipeItem ((id, name), subject, IRecipeItem) 함수 호출

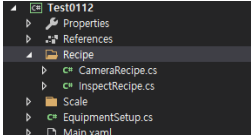
- iii 변경 시에는 Item Changed 이벤트를 날린다.

```
RecipeManager.SaveRecipeItem((recipeInfo.Id, recipeInfo.Name), RecipeIds.Inspection, Inspection);
RecipeManager.SaveRecipeItem(RecipeIds.Inspection);
```

2. 구현

| 2.1 Recipe 생성

- Recipe 폴더 생성 → 솔루션 (Test0112) 우클릭 - 추가 - 새 폴더 Recipe 생성



- IRecipeItem을 상속받은 Class를 생성. → Recipe 폴더 우클릭 - 추가 - 클래스 InspectRecipe.cs

```
using Aurora.Api;
using PropertyTools.DataAnnotations;

namespace Test0112.Recipe
{
    class InspectRecipe: IRecipeItem
    {
    }
}
```

- Recipe Item들을 Property로 등록.

```
using Aurora.Api;
using PropertyTools.DataAnnotations;

namespace Test0112.Recipe
{
    class InspectRecipe : IRecipeItem
    {
        public int Spec1 { get; set; }
        public int Spec2 { get; set; }
    }
}
```

주의 1 : 수치 데이터를 string으로 사용하지 말 것. 시간은 int(ms), 거리, 무게 등은 double로 속성에 맞는 타입을 지정하자.

주의 2 : 개발 중간에 Recipe Item의 Type이 변경될 경우 기존 Recipe 로드 시 프로그램 Exception발생. 기존에 저장된 데이터를 Json파일에서 삭제 혹은 적절한 형변환 후 사용 가능하다.

주의 3 : 개발 중간에 Recipe Item의 이름이 변경될 경우 기존 값을 사용 못함. 백업 후 Json파일 상에서 수동 이전이 필요하다.

- 장비Class에서 Recipe를 등록하고, 디폴트 레시피뷰를 추가.

- Main View 등록 부분은 [01] View 생성 및 등록 을 참고 후 작성해도 좋다

```
using Test0118.Recipe; // 네임스페이스도 추가 필요

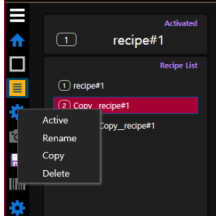
public void RegisterRecipes(RecipeRegister register)
{
    register(
        "Inspection",
        new InspectRecipe()
    );
}

public void RegisterViews(IViewManager manager)
{
    manager.RegisterView("Main", new Main());
    manager.RegisterView("Recipe", new Aurora.View.UCRecipe());
    manager.RegisterView("Setting", new Aurora.View.EquipSettingView());
    manager.RegisterView("Device", new Aurora.View.DeviceSettingView(), PermissionLevel.LEVEL3);
}
```

- Object Property와 Data Annotation은 설정을 참고.

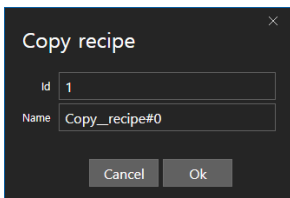
| 2.2 Context Menu

- 레시피는 마우스 우클릭을 통해 Context Menu를 제공.



→ Recipe가 하나만 있을때는 다른 동작을 수행할 수 없다.

| 2.3 Copy

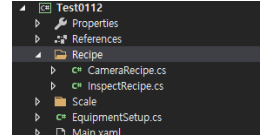


- 복사된 레시피의 Context
→ Remove는 Activate이 되지 않은 Recipe만 삭제할 수 있다!!

| 2.4 추가 구현

- 레시피 안에 여러가지 Sub항목들을 추가해보자.

1) Recipe 폴더에 CameraRecipe Class를 추가하였다.



```
class CameraRecipe : IRecipeItem
{
    public int Value1 { get; set; }
    public int Value2 { get; set; }
    public string MyProperty { get; set; }
}
```

- 장비 Class내에 Recipe도 추가 등록하자.

```
public void RegisterRecipes(RecipeRegister register)
{
    register(
        "Inspection",
        new InspectRecipe()
    );
    register(
        "Camera",
        new CameraRecipe()
    );
}
```

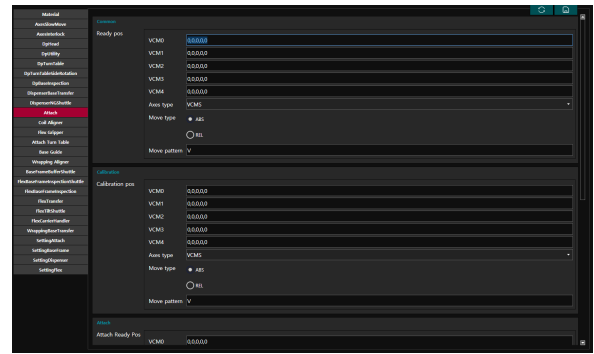
- 3) Recipe등록에서 CustomView까지 파라미터로 넣어주자.

```
void IEquipmentSetup.RegisterRecipes(RecipeRegister register)
{
    register(
        "Camera",
        new CameraRecipe(),
        (recipe, item) => new CameraRecipeEditor(item)
    );
    ...
}
```

Recipe View

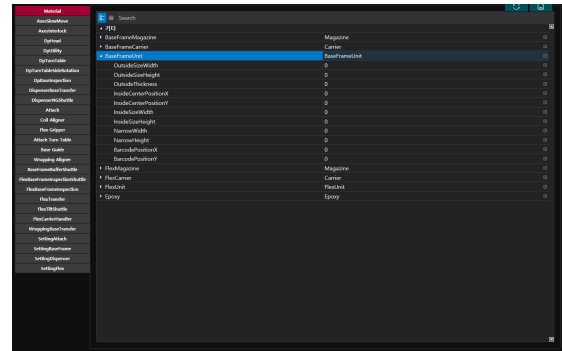
- 기본 PropertyGrid사용하기
세번째 인자로 아무것도 넣지 않는다.

```
public void RegisterRecipes(RecipeRegister register) {
    register(ModelSubjects.Attach, new ModelAttach());
}
```



- Dev Property Grid View사용하기
세번째 인자에 UCDevPropertyGrid를 넣어준다.

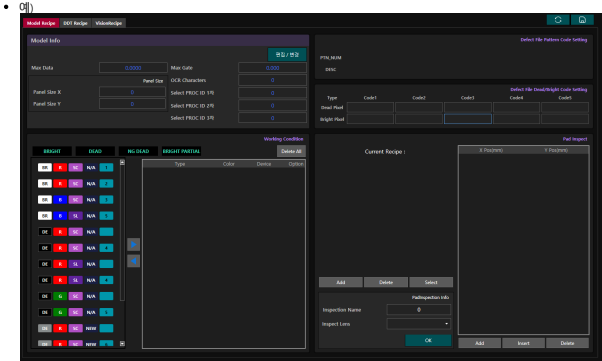
```
public void RegisterRecipes(RecipeRegister register) {
    register(ModelSubjects.Material, new MaterialRecipe(), (id, item) => new UCDevPropertyGrid());
}
```



- 1) Tree구조로 제공되며 디폴트로 접힌 상태이다.
- 2) 속성이 많아 성능에 문제 발생시에 특히 이 Control을 사용하는것이 좋겠다.

Recipe View 3가지

- 기본 PropertyGrid, Dev Property Grid, Custom View 중 Custom View 제작 및 사용법에 대한 설명을 다룬다.

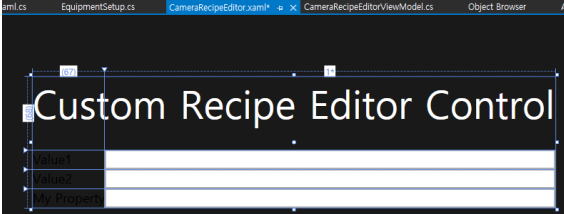


| 1.1 View / ViewModel / Register

- Sample Recipe

```
public class CameraRecipe : IRecipeItem
{
    public int Value1 { get; set; }
    public int Value2 { get; set; }
    public string MyProperty { get; set; }
    public string CameraIngredient { get; set; }
}
```

- Recipe Data를 표현할 UserControl(WPF)을 추가한다. (아래 예시는 CameraRecipeEditor라는 UserControl 생성)



- Sample Xaml : CameraRecipeEditor.xaml

```
<UserControl
    x:Class="Aurora.Equipment.Recipe.CameraRecipeEditor"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Aurora.Equipment.Recipe"
    d:DataContext="{d:DesignInstance Type=local:CameraRecipeEditorViewModel}"
    mc:Ignorable="d">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <TextBlock
            Grid.ColumnSpan="2"
            Margin="0,0,0,20"
            FontSize="36" Foreground="White"
            Text="Custom Recipe Editor Control"/>
        <TextBlock
            Grid.Row="1"
            Grid.Column="0"
            Text="{Binding Recipe.Value1}"/>
        <TextBlock
            Grid.Row="1"
            Grid.Column="1"
            Text="{Binding Recipe.Value2}"/>
        <TextBlock
            Grid.Row="2"
            Grid.Column="0"
            Text="{Binding Recipe.Value2}"/>
        <TextBlock
            Grid.Row="3"
            Grid.Column="0"
            Text="My Property"/>
        <TextBlock
            Grid.Row="3"
            Grid.Column="1"
            Text="{Binding Recipe.MyProperty}"/>
    </Grid>
</UserControl>
```

- UserControl의 ViewModel을 생성한다. 이때 ViewModel은 IRecipeViewModel을 상속하여 구현하여, 화면에 표시할 레시피가 바뀌었을 때 Platform이 알려줄 수 있도록 한다.

```
public interface IRecipeViewModel
{
    /// 현재 뷰어의 레시피가 변경되었음을 알림.
    /// 뷰모델에서 해당 레시피로 갱신해주어야 한다.
    void OnChanged((int id, string name) recipe);

    /// 현재 뷰어의 레시피가 리로드되었음을 알림.
    void Reloaded((int id, string name) recipe);

    /// 현재 편집중인 화면에서 모델이 변경되었고, 모델이 변경되었을때 레시피화면에
    /// 모델정보를 표시하고 있다면, 다른 데이터가 영향을 받는 경우 처리요청
    void OnModelSelectChanged(string modelName);

    /// Save되기 전에 호출
    /// false를 리턴하면 저장되지 않는다.
    bool PreSaved(int id, string recipeName);

    /// Save되고나서 전에 호출
    void OnSaved(int id, string recipeName);
}
```

```
class CameraRecipeEditorViewModel : NotifyPropertyChangedBase, IRecipeViewModel
{
    private CameraRecipe _Recipe;
    private readonly IRecipeManager _Manager;
    private (int id, string name) _RecipeId;
    public CameraRecipe Recipe
    {
        get => _Recipe;
        private set => SetProperty(ref _Recipe, value);
    }

    public CameraRecipeEditorViewModel(IAuroraEnvironment auroraEnv, (int, string) recipe, IRecipeItem recipeItem)
    {
        _RecipeId = recipe;
        _Manager = auroraEnv.RecipeManager;
        Recipe = recipeItem as CameraRecipe;
    }

    public void OnChanged((int id, string name) recipe)
    {
        Recipe = _Manager.RecipeItem(recipe.id, "Camera") as CameraRecipe;
    }

    public void Reloaded((int id, string name) recipe)
    {
        Recipe = _Manager.RecipeItem(recipe.id, "Camera") as CameraRecipe;
    }

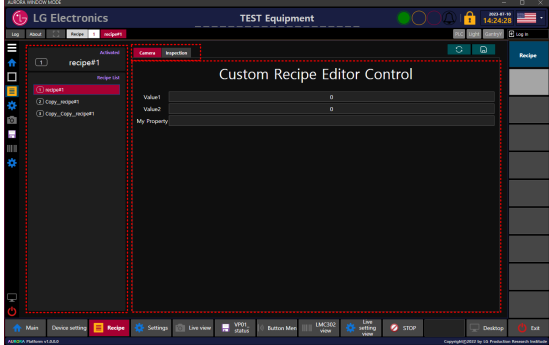
    public void OnModelSelectChanged(string modelName)
    {
    }
}
```

- CustomView를 RecipeRegister에서 등록해준다.

```
void RegisterRecipes(RecipeRegister register)
{
    register("Camera", new CameraRecipe(), (recipe, item) => new CameraRecipeEditor(AuroraEnv, recipe, item) {
        DataContext = new CameraRecipeEditorViewModel(AuroraEnv, recipe, item)
    });

    register(
        "Inspection",
        new InspectRecipe()
    );
}
```

- CameraRecipeEditor는 고정부에서 호출될때 Recipe(id, name)와 Item(RecipeItem)을 가져올 수 있는데 DataContext로 활용할 Item을 생성자에 넣어주었다.



Recipe사용하기

- RecipeManager를 통해 수행한다.

```
public interface IRecipeEntry
{
    IList<(int id, string name)> Recipes(); // 레시피 목록 조회
    IList<string> Subjects(); // 레시피 항목의 subject 목록 조회

    (int id, string name) ActivatedRecipe(); // 활성화 레시피의 ID와 이름 조회

    int ActivateRecipe( // 특정 레시피를 활성화 함
        int id,
        string activator
    );

    Dictionary<string, IRecipeItem> ActivatedRecipeItems(); // 활성화 된 레시피 항목들을 조회

    IRecipeItem ActivatedRecipeItem( // 활성화 레시피의 특정 subject를 가지는 항목을 조회
        string subject
    );

    IRecipeItem RecipeItem( // 특정 레시피의 특정 특정 subject를 가지는 항목을 조회
        int id,

```

```
        string subject
    );

    IRecipeItem this[string subject] { get; } // 활성화 레시피의 특정 subject를 가지는 항목을 조회
}
```

→ Recipe는 여러개가 될 수 있으므로, ActivatedRecipe를 사용하여 접근하는 것이 좋겠다.

```
var insRecipe = AuroraEnv.RecipeManager["Inspection"] as InspectRecipe;
Console.WriteLine(insRecipe.Spec1);

var activatedRecipe = AuroraEnv.RecipeManager.ActivatedRecipe();
Console.WriteLine(activatedRecipe.id + ", " + activatedRecipe.name);

var activateInsRecipe = AuroraEnv.RecipeManager.ActivatedRecipeItem("Inspection") as InspectRecipe;
Console.WriteLine(activateInsRecipe.Spec1);

var activateRecipes = AuroraEnv.RecipeManager.ActivatedRecipeItems();
foreach (var recipeItem in activateRecipes) {
    if (recipeItem.Value is InspectRecipe) {

    } else if (recipeItem.Value is CameraRecipe) {

    }
}
```

Recipe View 외의 화면에서 Recipe 접근 시 주의사항

- Recipe View를 통해 Recipe값을 변경할 수 있지만 간혹 일부 Recipe의 경우 다른 화면(ex. Manual 조작)에서도 입력할 수 있도록 한다.
이 때 해당 화면 진입 시 Recipe값을 다시 읽어오는 코드를 반드시 추가하도록 하자.(VisibleChanged는 12장 UI-View의 Visibility Change를 참고하자.)

```
private void VisibleChanged(object obj)
{
    bool visible = (bool)obj;
    if (visible) {
        var insRecipe = AuroraEnv.RecipeManager["Inspection"] as InspectRecipe;
        LoadInspectValue();
    } else {

    }
}
```

[08] Model/Ingredient

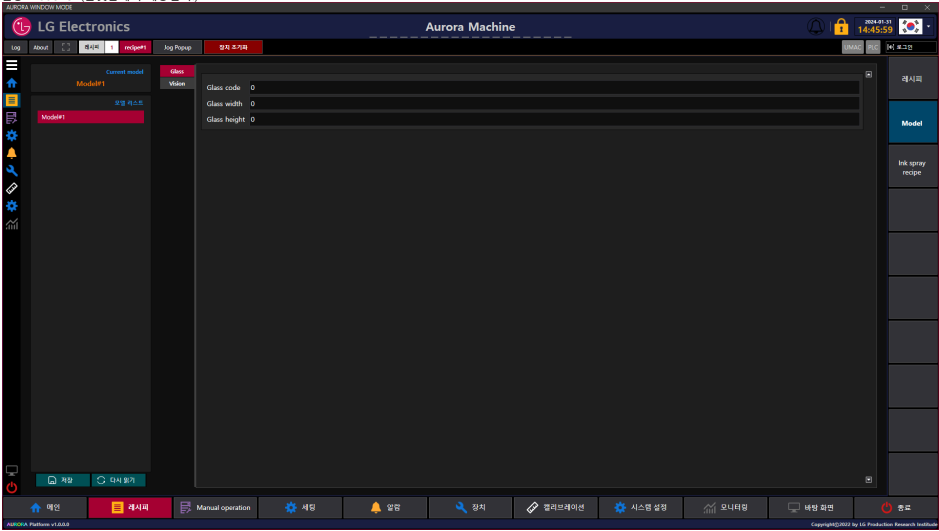
Model

0. Overview

- Model사용법에 대해 알아본다.
- Model은 레시피에서 사용할 모델정보를 저장한다.
- 모델에 따라서 변화하는 데이터도 함께 모델Data로 정의한다.
- 모델과 레시피는 수평관계이다. 여러 레시피는 각각 자기가 사용할 모델을 선택할 수 있다.
- 모델만 존재하는 프로젝트의 경우에는 모델=레시피이다. 즉 레시피를 사용해야한다.

1. 기본 UI

- 전장 Model UI (플랫폼에서 제공한다.)



2. 구현

| 2.1 Model생성

- IModelItem을 상속받은 Class로 정의한다.

```
class GlassModel : IModelItem
{
    public int Width { get; set; }
    public int Height { get; set; }
}

class VisionModel : IModelItem
```

```
{
    public double MaxData { get; set; }
    public double MaxGate { get; set; }
    public int OCRCharacters { get; set; }
}
```

| 2.2. Model 등록

- EquipmentSetup에서 Model을 Register해준다.

```
public void RegisterModels(ModelRegister register)
{
    register(
        "Glass",
        new GlassModel()
    );
    register(
        "Inspection",
        new VisionModel()
    );
}
```

| 2.3 Model Viewer등록

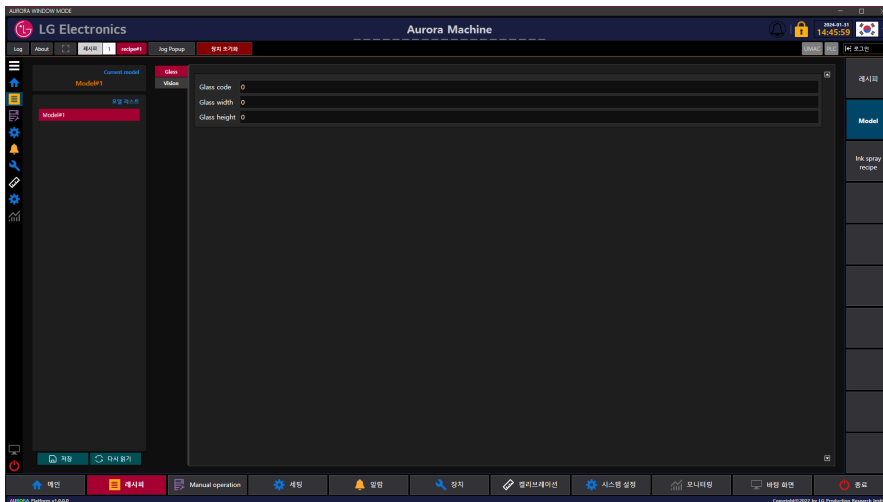
- 모델 뷰어를 등록한다.

```
var Recipe = new MainMenu("Recipe", MenuIcon.GetMenuIcon(eMenuIcon.RECIPE));
Recipe.AddSubMenu("Recipe", new URecipe()); // new URecipe();
Recipe.AddSubMenu("Model", new UModel());
```

- 전창모델 시나리오

→ 기본적으로는 현재 Activated된 Recipe가 Current Model이 된다. 없으면 **None으로 표기

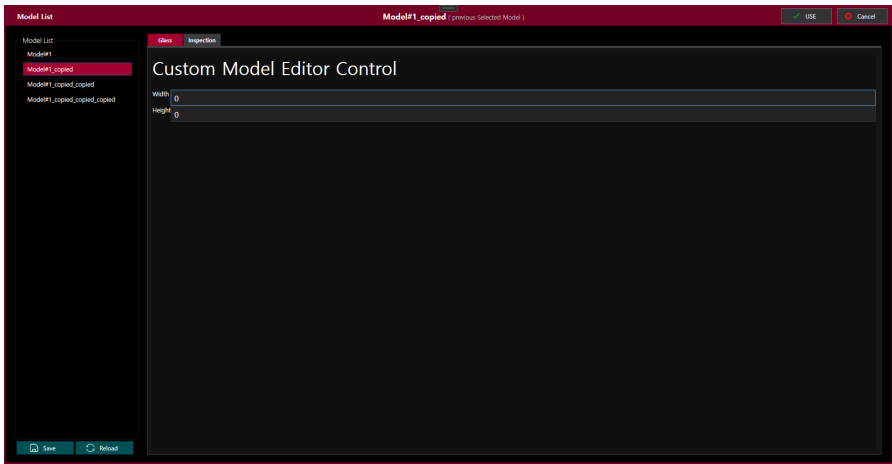
→ SYS_REQ_MODEL_EDITING_CHANGE이벤트를 수신한 이후에 첫 진입하면 해당 이벤트로 발송된 모델을 표시한다.



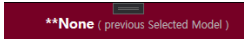
- Model Dialog

→ 직접 모델 Dialog를 띄우고 싶을때는 아래 코드를 사용한다.

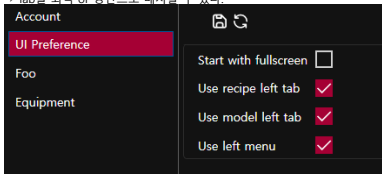
```
var win = new WinModelList(selectedModel: ModelName);
win.Owner = Application.Current.MainWindow;
var res = win.ShowDialog();
if ((bool)res)
{
    if (_modelName != win.SelectedModel)
    {
        ModelName = win.SelectedModel;
    }
}
```



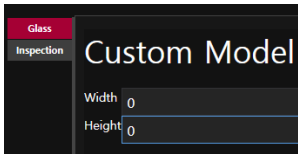
→ 선택한 모델이 없을 경우는 **Model로 표기한다.



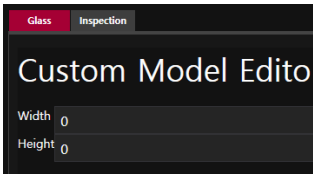
- UI Option
 - Tab을 좌측 or 상단으로 배치할 수 있다.



→ 좌측배치 vs 상단배치



vs



| 2.4 Model Custom Viewier

- 위 GlassModel을 Custom하게 표현하는 UserControl을 생성하여 등록하기
- GlassModelEditor작성 (Xaml)

```
<UserControl
  x:Class="Aurora.Equipment.Model.GlassModelEditor"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:local="clr-namespace:Aurora.Equipment.Model" d:DataContext="{d:DesignInstance Type=local:GlassModelEditorViewModel}"
  mc:Ignorable="d" Background="#121212">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="Auto"/>
      <ColumnDefinition Width="*/>
    </Grid.ColumnDefinitions>

    <TextBlock
      Grid.ColumnSpan="2"
      Margin="0,0,0,20"
      FontSize="36"
      Text="Custom Model Editor Control"/>

    <TextBlock
      Grid.Row="1"
      Grid.Column="0"
      Text="Width"/>
    <TextBox
      Grid.Row="1"
      Grid.Column="1"
      Text="{Binding Model.Width}"/>

    <TextBlock
      Grid.Row="2"
      Grid.Column="0"
      Text="Height"/>
    <TextBox
      Grid.Row="2"
      Grid.Column="1"
      Text="{Binding Model.Height}"/>

  </Grid>
</UserControl>
```

- ViewModel 생성
 - IModelViewModel을 상속받는다.

```
class GlassModelEditorViewModel : NotifyPropertyChangedBase, IModelViewModel
{
  private GlassModel _Model;
  private readonly IModelManager _Manager;

  public GlassModel Model
  {
    get => _Model;
  }
}
```

```

        private set => SetProperty(ref _Model, value);
    }

    public GlassModelEditorViewModel(IModelManager manager, IModelItem item)
    {
        _Manager = manager;
        Model = item as GlassModel;
    }

    public void OnChanged(string modelName)
    {
        Model = _Manager.ModelItem(modelName, "Glass") as GlassModel;
    }

    public void Reloaded(string modelName)
    {
        Model = _Manager.ModelItem(modelName, "Glass") as GlassModel;
    }
}

```

- 모델 등록 : 두번째 인자는 모델편집기 Custom UI이다. 지정해주지 않으면 디폴트로 PropertyGrid가 연결된다.

```

public void RegisterModels(ModelRegister register)
{
    register(
        "Glass",
        new GlassModel(),
        (model, item) => new GlassModelEditor(AuroraEnv, item)
    );
    register(
        "Inspection",
        new VisionModel()
    );
}

```

| 2.5 Events

- Events

```

public const string SYS_MODEL_CHANGED = "AURORA_SYS_MODEL_CHANGED";
public const string SYS_MODEL_RELOADED = "AURORA_SYS_MODEL_RELOADED";
public const string SYS_MODEL_ITEM_CHANGED = "AURORA_SYS_MODEL_ITEM_CHANGED";
public const string SYS_MODEL_NAME_CHANGED = "AURORA_SYS_MODEL_NAME_CHANGED";
public const string SYS_MODEL_ENTRY_CHAGNED = "AURORA_SYS_MODEL_ENTRY_CHAGNED";
public const string SYS_REQ_MODEL_EDITING_CHANGE = "AURORA_SYS_REQ_MODEL_EDITING_CHANGE";

```

| 2.6 Model 사용하기

- 모델 Manager로부터 현재 활성화된 모델이름 가져오기 (현재 Activated Recipe의 모델이름을 가져올 수 있다. 단, ModelRecipeItem을 상속받은 레시피가 존재해야한다.)
→ 모델이름으로부터 ModelItem을 가져올 수 있다.

```

var currentModelName = _Manager.ActivatedModelName();
var visionItem = _ModelManager.ModelItem(currentModelName, "VisionModel");

```

→ 현재 활성화된 레시피로부터 모델이름을 가져와서 해당 이름을 통해 모델을 가져올 수 있다.

```

var activatedRecipe = _Manager.ActivatedRecipe();
var modelName = _Manager.ModelName(activatedRecipe.id);
var visionItem = _ModelManager.ModelItem(modelName, "VisionModel");

```

| 2.7 Model Recipe

- 모델 관리는 두가지 방식으로 제공된다.
1) 모델 관리를 변동부에서 직접 수행하는 경우
2) 모델 관리를 플랫폼에 이관하는 경우
- 여기서는 두번째 모델관리를 플랫폼에 이관하는 경우에 대해 기술한다.
- Recipe에서 ModelName을 담고 있는 Item은 ModelRecipeItem을 상속받도록 한다. (ModelRecipeItem을 상속받은 아이템이 없으면 변동부에서 직접관리하는 것으로..확인)

```

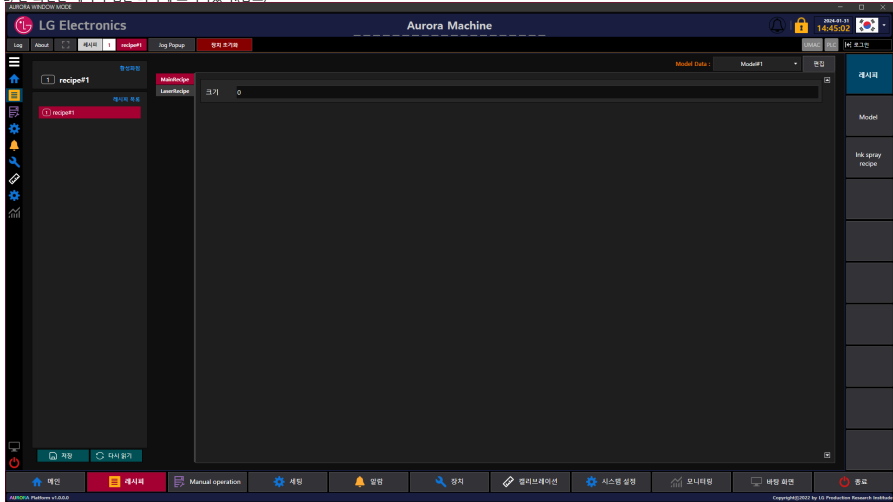
public abstract class ModelRecipeItem : NotifyPropertyChangedBase, IRecipeItem
{
    private string _ModelName;

    [Browsable(false)]
    public string ModelName
    {
        get => _ModelName;
        set => SetProperty(ref _ModelName, value);
    }
}

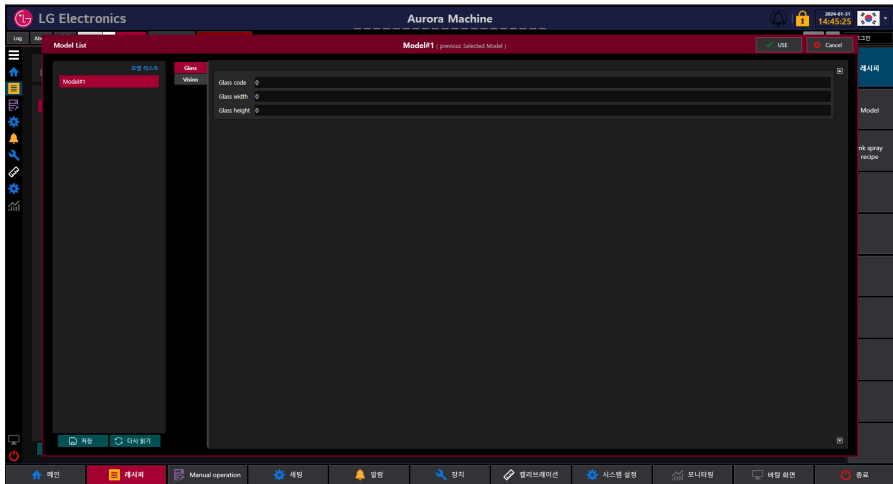
```

- ModelRecipeItem을 상속받은 RecipeItem이 있다면 RecipeViewer는 아래와 같이 표현된다.
1) 상단에 Model 선택할 수 있는 ComboBox가 표시된다.

2) 본 화면은 레시피 탭을 좌측에 표시하였다.(참고)



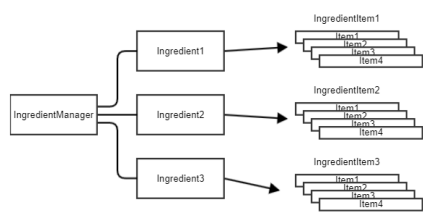
■ 편집버튼을 누르면 Dialog가 띄워지며 현재 선택된 Item이 Editing Model로 표시된다.



Ingredients

Ingredient는 “재료”라는 의미로, 레시피/모델 등 다양한 곳에서 사용하는 Sub Recipe를 의미한다.

좀 더 정확히 표현하면 Ingredient Recipe로 표현할 수 있으며, 다양한 재료들의 레시피로 구성된다.



| IngredientItem 구현

→ IIngredientItem을 상속하여 구현한다.

```
public class CameraItem : IIngredientItem
{
    public int Value1 { get; set; }
    public int Value2 { get; set; }
    public string MyProperty { get; set; }
}
```

| Register Ingredients

→ 물론 UserControl을 등록할 수 있다.

```
public void RegisterIngredients(IngredientsRegister register)
{
    register(
        "Camera", new CameraItem()
    );
    //throw new NotImplementedException();
}
```

| Register 전창 모델 Viewer

→ UCIngredient로 등록하되 **Ingredient Name**을 써주어야 한다.

```
var Recipe = new MainMenu("Recipe", MenuIcon.GetMenuIcon(eMenuIcon.RECIPE));
Recipe.AddSubMenu("Recipe", new UCRecipe()); // new UCRecipe();
Recipe.AddSubMenu("Model", new UCModel());
Recipe.AddSubMenu("Ingredients", new UCIngredient("Camera"));
manager.RegisterMainMenu(Recipe);
```

| Default Viwer

→ PropertyGrid

| Ingredient Custem Viwer 구현방법

→ ViewModel은 IIngredientViewModel을 상속해서 구현해야한다.

```
public interface IIngredientViewModel
{
    void OnChanged(string itemName);

    void Reloaded(string itemName);
}
```

| Events

```
public const string SYS_INGREDIENT_ENTRY_CHAGNED = "AURORA_SYS_INGREDIENT_ENTRY_CHAGNED";

public const string SYS_INGREDIENT_ITEM_CHAGNED = "AURORA_SYS_INGREDIENT_ITEM_CHAGNED";

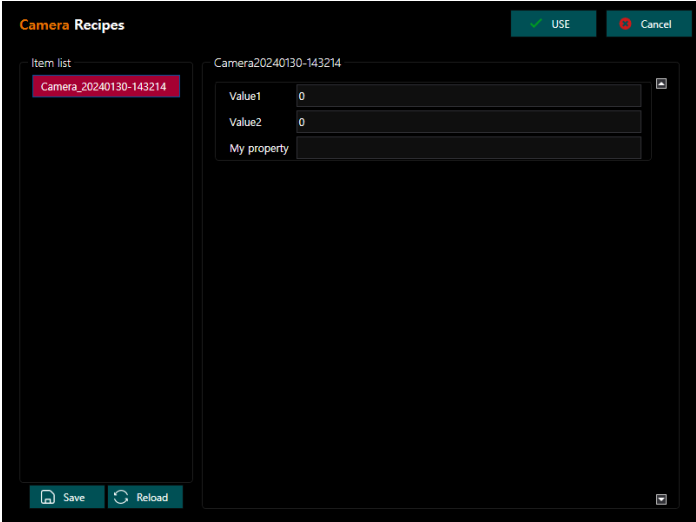
public const string SYS_INGREDIENT_RELOADED = "AURORA_SYS_INGREDIENT_RELOADED";
```

| Ingredient Dialog

→ 직접 Dialog를 띄우고 싶을때는 아래 코드를 사용한다. 파라미터에는 Ingredient Name을 써주어야 한다.

→ Ingredient는 Model대비 설정값이 적을 수 있어 Dialog활용도 괜찮을 것 같다.

```
var win = new WinIngredient("Camera");
win.Owner = Application.Current.MainWindow;
win.Width = 600;
win.Height = 600;
win.ShowDialog();
```



| Ingredient 사용하기

Ingredients = _IngreManager["Camera"].Items();

```
_IngredientManager = auroraEnv.IngredientsManager;
var cameraRecipe = _IngredientManager["Camera"];

var cameratItem = cameraRecipe.IngredientItem("Camera0101") as CameratItem;
```

| IngredientManager Interface

```
public interface IIngredientsManager
{
    IList<string> DataKeys(); <- Ingredient의 종류

    IIngredient Ingredient(string key);

    IIngredient this[string key] { get; }

    string Path(); // Ingredient가 저장되는 경로 알아내기.
```

}

| Ingredient Interface

→ Ingredient는 2가지 방법으로 사용가능하며 혼용도 가능하다.

- 1) Recipe / Model에서 Ingredient Item이름을 저장하고, 필요시 꺼내서 사용.
- 2) Ingredient만 독립적으로 관리하며 Activate 시켜서 사용하기. → 이 경우 마지막 사용했던 Ingredient Active Item을 기억하고 재시작할때 설정하는 것이 필요할 수도 있다.

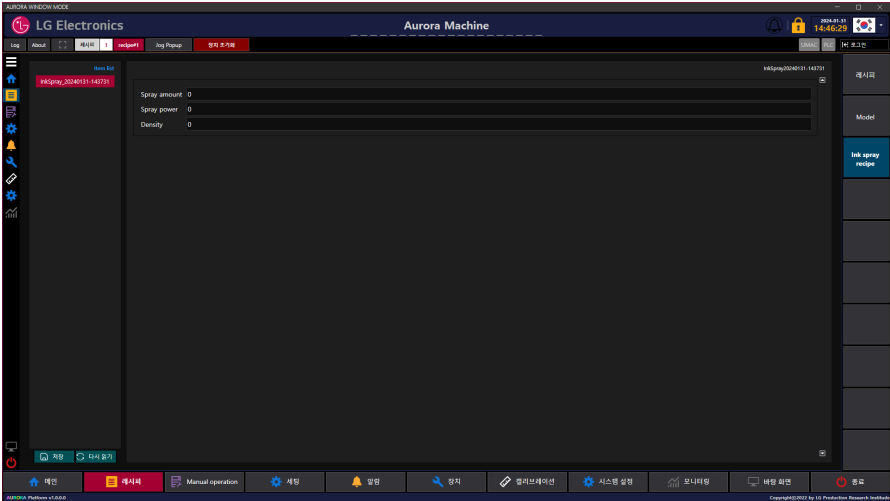
```
public interface IIngredient
{
    IList<string> Items();

    int ActivateItem(string itemName);
    string ActivatedItem();

    IIngredientItem IngredientItem(string itemName);
    IIngredientItem this[string itemName] { get; }

    IIngredientItem Clone(string itemName);
}
```

| 전창



[09] Event

0. Overview

- Aurora의 이벤트는 Message로 표현하며 이벤트를 보내는 Event Publish와 받는 Subscribe가 있다.

1. Event Publish

- 기본적으로 Pub/Sub구조를 따르며 Reactive Framework으로 구현되었다.
- Aurora의 Event Manager를 통해 이벤트를 Publish한다.

EventPublisher에 Access하는 방법

→ 전역 메소드인 AuroraEnv()를 통해 EventPublisher를 가져올 수 있다.

```
IEventPublisher EvPublisher = Application.Current.AuroraEnv().EventPublisher
```

→ 장비 Class의 AuroraEnv를 통해 마찬가지로 AuroraEnv.EventPublisher를 Access할 수 있다. (Aurora고정부에서 Set해준다.)

```
class EquipmentSetup : IEquipmentSetup
{
    static readonly Logger LOG = new Logger("Equip");

    public IAuroraEnvironment AuroraEnv { get; set; }
    ...
}
```

→ SequenceContext를 통해 가져올 수 있다.

```
public static SequenceBuilder Builder()
{
    SequenceBuilder sb = new SequenceBuilder(2 * 1000);
    sb.AddAction(SEQ_CHECK_ENV, CheckEnv);
    ...
}

private static SequenceActionResult CheckEnv
(SequenceContext context, string id, SequenceActionResult param)
{
    ...
    context.EventPublisher.Publish(
        new EventMessage(
            context.SequenceId,
            AuroraSysEvent.REQ_HIDE_VIEW
```

```
    )
};
...
}
```

Event Message : Event Publish할때 보내는 메시지이다.

```
public class EventMessage
{
    public string Subject { get; }
    public object Param { get; }
    public string From { get; }

    public EventMessage(
        string from,
        string subject,
        object param = null)
    {
        From = from;
        Subject = subject;
        Param = param;
    }
}
```

사용예
→ Param은 object이기 때문에 Publish하는 곳과 Subscribe하는 곳에서 정해진 구조를 따른다.

```
publisher.Publish(
    new EventMessage("TESTER", SUBJECT_FOO, null)
);
```

```
var param = new ParamAlarmEvent(
    timestamp,
    code,
    alarmDef.Heavy,
    alarmMessage
);
mEventPublisher.Publish(
    new EventMessage(
        "AlarmManager",
        CTQEventIds.ALARM_OCCURED,
        param
    )
);
```

2. Event Subscribe

Publish한 Event를 Subscribe하는 방법이다.

EventObservable에 Access하는 방법.

→ Publisher와 같다.

→ 즉 AuroraEnv를 통해 가져오거나 SequenceContext를 통해 Access할 수 있다.

```
IObservable Observable = Application.Current.AuroraEnv().EventObservable;
```

Subscribe

1) EventObservable을 통해 Subscribe.

AsObservable로 Subscribe할 이벤트를 정해준다.

```
{
    var eventObservable = Application.Current.AuroraEnv().EventObservable;
    _DisposableEvent = eventObservable
        .AsObservable("ScaleGenerated")
        .ObserveOnDispatcher()
        .Subscribe(OnWeight);
}

private void OnWeight(EventMessage obj)
{
    Weight = (double)obj.Param;
}
```

2) UI Thread에서 메시지를 Subscribe하기 위한 용도 : ObserveOnDispatcher(), ObserveOn(Application.Current.Dispatcher)

→ Version차이에 의해 두가지중 하나를 사용하도록 한다.

```
1) DisposableEvent = eventObservable
    .AsObservable("ScaleGenerated")
    .ObserveOnDispatcher()
    .Subscribe(OnWeight);

2) DisposableEvent = eventObservable
    .AsObservable("ScaleGenerated")
    .ObserveOn(Application.Current.Dispatcher)
    .Subscribe(OnWeight);
```

3) Linq에서 제공하는 Select와 Where를 통해 변환과 필터링을 사용할 수 있다.

```
mDisposableAlarmEvent = eventObservable
    .AsObservable(CTQEventIds.ALARM_OCCURED)
    .Select(evt => evt.Param as ParamAlarmEvent)
    .ObserveOn(Application.Current.Dispatcher)
    .Subscribe(OnAlarmOccurred);

mEventObserver = eventObservable
    .AsObservable(AuroraSysEvent.SYS_SETTING_CHANGED)
    .Select(msg => msg.Param as string)
```

```
.Where(param => param == AuroraSettings.Equipment)
.ObserveOn(Application.Current.Dispatcher)
.Subscribe(_ => OnEquipInfoChanged());
```

Dispose
Subscribe가 Return해주는 것은 IDisposable로 Lifecycle에 따라 적절히 Dispose해주어야 한다.
Destructor나 Dispose 또는 이벤트 구독을 중단하고자 할때등 Dispose를 적절히 호출해주자. 프로그램 Lifecycle과 같다면 안해줘도 되겠지만...

```
readonly IDisposable _DisposableEvent;

public MainModel()
{
    var eventObservable = Application.Current.AuroraEnv().EventObservable;
    _DisposableEvent = eventObservable
        .AsObservable("ScaleGenerated")
        .ObserveOnDispatcher()
        .Subscribe(OnWeight);
}

~MainModel()
{
    _DisposableEvent?.Dispose();
}

private void OnWeight(EventMessage obj)
{
    Weight = (double)obj.Param;
}
```

3. System Event

Aurora에서 정의된 System Event는 다음과 같다. 적절히 활용하면 되겠다.

System → 변동부

Subject	Param(Object) Type	Description
AuroraSysEvent.SYS_EQUIPMENT_LOADED	IEquipmentSetup	Instance
AuroraSysEvent.SYS_EQUIPMENT_LOAD_FAILED	null	
AuroraSysEvent.SYS_SETTING_CHANGED	string	변경된 Setting 항목의 이름
AuroraSysEvent.SYS_DEVICE_ACTIVATED	string	Activate된 deviceId
AuroraSysEvent.SYS_DEVICE_DEACTIVATED	string	Deactivate된 deviceId
AuroraSysEvent.SYS_DEVICE_IS_COMM_ERROR	(string deviceId, bool isCommError)	통신에러가 발생된 deviceId, 에러 상태
AuroraSysEvent.SYS_RECIPE_ACTIVATED	(int id, string name)	Activate된 Recipe의 Id와 이름
AuroraSysEvent.SYS_RECIPE_ENTRY_CHANGED	RecipeEntryChangedParam	New, Copy, Remove시에 해당하는 Recipe Id정보
	public int RemovedId { get; set; } = -1; public int AddedId { get; set; } = -1; public int CopiedId { get; set; } = -1;	
AuroraSysEvent.SYS_RECIPE_RELOADED		현재 편집중인 Recipe가 Reload되었음을 알린다.
AuroraSysEvent.SYS_RECIPE_ITEM_CHANGED		레시피 저장시 발생한다.
AuroraSysEvent.SYS_MODEL_CHANGED	string	특정 모델이 Active되었을때 활성화된 모델이름
AuroraSysEvent.SYS_MODEL_ITEM_CHANGED	string	특정 모델이 저장되었을때 저장하는 모델 이름
AuroraSysEvent.SYS_MODEL_ENTRY_CHAGNED		모델리스트의 갯수가 변경되었을때(Copy, SaveAs, Remove)
AuroraSysEvent.SYS_MODEL_EDITING_CHANGED	null	
AuroraSysEvent.SYS_LANGUAGE_CHANGED	string	language(ko-KR, ...)
AuroraSysEvent.SYS_MENU_CHANGED	(mainName, subName)	메뉴가 변경되었을때
AuroraSysEvent.SYS_ALARM_SET_ALARM	AlarmDescription	특정 알람이 발생했음을 알림
AuroraSysEvent.SYS_ALARM_RESET_ALARM	AlarmDescription	특정 알람이 제거되었음을 알림
AuroraSysEvent.SYS_ALARM_RESET_ALL_ALARM	List<AlarmDescription>	현재 발생한 모든 알람이 Clear되었음을 알림
AuroraSysEvent.SYS_PERMISSION_CHANGED		로그인/로그아웃으로 인해 Permission이 변경되었음을 알림. ApplicationExt의 CurrUserId, PermissionLevel로 확인 가능.

변동부 → System

Subject	Param(Object) Type	Description
AuroraSysEvent.SYS_REQ_MAIN_PROGRESS	int	title영역의 프로그레스위치를 변경한다.
SYS_REQ_SCREEN_INTERLOCK	bool	스크린 lock 여부
AuroraSysEvent.SYS_REQ_MAIN_LOCK	FlyoutParam	화면상단의 DoorLock잠금 상태 변경
AuroraSysEvent.SYS_REQ_MAIN_LAMP	eTowerLamp	화면상단의 TowerLamp 색상 변경
AuroraSysEvent.SYS_REQ_MENU_CHANGE	(mainName, subName)	메뉴변경요청
AuroraSysEvent.SYS_REQ_MAIN_MENU_ENABLE	bool	메뉴조작 Enable/Disable 공정중, Run중 특정 탭에 접근하지 못하게 한다던지 할때 사용.
SYS_REQ_WINDOW_MODE		
SYS_REQ_WINDOW_FULL_SCREEN	bool	full screen인지 여부
AuroraSysEvent.SYS_REQ_MANUAL_BACKUP_START	string	수동백업에 대한 시작요청. 파라미터가 없으면 전체 백업
SYS_REQ_TOAST_MESSAGE	(int 종류, string message)	종류 : 0 - Infomation 1 - Success 2 - Error 3 - Warning message : 토스트 팝업에 보여주려는 message

Subject	Param(Object) Type	Description
SYS_REQ_TOAST_MESSAGE_CLEAR	null	
SYS_REQ_CHANGE_LANGUAGE	string	바꾸려는 언어 코드
SYS_REQ_MODEL_EDITING_CHANGE	string	편집할 모델 id

[10] Devices

0. OverView

- 장치 Driver구현에 대해 알아보자.
- Device Type
 - Device에 Device Type을 enum으로 설정할 수 있으며 강제사항은 아님.

```
public enum DeviceType
{
    Etc,
    Camera,
    Trigger,
    ArgosVP,
    DefectCollector,
    Motor,
    Mover,
    Light,
    DigitalIO,
    LineProfiler,
    PowerMeter,
    HeightSensor,
    Chiller,
    Laser,
    DustCollector,
    IO,
    Controller,
    PLC,
}
```

1. Device 구현

- Device는 장치 Driver를 구현한 Device와 연결정보/파라미터를 설정하는 DeviceConfig 두가지 클래스로 구성한다.

1.1 Device

- 특정 Device를 제작할 때에는 Device 추상클래스를 상속받아서 만든다.

```
public abstract class Device
{
    public DeviceType Type { get; set; }
    public CommunicationType CommType { get; set; }
    public string DeviceId { get; set; }
    public virtual IntPtr NativeHandle => IntPtr.Zero;

    protected IEventPublisher EventPublisher { get; private set; }

    public bool IsHiddlenDevice { get; set; }
    public bool IsActivated { get; private set; }
    public bool IsSimul { get; set; }

    protected List<int> _alarmList;
    public bool IsAlarm { get => (_alarmList != null) && (_alarmList.Count > 0); }

    public void AddAlarm(int alarmCode){...} // code형태의 알람을 추가할 수 있다.
    public virtual int GetLastAlarm(){...} // 마지막 알람코드 가져오기
    public virtual List<int> GetAlarmList() // 알람코드 리스트 가져오기
    public virtual void ClearAlarms() // 알람리스트 초기화하기.

    public int Activate()
    => Activate(ConfigProvider?.Invoke()); //

    public int Activate(IDeviceConfig config)
    {
        var err = DoActivate(config);
        if (err != 0)
            return err;

        IsActivated = true;
        PublishActivatedEvent(EventPublisher, DeviceId); // Activate이 되면 AuroraSysEvent.SYS_DEVICE_ACTIVATED 이벤트를 날려준다.(deviceId)
        return err;
    }

    public int Deactivate()
    {
        if (!IsActivated)
```

```

    {
        return El.OK;
    }

    var err = DoDeactivate(); // Deactivate이 되면 AuroraSysEvent.SYS_DEVICE_DEACTIVATED 이벤트를 날려준다. (deviceId)
    if (err != 0)
        return err;
    IsActivated = false;
    PublishDeactivatedEvent(EventPublisher, DeviceId);
    return err;
}

protected abstract int DoActivate(IDeviceConfig config); // 필수 구현요소
protected abstract int DoDeactivate(); // 필수 구현요소
...
}

```

→ IsActivated : 현재 연결되었는지 상태 체크한다.

→ IsHiddenDevice : 숨겨진 디바이스인지 체크한다. 숨겨지면 Device설정창에서 표시되지 않는등의 역할을 한다.

→ DeviceId : 생성시 설정하는 DeviceId로 각 디바이스를 Access시 활용한다.

→ Communicatin Type : 해당 디바이스의 연결타입을 설정한다. 필수요소는 아니지만 되도록 설정하도록 하자.

```

public enum CommunicationType
{
    Serial = 0,
    Socket = 1,
    ModbusTCP = 2,
    ModbusRTU = 3,
    IPAndStationNo = 4,
    USB = 5,
    Etc = 6
}

```

→ IsSimul : 현재 디바이스가 시뮬레이션인지 확인하거나 설정한다.

→ IsAlarm : 현재 디바이스가 알람이 발생했는지 확인한다.

→ EventPublisher : 디바이스에서 특정상황에서 이벤트를 송신할때 사용할 수 있다.

- Device의 생성

→ 플랫폼에서 제공하는 Device를 상속받아 구현한다.

```

class TestDevice : Device
{
    internal TestDevice(bool isSimul)
    {
        IsSimul = isSimul;
        CommType = CommunicationType.Serial;
        Type = DeviceType.Controller;
    }

    protected override int DoActivate(IDeviceConfig config)
    {
        var testConfig = config as TestDeviceConfig;
        if (testConfig.IsSimul) {
            return El.OK;
        }

        return El.OK;
    }

    protected override int DoDeactivate()
    {
        return El.OK;
    }
}

```

→ DoActivate(연결하기)과 DoDeactivate(연결끊기)을 구현해야하며, DoActivate에는 DeviceConfig가 인자로 전달된다.

| 1.2 DeviceConfig

- Device Config를 만들 때에는 IDeviceConfig Interface를 상속 받아서 만든다.

```

public interface IDeviceConfig
{
    IConnConfig ConnConfig { get; set; }
}

```

→ ConnConfig : 연결설정에 대한 정보이다. 플랫폼에서는 SerialConnConfig와 SocketConnConfig, ModbusSerialConfig, ModbusTcplpConfig 네가지를 제공한다.

```

public class SerialConnConfig : IConnConfig
{
    public SerialConnConfig();
}

```

```

        public int PortNum { get; set; }
        public int BaudRate { get; set; }
        public Parity Parity { get; set; }
        [SelectorStyle(SelectorStyle.ComboBox)]
        public StopBits StopBits { get; set; }
        public int DataBits { get; set; }
        public bool DtrEnable { get; set; }
        public bool RtsEnable { get; set; }
        [Browsable(false)]
        public string NewLine { get; set; }
        public int ReadTimeout { get; set; }
        public int WriteTimeout { get; set; }
        public int ReadRetryCount { get; set; }
    }

    public class SocketConnConfig : IConnConfig
    {
        public SocketConnConfig();

        public SocketAddress Address { get; set; }
        public int ResponseTimeMS { get; set; }
    }

    public class ModbusSerialConfig : IConnConfig
    {
        public SerialConnConfig Serial { get; set; }

        public int Slave { get; set; } = 1;
    }

    public class ModbusTcpIpConfig : IConnConfig
    {
        public string Address { get; set; }

        public int Slave { get; set; }
    }

    public class SocketAddress
    {
        public SocketAddress();

        public string IpAddress { get; set; }
        public int PortNumber { get; set; }
    }

```

- DeviceConfig Class 구현.

```

public class TestDeviceConfig : IDeviceConfig
{
    [DisplayName("Simulation Mode (this device)")]
    public IConnConfig ConnConfig { get; set; }

    public static IDeviceConfig GetDeviceConfig()
    {
        return new TestDeviceConfig();
    }

    public TestDeviceConfig()
    {
        ConnConfig = new SerialConnConfig() {
            PortNum = 13,
            BaudRate = 38400,
            Parity = System.IO.Ports.Parity.None,
            StopBits = System.IO.Ports.StopBits.One,
            DataBits = 8,
            NewLine = "\n",
            ReadTimeout = 1000
        };
    }
}

```

→ ConnConfig에 SerialConnConfig를 대입하였다.

1.3 Device Register

- Device는 EquipmentSetup Class의 Register Devices내에서 등록

View는 RegisterViews를 통해 Device Setting Views를 등록한다.

```

ExecutionResult Register(string deviceId, Func<Device> deviceFactory, IDeviceConfig defaultConfig = null, Func<Device, ContentControl> settingUIFactory = null, bool IsHiddenDeviceList = false);

```

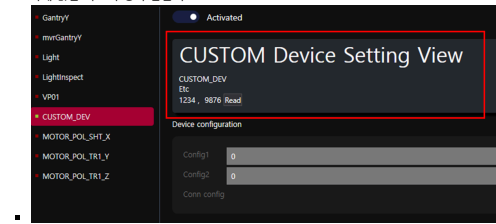
- deviceId : DeviceManager를 통해 받아올 때 사용할 Device의 ID
- deviceFactory : Device를 리턴 해 주는 delegate 함수
- defaultConfig : IDeviceConfig 상속받아 만든 Config의 default값. 저장된 Config가 없는 초기 부팅 시 적용될 값
- settingUIFactory : 입력 파라메터로 Device를 받고, Device를 테스트할 수 있는 뷰를 리턴하는 delegate 함수
뷰 작성 가이드는 1.4항 참조

```
public void RegisterDevices(IDeviceRegistry registry)
{
    registry.Register("Test", () => new TestDevice(true), TestDeviceConfig.GetDeviceConfig());
}

public void RegisterViews(IViewManager manager)
{
    // device
    var device = new MainMenu(ViewIds.Device, MenuIcon.GetMenuIcon(eMenuIcon.DEVICE), PermissionLevel.NONE, false);
    device.AddSubMenu(ViewIds.Device, new DeviceSettingView()); // "ViewIds.Setting" 는 EquipDef.cs 에서 정의한 ViewId
    manager.RegisterMainMenu(device);
}
```

1.4 Test View 만들기

- View와 ViewModel을 만들고 ViewModel이 해당 Device를 가지는 구조
- 화면 진입/진출 시 VisibleChanged 이벤트를 반드시 등록해서 '타이머'나 '이벤트' 등의 작업이 백그라운드에서 불필요하게 동작되는 일이 없도록 한다.
- 아래 붉은 박스의 영역 만들기



- View(xaml)

```
<behavior:Interaction.Triggers>
  <behavior:PropertyChangedTrigger Binding="{Binding IsVisible, ElementName=uc}">
    <behavior:InvokeCommandAction Command="{Binding CMD_VisibleChanged}" CommandParameter="{Binding IsVisible, ElementName=uc}" />
  </behavior:PropertyChangedTrigger>
</behavior:Interaction.Triggers>
<GroupBox Style="{StaticResource StGroupBox23}">
  <GroupBox.Resources>
    <Style TargetType="TextBlock" >
      <Setter Property="VerticalAlignment" Value="Center" />
    </Style>
  </GroupBox.Resources>
  <GroupBox.Header>
    <TextBlock Text="CUSTOM Device Setting View" FontSize="34" />
  </GroupBox.Header>
  <StackPanel Orientation="Vertical">
    <TextBlock Text="{Binding DeviceId}" />
    <TextBlock Text="{Binding Type}" />
    <StackPanel Orientation="Horizontal">
      <TextBlock >
        <Run Text="{Binding Measure1}" />
        <Run Text=", " />
        <Run Text="{Binding Measure2}" />
      </TextBlock>
      <Button Content="Read" Margin="4 0" Command="{Binding CMD_Read}" />
    </StackPanel>
  </StackPanel>
</GroupBox>
```

- ViewModel(cs)

```
class CustomDeviceViewModel : NotifyPropertyChangedBase
{
    private CustomDevice _customDevice;
    public CustomDeviceViewModel(Device dev)
    {
        _customDevice = (CustomDevice)dev;

        CMD_VisibleChanged = new DelegateCommand(CmdVisibleChanged);
        CMD_Read = new DelegateCommand(Read);
    }

    private void Read(object obj)
    {
        var val = _customDevice.GetMeasure();
        Measure1 = val.Item1;
        Measure2 = val.Item2;
    }

    private void CmdVisibleChanged(object obj)
    {
        // 화면에 들고 날 때 timer start/stop이나 이벤트 수신/제거 등의 작업은 여기서 구현한다.
        if (obj is bool visible)
        {
            if (visible)
            {
            }
            else
            {
            }
        }
    }

    public ICommand CMD_VisibleChanged { get; }
```

```

public ICommand CMD_Read { get; }
//

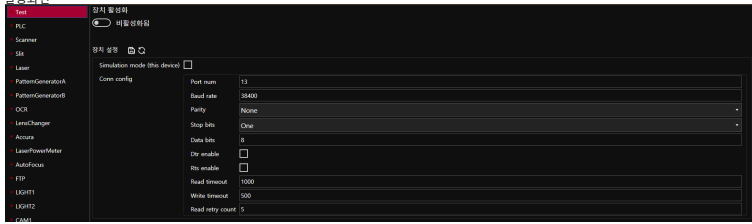
public string DeviceId { get => _customDevice.DeviceId; }

private string type;
public string Type { get => _customDevice.Type.ToString(); }

private double measure1;
public double Measure1 { get => measure1; set => SetProperty(ref measure1, value); }
private double measure2;
public double Measure2 { get => measure2; set => SetProperty(ref measure2, value); }
}

```

실행화면



2. 통신 프로토콜 구현

- Aurora++에서는 SerialPort나 Socket에 대해서 추상화 Class를 제공하였으나, Aurora2에서는 제공하지 않음
- Device를 상속받는 SerialPortDevice/SocketDevice는 삭제한다.
- 기존 SerialConn과 SocketConn등 장치 Connection Protocol을 Wrapping한 Class를 통신 모듈(가칭)이라 칭한다.
- 따라서 아래와 같이 구현한다.

1) Device 기본

```

class SerialTestDevice : Device
{
    public SerialTestDevice()
    {
        Type = DeviceType.Controller;
        CommType = CommunicationType.Serial;
    }

    protected override int DoActivate(IDeviceConfig config)
    {
        // Activation코드
    }

    protected override int DoDeactivate()
    {
        // Deactivation코드
    }

    // 각종 Api 및 구현부
}

```

2) Device Monitoring

```

class SerialTestDevice : Device
{
    SerialConn _Comm;
    string Value = string.Empty;

    public SerialTestDevice()
    {
        Type = DeviceType.Controller;
        CommType = CommunicationType.Serial;
        SetMonitoringService(OnMonitoring, 100);
    }

    private void OnMonitoring ()
    {
        if (IsCommError)
        {
            return;
        }

        // 모니터링할 무언가
    }

    protected override int DoActivate(IDeviceConfig config)
    {
        // Activation코드
    }

    protected override int DoDeactivate()
    {
        // Deactivation코드
    }
}

```

```
// 각종 Api 및 구현부
}
```

3) 통신 모듈 활용방법 (시리얼통신모듈)

- SetCommError는 Device에 이미 구현된 IsCommError설정 Property에 Set해주는 Api로 SerialConn에서 통신 문제가 생기면 알려주는 용도로 활용한다.
- CommError시 무언가 추가로 할일이 디바이스 내부에 있다면 Action Delegate를 만들어서 넣어주면 되겠다.

```
class SerialTestDevice : Device
{
    SerialConn _Comm;
    public SerialTestDevice()
    {
        Type = DeviceType.Controller;
        CommType = CommunicationType.Serial;
        SetMonitoringService(OnMonitoring, 100);
    }

    private void OnMonitoring()
    {
        if (IsCommError)
        {
            return;
        }

        // 모니터링할 무언가
    }

    protected override int DoActivate(IDeviceConfig config)
    {
        _Comm = new SerialConn(DeviceId, IsSimul, SetCommError);
        return _Comm.Open(config.ConnConfig as SerialConnConfig);
    }

    protected override int DoDeactivate()
    {
        return _Comm.Close();
    }

    // 각종 Api 및 구현부
}
```

4) 2개 이상의 디바이스를 하나의 디바이스로 구현시는 기존 구현방법을 다르게 구현한다.

- 1) 2개 이상의 디바이스의 interface 정의, ...

3. Device Monitoring Service

- 하나의 디바이스에서 모니터링을 구현해야하는 경우 일반적으로 Thread(or Task)를 만들어 모니터링을 구현하거나, 모니터링 시퀀스를 통해 구현한다.
- 구현 편의를 위해 플랫폼에서 Monitoring Service를 제공하며 이에 대해 알아본다.
- 제공 Api

```
protected void SetMonitoringService(Action job, int interval = 100);
//
// Summary:
//     StartMonitoringService. Protected Method.
protected void StartMonitoringService();
//
// Summary:
//     StopMonitoringMethod. Protected Method.
protected void StopMonitoringService();
```

SetMonitoringService : 모니터링을 수행할 Action Delegate와 모니터링 주기를 지정한다. (Interval은 기본 100ms로 설정되어있다.)

→ MonitoringService로 등록된 Job은 Interval로 설정된 간격마다 수행되며, Device가 Activate된 상태에서만 동작한다.

→ Interval은 Job수행시간을 포함하여 계산된다. (ex. 100ms면서 Job수행시간이 10ms면 90ms이후에 다음 Job이 실행된다.)

Start와 Stop은 일반적으로 사용하지 않으나, 디바이스 연결상태에서 수동으로 Start/Stop하기 위한 용도로 제공된다.

- Monitoring Example

```
public class PumpController : Device
{
    Logger Log = new Logger("PumpController");

    public PumpController(ePumpController type)
    {
        _Type = type;
        Type = DeviceType.Etc;
        // 디바이스 생성자에서 Job등록
        SetMonitoringService(OnMonitoring, 100);
    }

    private void OnMonitoring()
    {
        for (int i = 0; i < _BoardCount; i++) {
```

```

var pumpData = PumpData[i];
for (int nCh = 0; nCh < MAX_CHANNEL_COUNT; nCh++) {
    if (_PumpController.GetOutputVoltage(pumpData.BoardNumber, nCh, out double voltage)) {
        pumpData.Voltage[nCh] = voltage;
    } else {
        // 통신에러 체크. 통신에러가 True가 되면 다시 정상일때까지 Job은 수행되지 않는다.
        IsCommError = _PumpController.IsCommError();
        return;
    }
}
}
}
}
}
}

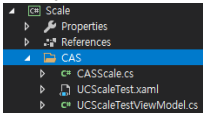
```

4. Device 만들기 실습

- SerialPortDevice를 이용해서 Scale Device를 만들어 보자.

→ Serial Type의 저울 Device 만들기.

1) Scale Project를 생성하였다. (x64로 Configuration바꾸기)



1) 저울은 제조사가 다양하기 때문에 제조사 폴더를 생성해보았다. 여기서는 임의로 CAS라고 가정하고 생성

2) 하위에는 Device인 CASScale Class를 만들고, Device Configuration View에서 장비테스트를 할 수 있는 UCScaleTest라는 Custom Control도 하나 생성하였다.

2) CASScale Device와 Config

→ Serial통신을 사용하는 Device상속받은 SerialPortDevice를 사용한다.

```

public class CASScale : SerialPortDevice
{
    public CASScale(string deviceId, bool isSimul) : base(deviceId, isSimul)
    {
    }

    protected override int DoActivate(IDeviceConfig config)
    {
        var casConfig = config as CASScaleConfig;
        return Open(casConfig.ConnConfig as SerialConnConfig);
    }

    protected override int DoDeactivate()
    {
        return Close();
    }

    public double GetWeight()
    {
        Random r = new Random();

        return r.NextDouble() * 1100;
    }
}

public class CASScaleConfig : IDeviceConfig
{
    public IConnConfig ConnConfig { get; set; }
    public int MinValue { get; set; } = 10;
    public int MaxValue { get; set; } = 1000;

    public CASScaleConfig()
    {
        ConnConfig = new SerialConnConfig() {
            PortNum = 8,
            BaudRate = 9600,
            Parity = System.IO.Ports.Parity.Odd,
            StopBits = System.IO.Ports.StopBits.One,
            DataBits = 8,
            NewLine = "\n",
            ReadRetryCount = 5
        };
    }
}

```

→ 저울이라 GetWeight를 하나 추가하고 Get을 하면 랜덤값을 리턴하도록 해주었다.

→ 저울의 최소/최대값을 추가해주고, 이상치 확인에 사용하도록 해주었다.

3) 디바이스 등록

```

public void RegisterDevices(IDeviceRegistry registry)
{
    var setting = AuroraEnv.SettingEntry[AuroraSettings.Equipment] as EquipmentInfo;

    registry.Register(DeviceIds.Scale, () => new CASScale(DeviceIds.Scale, setting.SimulationMode)
        , new CASScaleConfig(), device => new UCSTest(AuroraEnv, device as CASScale));
}

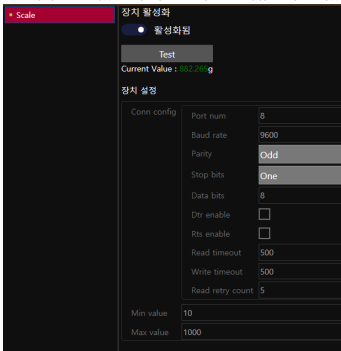
public void RegisterViews(IViewManager manager)
{
    // device
    var device = new MainMenu(ViewIds.Device, MenuIcon.GetMenuIcon(eMenuIcon.DEVICE), PermissionLevel.NONE, false);
    device.AddSubMenu(ViewIds.Device, new DeviceSettingView()); // "ViewIds.Setting" 는 EquipDef.cs 에서 정의한 ViewId
    manager.RegisterMainMenu(device);
}

```

- 시뮬여부는 장비설정에서 가져와서 사용하였다.
- CustomControl도 디바이스 등록시 추가

4) CustomControl

- 장비가 연결되었을때 테스트 가능한 메뉴를 구성하였다.



- 버튼이 눌리면 이상치인지 확인하는 코드도 추가해주었다.

ViewModel

```

private bool _IsNormal;
public bool IsNormal { get => _IsNormal; set => SetProperty(ref _IsNormal, value); }

private void Click(object obj)
{
    CurrentValue = _Scale.GetWeight();
    var config = _DeviceManager.GetDeviceConfig(_Scale.DeviceId) as CASScaleConfig;
    if (CurrentValue < config.MinValue || CurrentValue > config.MaxValue) {
        IsNormal = false;
    } else {
        IsNormal = true;
    }
}

```

- DeviceConfig는 DeviceManager를 통해서 가져올 수 있다.

Xaml

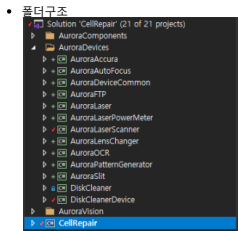
```

<Grid HorizontalAlignment="Left" IsEnabled="{Binding IsConnected}">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <Button Command="{Binding CMD_Click}" Height="30">Test</Button>
    <TextBlock Grid.Row="1" Foreground="White">
        Current Value : <Run Text="{Binding CurrentValue, StringFormat={}{0:0.000}}" Foreground="{Binding IsNormal, Converter={StaticResource convColor}}"/>
    </TextBlock>
</Grid>

```

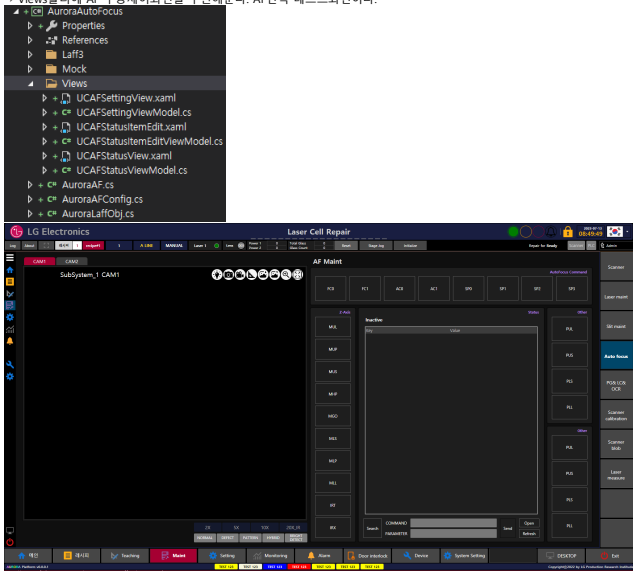
- Xaml과 ViewModel을 완성해보자.

4. 프로젝트 구조



- 디바이스는 모두 AuroraDevices안에서 각각의 프로젝트로 구분하여 구현한다. : 디바이스전개효율화
- 동일 타입의 디바이스는 한 곳으로 모은다. (Laser면 Laser프로젝트안에 여러가지의 Laser를 구현하도록 한다.)

- 디바이스의 수동제어 화면은 디바이스와 함께 제공한다. (다른 디바이스와 연관관계를 갖지 않는 선에서.) → 이렇게 하면 디바이스 재사용시 수동제어화면도 같이 재활용할 수 있다.
→ Views폴더에 AF 수동제어화면을 구현해준다. AF단독 테스트화면이다.



*. 참고 : [16][07] Device List

5. Motion Device 개발 가이드

- ACS, UMAC 등의 자체 프로그램 가능한 Motion제어의 경우 Device에서 처리 가능한 최대한의 기능을 자체 프로그램에서 개발한다.
→ PC의 부하를 낮추고 PC는 장비의 주요 기능에 집중할 수 있도록 하기위함.
- Interlock
 - 개별 축 및 Unit의 관련 축에 대한 Interlock은 자체 프로그램으로 개발
 - Unit간 Interlock은 Aurora에서 개발
- Fast/Slow 모션이나 회피동작 등은 가급적 자체 프로그램으로 개발

[11] Sequence

0. Overview

- Sequence 작업을 위한 기본 코드, Node 종류에 따른 생성, Action 함수표 및 예제 Sequence 코드들과 Draw로 그린 pdf에 대해 설명합니다.

1. Sequence 작업을 위한 기본 코드

Equipment.cs에 SequenceBuilder 등록

```
public void RegisterSequences(SequenceRegister register)
{
    register("Seq02", Seq02.Builder()); // 호출인자 ==> seq id 문자열, SequenceBuilder 객체 생성자
}
```

SequenceBuilder를 통한 sequence 작성

```
public class Seq02
{
    public static SequenceBuilder Builder()
    {
        SequenceBuilder sb = new SequenceBuilder(1000);
        sb.AddAction("Act1", Act1Func);
        sb.AddAction("Act2", Act2Func);

        sb.Flows(sb.Init > sb["Act1"] > sb["Act2"] > sb.Final);
        sb.Draw("Seq02.pdf");

        return sb;
    }

    // 위에서 사용할 delegate 함수도 구현한다.
    private static SequenceActionResult Act1Func(SequenceContext context, string id, SequenceActionResult param)
    { return SequenceAction.Ok(); }

    private static SequenceActionResult Act2Func(SequenceContext context, string id, SequenceActionResult param)
    { return SequenceAction.Ok(); }
}
```

- public SequenceBuilder(int stopTimeout, bool isFlowedLogLowLevel = false, LogLevel logLevel = LogLevel.I)
 - SequenceBuilder의 paramter는 StopTimeout 즉, Stop명령(혹은 Stop 버튼) 수행 시 대기시간이며 해당 시간 내에 수행중인 함수가 종료되지 않을 경우 Worker(Thread)를 강제종료 시킨다는 뜻
 - isFlowedLogLowLevel : true이면 로그를 Information Level이하로 출력한다. (D로그). 로그뷰어를 화면에 붙일때 너무 많은 로그를 화면에 뿌리지 않기 위함.
 - logLevel : Sequence 천이 로그의 Level을 변경할 때 사용.

```
public class Seq02
{
    public static SequenceBuilder Builder()
    {
        SequenceBuilder sb = new SequenceBuilder(1000);

        return sb;
    }
}
```

→ Draw함수는 Sequence Builder가 가지고 있는 Node들과 Node간의 연결을 PDF로 그려준다. (옵션)

2. Node 종류에 따른 생성, Action 함수표

Node 종류	생성 함수 (public bool ~)	Action 함수 (public delegate ~)	설명
Action	AddAction(string id, DoAction action);	<u>SequenceActionResult</u> DoAction(SequenceContext context, string id, <u>SequenceActionResult</u> param);	현재 context와 이전 Action 결과를 받아서 작업 실행한 결과를 반환한다.
Decision	AddDecision(string id, DoDecision decision);	<u>string</u> DoDecision(SequenceContext context, string id, ICollection<string> <u>branches</u> , SequenceActionResult param);	선택 가능한 branches 중에서 다음에 실행할 Node 이름 문자열을 반환한다.
Decision by Event	AddDecisionByEvent(string id, DoAcceptEvent action, int timeout = -1);	<u>SequenceActionResult</u> DoAcceptEvent(SequenceContext context, string id, SequenceActionResult param, <u>EventMessage</u> eventMessage);	발생된 이벤트에 따라서 다음 Node가 결정된다.
Event Acceptor	AddEventAcceptor(string id, DoAcceptEvent action, int timeout = -1);	SequenceActionResult DoAcceptEvent(SequenceContext context, string id, SequenceActionResult param, EventMessage eventMessage);	대기 했다가 이벤트가 발생될 경우 실행된다.
EventQ	AddEventQ(string id, int concurrencyCount = 1);	없음	Queue에 넣고 처리할 내용이 있을 경우 사용된다. concurrencyCount : Queue에 명령이 쌓일 경우 해당 수만큼 동시실행 된다.
Event Sender	AddEventSender(string id, DoSendEvent action);	<u>object</u> DoSendEvent(SequenceContext context, string id, SequenceActionResult param);	Action함수 수행 직후 Event를 발생시킨다.
Fork	AddFork(string id);	없음	동시 실행을 위한 fork 노드를 만든다. child 노드들을 fork 노드에 연결시킨다.
Join	AddJoin(string id); ※ fork 예제 참고	없음	fork와 반대이다. child 노드들을 합해서 join 노드로 연결시킨다. Join과 연결된 Node들의 실행결과 (SequenceActionResult) Dictionary를 Join 다음 Node로 전달해준다.
Merge	AddMerge(string id);	없음	UML 문법을 맞추기 위해서 추가된 Node. 그 외 다른 작업은 수행하지 않음.
Subsequence	AddSubsequence(string id, <u>SequenceBuilder</u> builder);	없음	

3. 예제 Sequence 코드들과 Draw로 그린 pdf 들

```
public class Seq02
{
    public static SequenceBuilder Builder()
    {
        SequenceBuilder sb = new SequenceBuilder(1000);
        // Node를 생성한다.
        sb.AddAction("Act1", Act1Func);
        sb.AddAction("Act2", Act2Func);
        sb.AddAction("Act3", Act3Func);

        // Flow를 그린다.
        sb.Flows(sb.Init > sb["Act1"] > sb["Act2"] > sb["Act3"] > sb.Final);

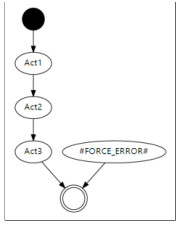
        // 순서도를 pdf로 그린다.
        sb.Draw("Seq02.pdf");

        return sb;
    }

    // 위에서 사용할 delegate 함수를 구현한다.
    private static SequenceActionResult Act3Func(SequenceContext context, string id, SequenceActionResult param)
    {
        return SequenceAction.Ok();
    }

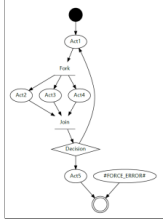
    private static SequenceActionResult Act2Func(SequenceContext context, string id, SequenceActionResult param)
    {
        return SequenceAction.Ok();
    }

    private static SequenceActionResult Act1Func(SequenceContext context, string id, SequenceActionResult param)
    {
        return SequenceAction.Ok();
    }
}
```



```

sb.Flows(sb.Init > sb["Act1"] > sb["Fork"] > sb["Act2"] > sb["Join"] > sb["Decision"] > sb["Acts"] > sb.Final)
.Flows(sb["Fork"] > sb["Act3"] > sb["Join"])
.Flows(sb["Fork"] > sb["Act4"] > sb["Join"])
.Flows(sb["Decision"] > sb["Act1"]);
  
```



4. Sequence API

1. int Start(object initParam = null, Action<SequenceActionResult> onCompleted = null);
 - a. 실행방법
 - i. SequenceButton을 누르면 해당 API가 수행된다.
 - ii. AuroraEnv에서 Sequence를 받아와서 직접 수행할 수 있다.
 1. 이 경우 initParam과 onCompleted 두 parameter를 사용할 수 있다.
 - a. initParam - 첫 번째 Node의 param으로 전달할 수 있다.
 - b. onCompleted - delegate를 등록하여 Sequence가 끝날 때 수행되게 할 수 있다.
2. int Stop();
 - a. 실행방법
 - i. SequenceButton을 누르면 해당 API가 수행된다.(처음 누르면 상태가 Stop버튼으로 변경된다.)
 - ii. AuroraEnv에서 Sequence를 받아와서 직접 수행할 수 있다.
 - b. Stop()이 수행되면 OnStopped 이벤트가 수행된다. (이벤트 항목에서 설명)
3. SequenceFlowed : 시퀀스의 Flow를 Callback으로 받을 수 있다.

```

private readonly ISequenceEntry Sequence;
private ISequence _MainSeq;

internal UCMainControlViewModel(IAuroraEnvironment aurora)
{
    Sequence = aurora.SequenceEntry;
    _MainSeq = Sequence[SequenceIds.MainManual];
}

private bool StartMainSequence(bool bConfirmUser = true)
{
    SeqParam = new MainParam();
    SeqParam.UseStepMode = UseStepMode;

    _MainSeq.SequenceFlowed += OnFlowed;
    _MainSeq.Start(SeqParam, OnComplete);

    EventPublisher.Publish(new EventMessage("", EquipEvent.SEQ_MAIN_START, true));

    return true;
}

private void ForceStop(object obj)
{
    _MainSeq.Stop();
}

private void OnFlowed(object sender, (string from, string nodeId, bool? activated) e)
{
}

private void OnComplete(SequenceActionResult obj)
{
    if (obj != null && obj.Error == E1.OK) {
        Dispatcher.Invoke(() => {
            IsRunning = false;
            LoadingFlowed = "Success.";
        });
    } else {
        Dispatcher.Invoke(() => {
            IsRunning = false;
            LoadingFlowed = "Fail.";
        });
    }
}
  
```

5. Sequence 이벤트

1. 두 가지 이벤트가 있으며 등록방법은 다음과 같다.

```

SequenceBuilder sb = new SequenceBuilder(1000);
sb.OnStopped(OnStopped);
sb.OnForceError(OnForceError);
  
```

2. OnStopped
 - a. 위 API 항목에서 언급한 대로 Stop() 수행 시 발생된다.
 - b. handler는 다음과 같다. (SequenceContext를 이용할 수 있다.)

```
private static void OnStopped(SequenceContext obj)
{
    logger.E("OnStopped");
}
```

3. OnForceError
- Node의 handler함수가 null을 return하면 다음 Node수행 전에 ForceError Node로 가고 Sequence는 끝이 나는데 이 때 발생된다.
 - ForceError Node로 보내고 싶을 때는 Node handler의 return을 null로 보내면 된다.
 - 참고로 Node의 handler 안에서 crash발생 시 return null이 되므로 역시 OnForceError가 발생된다.
 - 예외적으로 EventSend Node는 null을 return해도 발생되지 않는데 해당 값은 EventScept가 받아서 사용하기 때문이다.
 - handler는 다음과 같다. (SequenceContext를 이용할 수 있다.)

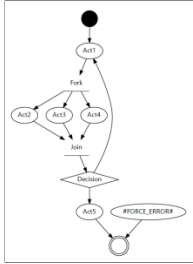
```
private static void OnForceError(SequenceContext obj, string arg2)
{
    logger.E("OnForceError");
}
```

→ arg2 : 에러가 난 Node Id를 갖고 있다.

Sequence Node 종류 및 사용법

1. Flows 연결 방법

- Node를 알아보기 앞서 Node를 연결하는 방법을 먼저 기술한다.
- sb.Flows()는 SequenceBuilder를 반환하기 때문에 아래와 같은 문법이 가능하다.
- Node를 먼저 그리고 > 를 이용하여 Node간 선을 연결하듯이 정의한다.
- Node를 연결하는 선을 중복해서 그려도 실제 수행 시에는 하나로 처리된다.



- sb.Flows()는 다음과 같은 구조를 가진다.

```
SequenceBuilder Flows(params Edge[] ..)
```

- 사용법

- Edge간 연결은 ">" 이용

```
sb.Flows(sb.Init > sb["Act1"] > sb["Fork"] > sb["Act2"] > sb["Join"]);
```

- Flows()함수 내 param을 ,로 구분

```
sb.Flows(sb.Init > sb["Act1"],
sb["Fork"] > sb["Act2"] > sb["Join"]);
```

- SequenceBuilder의 API이면서 SequenceBuilder를 리턴하기 때문에 다음과 같이 사용 가능하다.

```
sb.Flows(sb.Init > sb["Act1"]);
sb.Flows(sb["Act1"] > sb["Fork"]);
```

- 나눠서도 가능하다.

```
sb.Flows(sb.Init > sb["Act1"]);
sb.Flows(sb["Act1"] > sb["Fork"]);
```

예제 >

```
sb.Flows(sb.Init > sb["Act1"] > sb["Fork"] > sb["Act2"] > sb["Join"])
    .Flows(sb["Fork"] > sb["Act3"] > sb["Join"])
    .Flows(sb["Fork"] > sb["Act4"] > sb["Join"] > sb["Decision"] > sb["Act1"])
    .Flows(sb["Decision"] > sb["Act5"] > sb.Final);
```

2. Node의 종류

이름	설명	Diagram	용법
Action	- 가장 기본적인 Node - Return 값 : SequenceActionResult - 다음 Sequence의 SequenceActionResult param 으로 전달 - Error를 Return하더라도 Sequence가 멈추지는 않는다. (다음 Sequence에서 Error 내용을 가지고 판단하도록 한다.)		- Builder 함수 내<pre>sb.AddAction("Act1", Act1Func); sb.Flows(sb.Init > sb["Act1"] > sb["Act2"] > sb["Act3"] > sb.Final);</pre> - Handler 함수 - Params - context - SequenceContext - id - 자신의 Node ID - param - 이전 Node 수행결과 - Return : SequenceActionResult <pre>sb.AddAction("Act1", Act1Func); private static SequenceActionResult Act1Func(SequenceContext context, string id, SequenceActionResult param)</pre>

이름	설명	Diagram	용법
Decision & Merge	- 다음에 수행될 Nodes의 ID를 return한다. - 연결되지 않은 Node로 분기 시도 시 자신의 ID로 OnForceError 발생시킴 - Merge는 아무일도 수행하지 않지만 UML 규칙상 넣어두었음. (안 넣어도 동작은 동일함)		<pre>{ return SequenceAction.Ok(); }</pre> - Builder 함수 내 <pre>sb.AddDecision("Decision1", Decision1Func); private static string Decision1Func(SequenceContext context, string id, ICollection<string> branches, SequenceActionResult param) { return "Act2"; }</pre> - Handler 함수 - Params - context - SequenceContext - id - 자신의 Node ID - branches - 분기 가능한 Node ID들 - param - 이전 Node 수행결과 - Return : 분기될 다음 Node의 ID <pre>private static string Decision1(SequenceContext context, string id, ICollection<string> branches, SequenceActionResult param) { if (DateTime.Now.Second % 2 == 0) return "Act2"; else return "Act3"; }</pre>
Fork/Join	- Fork는 여러 Node를 동시수행 시에 사용한다. - Fork 후에는 Join하여 여러 경로의 수행결과를 기다린다. - Fork와 Join은 handler함수를 만들지 않는다.		- Builder 함수 내 <pre>sb.AddFork("Fork"); sb.AddJoin("Join"); sb.Flows(sb.Init > sb["Act1"] > sb["Fork"] > sb["Act2"] > sb["Join"] > sb["Act5"] > sb.Final) .Flows(sb["Fork"] > sb["Act3"] > sb["Join"]) .Flows(sb["Fork"] > sb["Act4"] > sb["Join"]);</pre> - Handler 함수 - 별도로 존재하지 않는다. - 단, Join은 Join 바로 전 Node들(Act2, Act3, Act4)의 결과를 Dictionary로 다음 Node(Act5)의 SequenceActionResult param에 넣어서 보낸다. (Act5에서 Act2, Act3, Act4의 결과를 받아서 처리할 수 있음)
SubSequence	- SubSequence를 Sequence와 동일한 방법으로 작성가능하다. - handler 함수대신 SequenceBuilder(subSb)를 넣는다.		- Builder 함수 내 <pre>SequenceBuilder sb = new SequenceBuilder(1000); SequenceBuilder subSb = new SequenceBuilder(1000); sb.AddAction("Act1", Act1); sb.AddAction("Act2", Act2); sb.AddAction("Act3", Act3); sb.AddFork("Fork"); sb.AddSubsequence("Sub", subSb); subSb.AddAction("Sub1", Sub1); subSb.AddAction("Sub2", Sub2); subSb.AddAction("Sub3", Sub3); subSb.Flows(subSb.Init > subSb["Sub1"] > subSb["Sub2"] > subSb["Sub3"] > subSb.Final); sb.Flows(sb.Init > sb["Act1"] > sb["Fork"] > sb["Act2"] > sb["Act3"] > sb.Final) .Flows(sb["Fork"] > sb["Sub"] > sb["Act3"]);</pre> - 주의 : 서비스웍스 시작 시 스레드 생성이 발생되므로 과도한 서비스웍스의 사용은 성능에 부정적인 영향을 줄 수 있다. - 간단한 동작은 서비스웍스보다 API를 만들어 사용하기를 권한다. - 빠르게 루프를 도는 경우 가급적 해당 루프안에 서비스웍스를 넣지 않는다. - Depth는 4단계 이하가 적당함.
Event Send	- EventSend Node의 handler가 수행된 후 정해놓은 이벤트를 발생시킨다.(Send(EventID)) - 이때 handler의 return 값은 발생되는 이벤트의 EventMessage 상에서 evtParam이 된다. - Flow를 그릴 때 해당 Node뒤에 .Send(EventID)를 붙여서 해당 Node가 수행된 후에 어떤 이벤트를 보낼지 정한다. - Event Send는 Event Acceptor와 함께 사용여도 되고 context.EventObservable로 직접 Event를 받아도 된다.		- Builder 함수 내 <pre>sb.AddEventSender("send_event", SendEvent1); sb.Flows(sb.Init > sb["act_1"] > sb["send_event"].Send("evt_trigger") > sb["act_2"] > sb.Final);</pre> - Handler 함수 - Params - Action Node와 동일(context, id, param) - Return - object를 return하며 Event받는 쪽의 evtParam이 된다. <pre>object SendEvent1(SequenceContext context, string id, SequenceActionResult param) { return null; }</pre>

이름	설명	Diagram	용법
Event Acceptor	- Node의 handler가 수행된 후 정해놓은 Event ID로 Event를 받아야 이후 Node가 수행된다. (OnEventID) - Event를 보낸 곳에서 evtParam을 보내면 이를 handler에서 받을 수 있다. - 정해놓은 대기시간이 지나도 handler가 수행되며 이 때는 evtParam에 null이 들어있다. - Event Send는 Event Acceptor와 함께 사용해도 되고 context.context.EventPublisher로 직접 Event를 발생시켜도 된다.		- Builder 함수 내 <pre>sb.AddEventAcceptor("accept_event", AcceptEvent1); sb.Flows(sb.Init > sb["act_1"] > sb["accept_event"].On("evt_trigger") > sb["act_2"] > sb.Final);</pre> - Handler 함수 - Params - context - id - param - eventMessage - Event를 보낸 곳에서 전달한 EventMessage, timeout 발생 시 null - Return - SequenceActionResult <pre>SequenceActionResult AcceptEvent1(SequenceContext context, string id, SequenceActionResult param, EventMessage eventMessage) // when timeout, it will be null { return null; }</pre> *. Acceptor의 경우 Flow의 NextNode를 끊을 수 없다. 연속으로 연결할것.
EventQ	- EventQ Node는 우측 순서도와 같이 어떤 Node로부터 연결되어 오지 않는다. - 시퀀스 내에서 File에 결과를 출력한다든지 다 수행되고 버리는 시퀀스를 EventQ로 보내서 처리할 수 있다. 즉, act1의 결과를 Queue로 보내서 handle_event는 File에 출력하고 버리는 예제이다.		Builder 함수 내 <pre>sb.AddAction("act_1", Act1); sb.AddEventSender("event_sender", SendEvent1); sb.AddDecision("decide_continue", Decide1); sb.AddEventQ("event_q", 3); sb.AddAction("handle_event", Act1); sb.Flows(sb.Init > sb["act_1"] > sb["event_sender"].Send(TEST_EVENT_ID) > sb["decide_continue"] > sb["act_1"], sb["decide_continue"] > sb.Final); sb.Flows(sb["event_q"].On(TEST_EVENT_ID) > sb["handle_event"] > sb.FlowFinal);</pre>
DecisionByEvent	이벤트에 의해 다음 Node를 선택한다. TimeOut이 발생하면 TimeOut이후에 수행할 Node를 수행할 수 있다.		- Builder 함수 내 <pre>sb.AddAction("act_1", Act1); sb.AddAction("act_2", Act1); sb.AddAction("act_3", Act1); sb.AddAction("act_4", Act1); sb.AddAction("act_5", Act1); sb.AddAction("act_6", Act1); sb.AddDecisionByEvent("decision", // id AcceptEvent1, // action 3 * 1000 // timeout); sb.AddMerge("merge"); sb.Flows(sb.Init > sb["act_1"] > sb["decision"], sb["decision"].On("evt_trigger") > sb["act_2"] > sb["merge"], sb["decision"].On("evt_move") > sb["act_3"] > sb["act_4"] > sb["merge"], sb["decision"].On("evt_cancel") > sb["act_5"] > sb["merge"], sb["decision"].OnTimeout() > sb["act_5"], sb["merge"] > sb["act_6"] > sb.Final);</pre> - Handler 함수 <pre>SequenceActionResult AcceptEvent1(SequenceContext context, string id, SequenceActionResult param, EventMessage eventMessage) // when timeout, it will be null { return null; }</pre>
Init Node	1. 최초 시작 노드		
Final Node	1. 마지막 노드		
ForceError Node	- 노드종료 후 다음노드 호출 시 이전 노드의 return 값이 null인 경우 수행 됨. - 노드 수행 중 예외 발생 시에는 자동적으로 null이 return되므로 이 경우에도 수행 됨. - 이렇게 종료되는 경우 Sequence의 OnForceError 이벤트에서 관련 처리동작 수행가능		- OnForceError 등록방법 <pre>SequenceBuilder sb = new SequenceBuilder(1000); sb.OnForceError(OnForceError);</pre> - OnForceError 함수 - SequenceContext를 이용할 수 있다. <pre>private static void OnForceError(SequenceContext obj) { logger.E("OnForceError"); }</pre>

→ TimeOut Node : EventAcceptor, DecisionByEvent, EventWaitAcceptor

3. SequenceActionResult

a) SequenceActionResult는 다음 형태를 가진다. (SequenceActionResult는 다른 Node에서도 사용되며 여기서만 설명한다.)

```
public class SequenceActionResult
{
    public SequenceActionResult(int err, object param = null);

    public int Error { get; }
    public object Param { get; }
}
```

b) 사용 방법

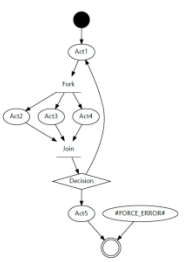
i. SequenceAction.OK(), SequenceAction.Error(100) 등으로 Error Number를 return할 수 있다.

```
return new SequenceActionResult(0);
return new SequenceActionResult(0, ParamObject);
return new SequenceActionResult(100, ParamObject); // Error Number is 100
return SequenceAction.Ok(); // same as first
return SequenceAction.Ok(ParamObject); // same as second
return SequenceAction.Error(100, ParamObject); same as third
```

4. Node 응용

a) Loop

i) 특별히 Loop를 위한 Node는 없고 Decision Node를 통해 과거 Node로 보내면 Loop를 돌게 된다.



```
sb.AddDecision("Decision", DecisionFunc);

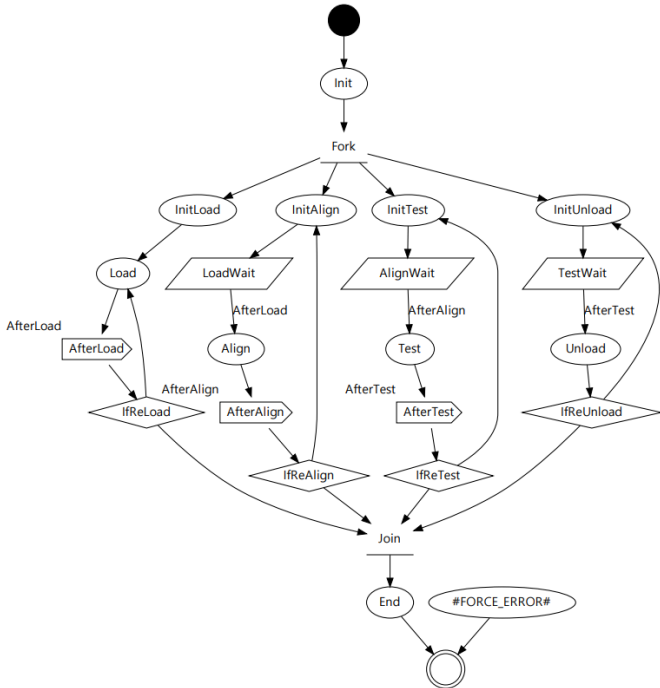
private static string DecisionFunc(SequenceContext context, string id, ICollection<string> branches, SequenceActionResult param)
{
    int retry = context.GetValue<int>("Retry");
    if (retry > 10)
        return "Act5";
    else
        return "Act1";
}
```

b) 단계별로 수행하되 각 단계가 동시 수행 되려면...

i. Load, Align, Test, Unload를 수행하는 장비가 있다.

ii. 각 단계는 직전 단계가 끝나면 수행되고 제품은 계속 투입되므로 각 단계가 쉬어서는 안 된다.

- 1. 4개의 단계를 Fork한다.
- 2. 각 단계는 이전 단계가 끝나기를 기다린다.
- 3. 자신이 끝나면 다음 단계에게 Event를 보낸다.
- 4. Load는 첫 단계이므로 Event를 기다릴 필요가 없다.
- 5. Unload는 마지막 단계이므로 Event를 보낼 필요가 없다.
- 6. 각자 Loop를 돈다.



```
sb.AddAction("Init", InitFunc);
sb.AddAction("InitLoad", InitEach);
sb.AddAction("InitAlign", InitEach);
sb.AddAction("InitTest", InitEach);
```

```
sb.AddAction("InitUnload", InitEach);
sb.AddAction("Load", LoadFunc);
sb.AddAction("Align", AlignFunc);
sb.AddAction("Test", TestFunc);
sb.AddAction("Unload", UnloadFunc);
sb.AddFork("Fork");
sb.AddJoin("Join");
sb.AddAction("End", EndFunc);
sb.AddEventSender("AfterLoad", AfterLoadFunc);
sb.AddEventSender("AfterAlign", AfterAlignFunc);
sb.AddEventSender("AfterTest", AfterTestFunc);
sb.AddEventAcceptor("LoadWait", LoadWaitFunc, 600000);
sb.AddEventAcceptor("AlignWait", AlignWaitFunc, 600000);
sb.AddEventAcceptor("TestWait", TestWaitFunc, 600000);
sb.AddDecision("IfReLoad", IfReLoad);
sb.AddDecision("IfReAlign", IfReAlign);
sb.AddDecision("IfReTest", IfReTest);
sb.AddDecision("IfReUnload", IfReUnload);

// Flow를 그린다.
sb.Flows(sb.Init > sb["Init"] > sb["Fork"] > sb["InitLoad"] > sb["Load"] > sb["AfterLoad"].Send("AfterLoad") > sb["IfReLoad"] > sb["Join"], sb["IfReLoad"] > sb["Load"],
        sb["Fork"] > sb["InitAlign"] > sb["LoadWait"].On("AfterLoad") > sb["Align"] > sb["AfterAlign"].Send("AfterAlign") > sb["IfReAlign"] > sb["Join"], sb["IfReAlign"] > sb["InitAlign"],
        sb["Fork"] > sb["InitTest"] > sb["AlignWait"].On("AfterAlign") > sb["Test"] > sb["AfterTest"].Send("AfterTest") > sb["IfReTest"] > sb["Join"], sb["IfReTest"] > sb["InitTest"],
        sb["Fork"] > sb["InitUnload"] > sb["TestWait"].On("AfterTest") > sb["Unload"] > sb["IfReUnload"] > sb["Join"] > sb["End"] > sb.Final, sb["IfReUnload"] > sb["InitUnload"]));
```

1. Sequence Context Property

```
public class SequenceContext
{
    public bool Stopped { get; }
    public string SequenceId { get; }
    public IDeviceEntry Devices { get; }
    public IRecipeManager Recipes { get; }
    public ISettingEntry Settings { get; }
    public ISequenceEntry Sequences { get; }
    public IEventPublisher EventPublisher { get; }
    public IObservable EventObservable { get; }

    public T GetValue<T>(string key);
    public void PutValue<T>(string key, T value);
    public bool RemoveValue(string key);
    public bool TryGetValue<T>(string key, out T value);
}
```

- Devices
 - 등록된 Device에 ID로 접근 가능하다.
- Recipes
 - 등록된 Recipes에 ID로 접근 가능하다.
- Settings
 - 등록된 Settings에 ID로 접근 가능하다.
- Sequences
 - 등록된 Sequences에 ID로 접근 가능하다.
- EventPublisher
 - [\[08\] 이벤트](#) 참고
 - Subject를 등록한 이벤트를 발생 시킨다.
 - UI등에서 받아서 처리가능

- ReadyToLoad 이벤트를 발생시키며 parameter로 true를 전달한다.

```
context.EventPublisher.Publish(new EventMessage("Act1Func", "ReadyToLoad", true));
```

- EventObservable
 - UI등에서 발생시간 이벤트를 Subject를 이용해 받을 수 있다.
 - UI에서 Load 이벤트를 발생시키면 이후 단계를 수행하기 위해 ready를 true로 변경하는 코드

```
IDisposable temp = context.EventObservable.AsObservable("Load").Subscribe(_ => ready = true);
```

- PutValue
 - Value를 저장하여 다른 Node에서 사용할 수 있게 한다.
 - IsAct1FuncRun에 true를 저장

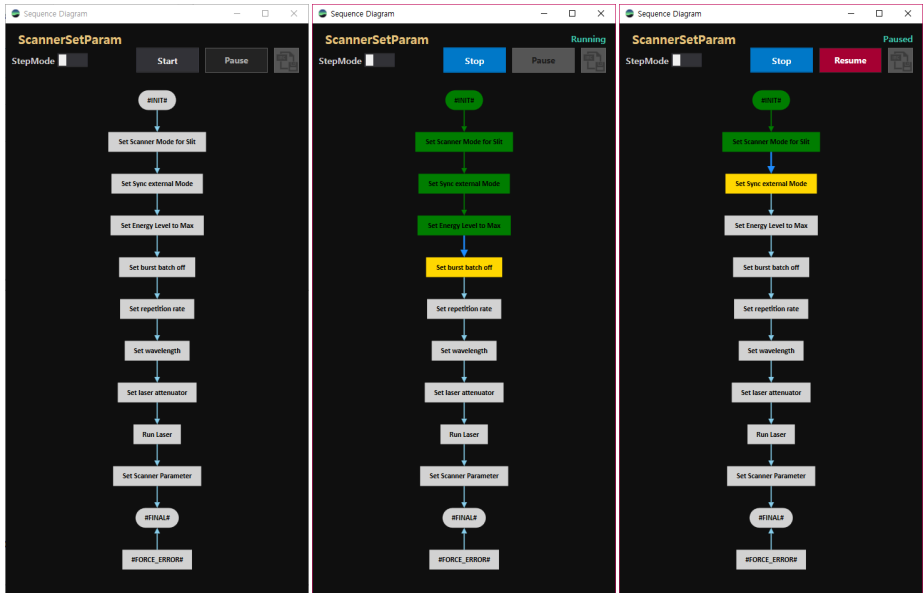
```
context.PutValue("IsAct1FuncRun", true);
```

- GetValue
 1. 특정 Node에서 넣어놓은 Value를 꺼내온다.
 2. IsAct1FuncRun의 값을 읽어서 bool 변수에 저장(반드시 Generic Type T를 사용해야 한다.

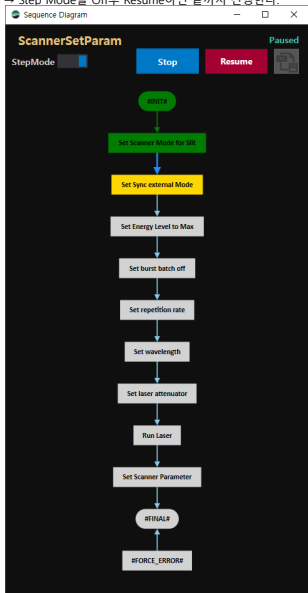
```
bool run = context.GetValue<bool>("IsAct1FuncRun");
```

SequenceDrawer

시퀀스를 시각적으로 표현하고 Flow를 보여주는 UI



- Start : 시퀀스 시작버튼. 누를시 Stop버튼으로 변경된다.
- Pause : 누르면 멈추고, Resume버튼으로 변경된다.
- Export 버튼 : 현재 시퀀스 상태를 이미지로 저장할 수 있다.
- 화살표 : 현재 진행중인 화살표가 더 강조되어있다.
- Step Mode : 한단계씩 진행하며 진행후 자동 Pause된다. Resume를 누르면 다음 단계로 진행한다.
→ Step Mode를 Off후 Resume하면 끝까지 진행한다.



- SequenceDrawer Dialog 띄우기

```
WinSequenceDiagram win = new WinSequenceDiagram();
if (_WorkingCondition.Device == eRepairDevice.Scanner) {
    win.SetParam(new SeqDrawerParam() {
        Builder = SequenceManager.GetBuilder(SequenceIds.SCANNER_SET_LASER_PARAM),
        Sequence = sequence,
        DiagramType = eDiagramType.Tree,
        LineType = AuroraDevUI.SequenceDrawer.eLineType.Curved,
        IsShowSubSequence = false,
        VisibleKeys = new List<string>()
    });
}

win.SequenceParam = _WorkingCondition;
win.Owner = Application.Current.MainWindow;
win.Width = 500;
win.ResultCallback += OnSeqResult;
win.ShowDialog();
```

→ SequenceParam : 시퀀스시작시 넣어줄 파라미터
→ ResultCallback : 시퀀스를 수행하면 원로시 결과값을 콜백으로 전달해준다.

```
private void OnSeqResult(int result)
{
    Application.Current.Dispatcher.InvokeAsync(() => {
        if (result == E1.OK) {
            _CurrentEnergy = _WorkingCondition.ENERGY;
            if (_WorkingCondition.Device == eRepairDevice.Slit) {
```

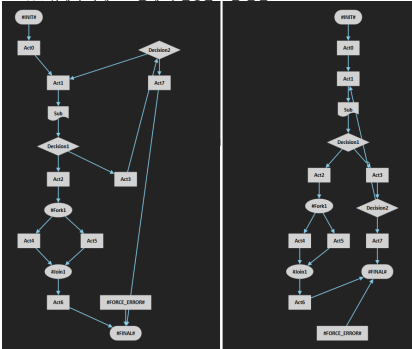
```

        CurrentFrequency = _WorkingCondition.FREQ;
        SlitX = _WorkingCondition.SLIT_X;
        SlitY = _WorkingCondition.SLIT_Y;
        SlitT = _WorkingCondition.SLIT_THETA;
    } else {
        CurrentFrequency = 1000;
    }
}
});
}

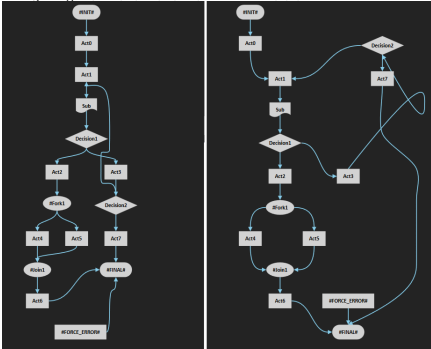
```

SeqDrawerParam

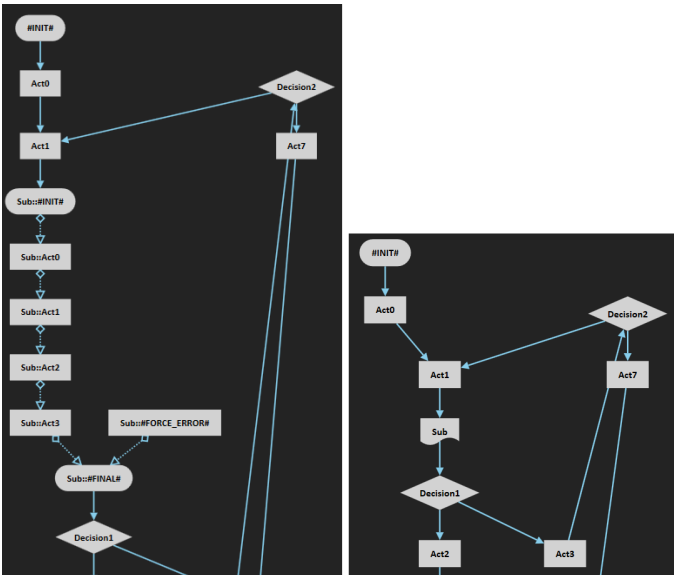
- Builder - 시퀀스 Flow를 정의한 Builder를 넣어줘야 한다.
- Sequence - `_SequenceManager.GetSequence(SequenceIds.SCANNER_SET_LASER_PARAM)` 처럼 시퀀스메니저로부터 넣어준다.
- DiagramType - Tree와 Layered가 있다.
→ 디폴트 (Layered): 그림의 배치에 신경을 더 쓰는 옵션
→ Tree: 위에서 아래로 흐름에 더 신경을 쓰는 옵션



- LineType
→ LineType을 Curved로 주면 Flow를 곡선으로 표시하고 Straight으로 주면 직선으로 표시한다.
→ DiagramType과 함께 적절히 어울리는 옵션을 선택하자.

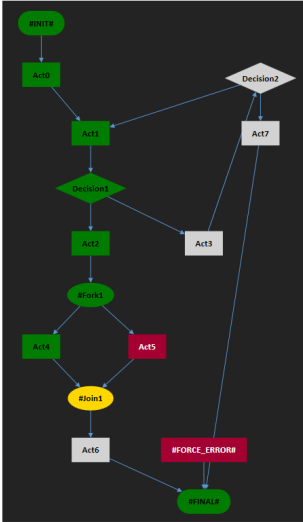


- IsShowSubSequence
→ Sub시퀀스를 접을지 펼칠지 결정한다. (보통은 복잡도로 인해 접는다.)

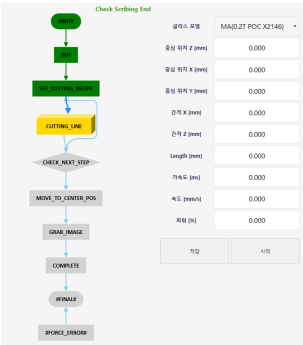


- VisibleKeys
→ SequenceContext의 Parameter이름 목록을 주면 시퀀스가 진행되는 동안 Parameter의 값을 표시해준다.

- Decision Node 표기와 Error발생시 화면



- 팝업에 CustomControl을 붙일 수 있다.



```
var focusReview = new FocusReviewView();
focusReview.VerticalAlignment = VerticalAlignment.Top;
focusReview.Width = 350;
focusReview.Height = 750;
focusReview.Margin = new Thickness(0, 20, 0, 0);

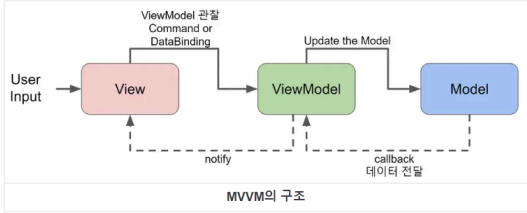
FocusReviewSeqWin = new WinSequenceDiagram();
FocusReviewSeqWin.SetParam(new SeqDrawerParam() {
    Builder = SequenceManager.GetFocusReviewBuilder(),
    Sequence = SequenceEntry[SequenceIds.FocusReviewSeq],
    DiagramType = eDiagramType.Tree,
    LineType = eLineType.Curved,
    IsShowSubSequence = false,
    VisibleKeys = new List<string>()
});
FocusReviewSeqWin.SetCustomControl(focusReview);
FocusReviewSeqWin.Closed += OnFocusReviewWinClosed;
FocusReviewSeqWin.Owner = Application.Current.MainWindow;
FocusReviewSeqWin.Width = 600;
FocusReviewSeqWin.Show();
```

[12] UI-View

[01] Aurora와 MVVM

1. 개요

- Model-View-view Model
- 출처 : 나무위키



- 아키텍처 패턴중 하나 : 데이터를 처리하는 모델, 사용자에게 보여지는 UI인 View, 뷰에 바인딩되어 모델과 뷰 사이를 이어주는 뷰모델로 분리하는 방식
- UI가 실제 처리 코드와 거의 완벽하게 분리된 것이 특징.

2. 장점

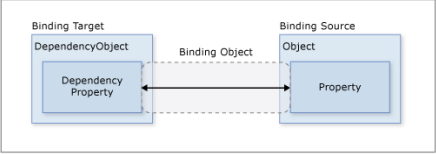
- 데이터 처리로직과 UI디자인과 상호 영향이 적어 개발이 용이하며, 개발자와 UI디자인자간의 협업이 용이하다.
→ 비팅과 개발팀 분리가 가능하고, 서로 각자의 영역에 집중하는데 유리하다.
- ViewModel이 UI와 분리되어있어 유닛테스트를 하기 유리하다.
- 패턴 구현을 위해 Command패턴과 데이터 Binding을 수행한다.
Command : 뷰에서 UI를 처리하기 위해 뷰 모델에 전달하는 요구사항
Data Binding : 뷰와 뷰모델사이의 데이터를 동기화시킨.
- WPF는 MVVM을 사용하여 프로그램을 쉽게 작성하기 위해 디자인되어있다. (Microsoft)
- View : ViewModel = N : 1이다.

3. 단점

- 1) 간단한 UI에서는 ViewModel 도입이 더 어려움이 생길 수 있다.
- 2) 표준화표명이 없어 사람마다 이해가 다르다 (같은 틀의 표준을 만들고 코딩하는 것이 중요)
- 3) 복잡해질수록 ViewModel 비대해지고, View표현하는 Presentation Logic이 늘어나며, Xaml을 재수정하여 코드가 난해해진다.

4. DataBinding

- View에 Data(Property)를 연결하는 프로세스이다.



Xaml

```
<Border Grid.Row="1" Grid.Column="1" BorderThickness="1 0 1 1" BorderBrush="#444444" Width="60">
    <TextBlock Foreground="White" Text="{Binding GlassCount}"/>
</Border>
```

ViewModel

```
class UCInformPanelViewModel : NotifyPropertyChangedBase, IDeviceInit
{
    private int glassCount = 0;
    public int GlassCount { get => glassCount; set => SetProperty(ref glassCount, value); }
    ...
}
```

→ Aurora에서는 **NotifyPropertyChangedBase**라는 BaseClass를 제공하고 **SetProperty**를 통해 INotifyPropertyChanged 이벤트가 발생하도록 소스코드를 간단하게 만든다.

5. ICommand

- Xaml에서 버튼, 라디오버튼, 체크박스버튼의 동작에 대한 이벤트를 처리하기 위한 인터페이스이다.
- ViewModel에서 손쉽게 Command를 View와 연결하기 위한 방법으로 RelayCommand를 사용하도록 권고한다.
(<https://learn.microsoft.com/ko-kr/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern>)
- Aurora에서는 이 RelayCommand를 사용하여 Command를 연결하는 방법을 같은 구조의 DelegateCommand를 사용하도록 제공한다.

```
class TestViewModel : NotifyPropertyChangedBase
{
    public ICommand CMD_ButtonClick { get; }
    public ICommand CMD_ResetCount { get; }

    public MainWindowViewModel()
    {
        CMD_ButtonClick = new DelegateCommand(ButtonCmdExe, CanButtonCmdExe);
        CMD_ResetCount = new DelegateCommand(ResetAlarm);
    }

    private void ButtonCmdExe(object param)
    {
        MessageBox.Show("버튼 클릭");
    }

    private bool CanButtonCmdExe(object param)
    {
        return true;
    }

    private void ResetAlarm(object obj)
    {
    }
}
```

→ Command Excute와 CanExcute 함수를 Delegate로 연결해준다.

6. Aurora ViewModel Class 작성 가이드 (표준화)

```
class UCSampleViewModel : NotifyPropertyChangedBase
{
    private static readonly Logger LOG = new Logger("Sample");

    public ICommand CMD_VisibleChanged { get; }
    public ICommand CMD_ResetAlarm { get; }
```

```

private int glassCount = 0;
public int GlassCount { get => glassCount; set => SetProperty(ref glassCount, value); }

private readonly DispatcherTimer _MonitorTimer;
private readonly IAuroraEnvironment _Aurora;

internal UCInformPanelViewModel(IAuroraEnvironment aurora)
{
    _Aurora = aurora;

    CMD_VisibleChanged = new DelegateCommand(VisibleChanged);
    CMD_ResetAlarm = new DelegateCommand(ResetAlarm);

    aurora.EventObservable.AsObservable(AuroraSysEvent.SYS_DEVICE_ACTIVATED)
        .Select(msg => msg.Param as string)
        .ObserveOnDispatcher()
        .Subscribe(x => OnDeviceActiveChanged(x));

    _MonitorTimer = new DispatcherTimer();
    _MonitorTimer.Tick += OnMonitoring;
    _MonitorTimer.Interval = TimeSpan.FromMilliseconds(100);
}

private void VisibleChanged(object obj)
{
    bool visible = (bool)obj;
    if (visible) {
        _MonitorTimer.Start();
    } else {
        _MonitorTimer.Stop();
    }
}

private void OnDeviceActiveChanged(string deviceName)
{
}

private void OnMonitoring(object sender, EventArgs e)
{
}

private void ResetAlarm(object obj)
{
}
}

```

- Logger의 사용 : 최 상단에 Logger를 선언한다.
- CMD 변수명 : "CMD_"로 작성하자.
- Private 변수명 규칙 : _소문자, _대문자를 사용하자.
- readonly속성인 경우 readonly를 붙이자.
- IAuroraEnvironment같은 플랫폼 Manager는 파라미터로 전달하자 .(가급적 Application에 접근하기 보다는 전달하자.)
- Visibility Change때에 따라 Loaded에 따라 모니터링 Timer를 시작/종지하라. (Event는 Dispose필요할 때가 있다.)
- Xaml에서의 예제

```

<behavior:Interaction.Triggers>
    <behavior:PropertyChangedTrigger Binding="{Binding IsVisible, ElementName=uc}">
        <behavior:InvokeCommandAction Command="{Binding CMD_VisibleChanged}" CommandParameter="{Binding IsVisible, ElementName=uc}"/>
    </behavior:PropertyChangedTrigger>
</behavior:Interaction.Triggers>

```

[13] BackUp

0. Overview

- Aurora에서 제공하는 문서를 참고하여 작성되었으며 해당 문서를 참고해도 된다.
- Aurora에서 제공하는 Auto Backup에 대해 알아본다.
- Aurora는 주기적으로(Daily) 특정폴더를 백업하고 압축하여 저장하는 기능을 제공한다.
- 특정 운영SW에서 발생하는 Output(이미지, 영상, 기타 데이터등)을 주기적으로 백업해놓을 수 있다.

1. Auto Backup

- 기능 활성화
 - 기능을 활성화 하기 위해서 먼저 Auto Backup View를 추가한다.

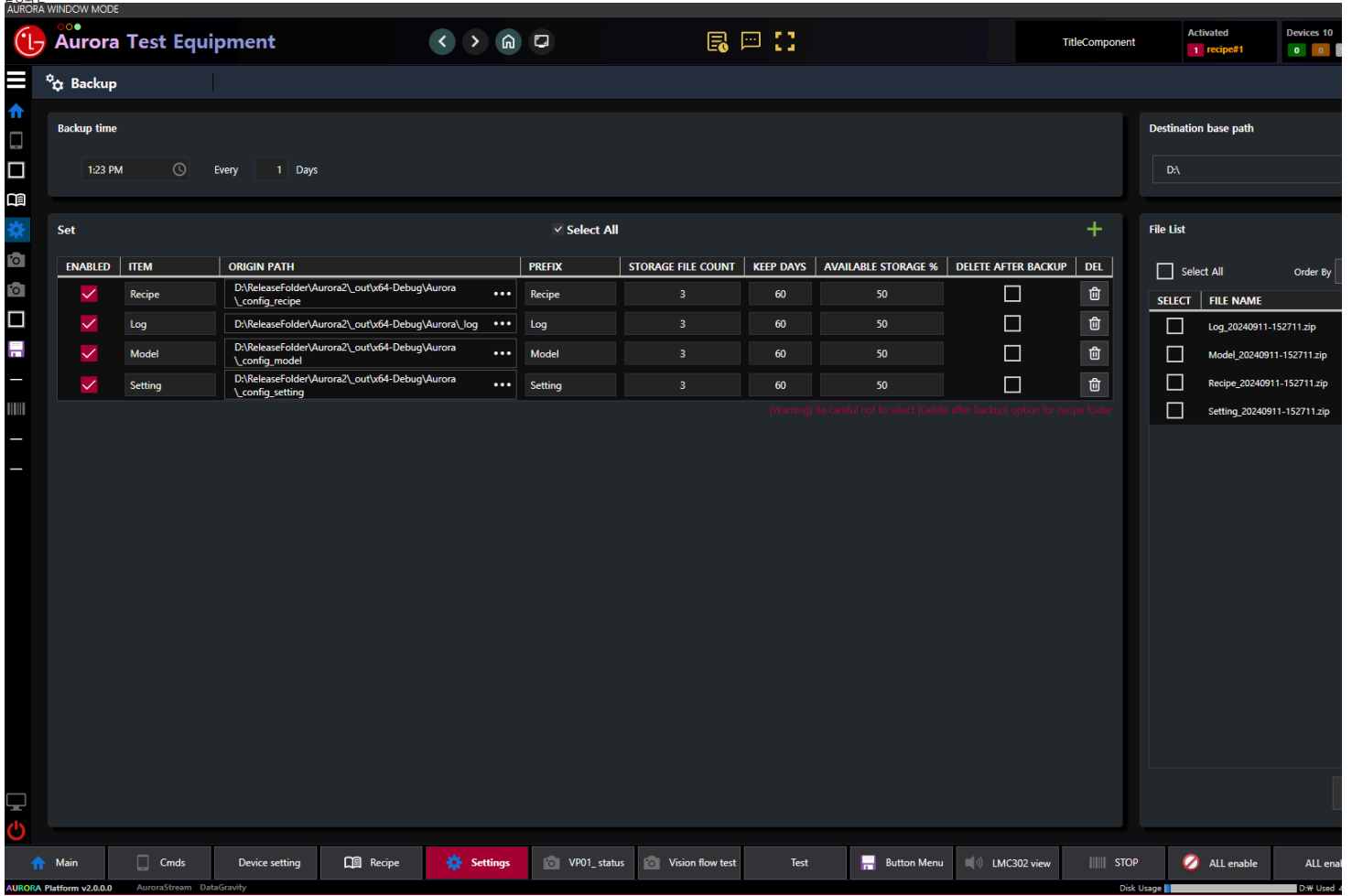
```

void IEquipmentSetup.RegisterViews(IViewManager manager)
{
    ...
    var autoBackupView = new Aurora.Framework.Managers.MainMenu("AutoBackupView", MenuIcon.GetMenuIcon(eMenuIcon.SAVE), PermissionLevel.LEVEL3);
    autoBackupView.AddSubMenu("AutoBackUPView", new AutoBackupView2 { DataContext = new AutoBackupViewModel() });
}

```

```
manager.RegisterMainMenu(autoBackupView) ;
...
}
```

• 실행 화면



- 좌측 상단 메뉴
 - Backup Time
 - Daily로 백업할 시간을 설정한다. 지나간 시간 설정시 다음날 백업
- 우측 상단 메뉴
 - Destination base path
 - 백업하여 저장할 폴더 위치를 선택한다.
- 좌측 하단 메뉴 : Backup Item 설정
 - Enabled
 - 백업 활성화 여부
 - Item
 - 백업에 대해 사용자가 인지할 이름을 정해준다. 여러가지백업을 지정할 수 있는데 이에 대한 구별자이다.
 - Origin Path
 - 실제 백업을 수행할 폴더 지정
 - Prefix
 - 백업하고 압축한 파일의 앞에 붙여줄 Prefix 이다. AuroraLog_20220210-111451.zip
 - Storage File Count (파일 저장 개수) : 최대로 저장할 파일 개수 (넘을 경우 오래된 파일부터 삭제)
 - Keep days (저장기간) : 저장할 기간 (넘을 경우 삭제)
 - Available Storage % (여유공간) : Disk공간 확인 (해당 로그가 백업되는 디스크의 공간이 부족할 경우 오래된 파일부터 삭제)
 - Delete After Backup (백업 후 삭제) : 백업 후, 백업 대상 폴더의 내용을 삭제
 - Del : 해당 Backup Item (한 줄)을 삭제
- 빨간색 표시 영역 : 좌→우 순서대로
 - Manual Backup (수동 백업) : 백업 설정대로 압축 파일 저장
 - Reload (불러오기) : 저장된 설정을 다시 불러오기
 - Save (설정 저장) : 현재 설정을 저장
- 우측 하단 메뉴 : 로그 리스트
- File List (파일목록) : 최근에 생성한 파일 목록을 보여줌.
- 아이콘 : 좌→우 순서대로
 - 설정 Reload
 - 설정저장
 - 수동백업 : 현재 설정된 대로 백업실시
 - 파일목록 Refresh
 - 파일 지우기
- ※ 삭제 순서
 - 1. KeepDays (보관일자) 체크
 - 2. NumOfStorageValue (파일 개수) 체크 : 개수 초과되면 오래된 것부터 삭제
 - 3. AvailableStoragePerc (파일 저장용량) 체크 : 용량 초과되면 오래된 것부터 삭제
- 관련 이벤트
AuroraSysEvent.SYS_REQ_MANUAL_BACKUP_START : 파라미터 string
→ 파라미터가 있을시 : 해당파라미터의 이름(Item)을 즉시 백업.
→ 파라미터가 없을시 : 현재 설정된 백업리스트를 즉시 백업

0. Overview

- Aurora 변동부의 Multi-Language 적용 방법에 대해서 확인합니다.
- Aurora 고정부에서 "LangTokens" 라는 Token 관리자가 구현되어 Runtime 에 언어가 변경되도록 지원합니다.

1. Multi-Language 구현 방법

- Aurora 고정부에서 LangTokens 라는 Token 관리자에 의해 지원합니다.
- LangTokens 를 Aurora App 의 Resource 에 Static Resource 로 등록됩니다.
- Resource 폴더에 StrToken.resx 와 언어에 맞는 resx 파일을 생성하여 동일한 Indexer 를 부여합니다.
- xaml 에서 TextBinding 시 StaticResource 를 참조하고 Indexer 에 의해 접근하여 Binding 됩니다.
- cs code 상에서는 FindStrToken 함수 혹은 등록된 "langTokens" Resources 에 Indexer 로 접근할 수 있습니다.

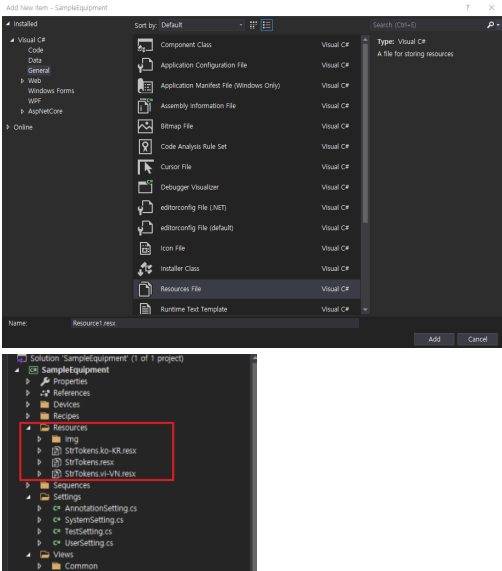
1.1 Resources File 생성 & 적용

- 각 언어로 된 Resources 파일을 생성합니다.
- Resources 파일들에 동일한 "Indexer" (key) 로 번역된 언어를 적어 줍니다.

- 장비 Project > Resources 폴더 생성

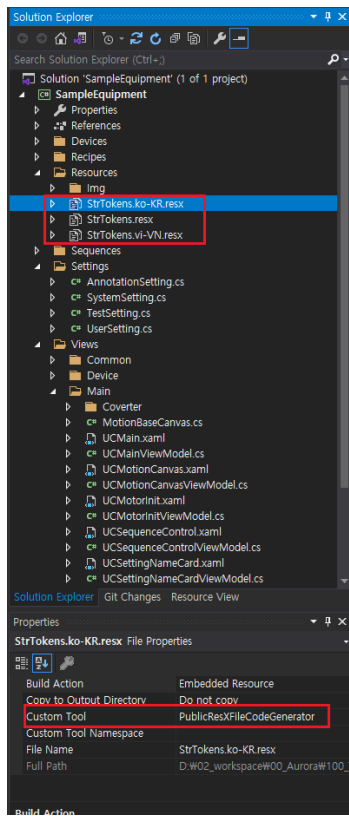
- Resources 폴더 > Add > New Item > Visual C# - General - Resources File

- 이름 형식
 - en-US : StrTokens.resx
 - ko-KR : StrTokens.ko-KR.resx
 - vi-VN : StrTokens.vi-VN.resx



→ 실습할때는 StrTokens라고 쓰지 마세요. 실수

- Custom Tool 이 다른 이름으로 되어 있거나 비어 있으면, 아래와 같이 수정해 주시기 바랍니다.
 - Custom Tool : PublicResXFileCodeGenerator



- 아래와 같이 Resources 파일들에 동일한 "Indexer" (key) 로 번역된 언어를 적어 줍니다.

<StrTokens.resx>

	Name	Value	Comment
▶	TokenTest	Token Test	
*			

<StrTokens.ko-KR.resx>

	Name	Value	Comment
▶	TokenTest	토큰 테스트	
*			

- Language Resources 를 사용할 수 있도록 EquipmentSetup.cs 에 등록해 줍니다.

```
public void BuildMainStatusView(MainStatusBuilder builder) {
    builder
        .ShowCI(true)
        .ShowClock(true)
        .AddDeviceActiveStatus(DeviceIds.Scale)
        .AddLanguage("ko-KR", "en-US", "vi-VN");
}

public void RegisterLanguageResource(LangResourceRegister register) {
    register(typeof(StrTokens));
}
```

주의 : register(typeof(**StrTokens**))에서 StrTokens 클래스는 본인이 위에서 추가한 StrTokens.resx파일이 자동생성한 Class타입이다.
간혹 플랫폼 쪽에서 정의된 StrTokens타입을 사용할 경우 다국어가 오동작하는 경우가 있다.

1.2 Xaml 에서의 접근

- "LangTokens" 라는 Token 관리자를 생성하여, StaticResource 에 "langTokens" 라는 이름으로 등록되어 있습니다.
- 이 StaticResource 에 Indexer 로 접근하여 적용 합니다.
- 아래는 예제 code 임니다.

```
<TextBlock Text="{Binding [TokenTest1], Source={StaticResource langTokens}, FallbackValue=TokenTest1}"></TextBlock>
<TextBlock >
    <Run Text="{Binding [TokenTest2], Source={StaticResource langTokens}, FallbackValue=TokenTest2}" />
    <Run Text="{Binding [TokenTest3], Source={StaticResource langTokens}, FallbackValue=TokenTest3}" />
    <Run Text="Test" />
</TextBlock>
```

- FallbackValue는 Key와 동일한 값을 써준다.

1.3 cs 에서의 접근

- "LangTokens" 라는 Token 관리자를 생성하여, StaticResource 에 "langTokens" 라는 이름으로 등록되어 있습니다.
- FindStrToken 함수 혹은 등록된 "langTokens" Resources 에 Indexer 로 접근할 수 있습니다
참고로, FindStrToken 함수를 사용하는 것이 일반적입니다.

```
// 1) FindStrTokens 를 통해 접근
var token1 = Application.Current.FindStrToken("TokenTest1");

// 2) Indexer 로 접근
var tokenResource = Application.Current.Resources["langTokens"] as LangTokens;
var token2 = tokenResource["TokenTest2"];
var token3 = tokenResource["TokenTest3"];
```

2. Language Changed Event

- Aurora 는 언어 변경을 **AuroraSysEvent.SYS_LANGUAGE_CHANGED** 이벤트를 통해 변경합니다.
- Aurora 변동부에서 이벤트 구독은 물론 소멸될 때 구독을 취소해줘야 하는 경우가 있으므로 IDisposable 로 처리해주는 것이 좋다. (이벤트 참고)

```
// AuroraEnv.EventObservable 을 통해 Subscribe

void ObserveLanguageChanged(
    IEventObservable eventObservable)
=> eventObservable.AsObservable(
    AuroraSysEvent.SYS_LANGUAGE_CHANGED
).ObserveOn(
    Application.Current.Dispatcher
).Subscribe(
    msg =>
    {
        LOG.D($"Language Changed : {msg.Param}");

        var langTokens = Application.Current.Resources["langTokens"] as LangTokens;
        var views = ViewManager.GetInstance().GetRegisteredViews();

        for (int i = 0; i < views.Count; i++)
        {
            (MainTab.Items[i] as TabItem).Header =
                Application.Current.FindStrToken(views[i].Item1);
        }
    }
);
```

[15] Alarm

0. Overview

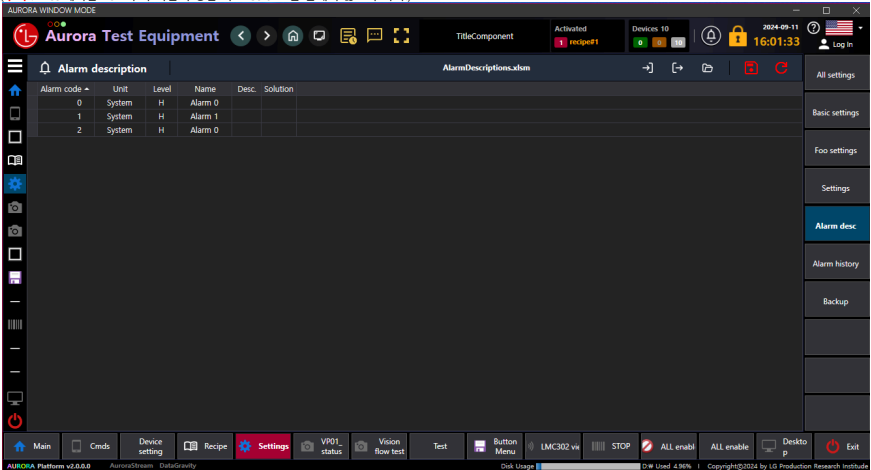
- Alarm기능 사용법을 기술한다.
 - Alarm View 추가하는 방법
 - Alarm Description 추가 및 수정방법
 - SetAlarm, Alarm Events

1. Alarm View

- View의 종류와 사용법, 등록방법을 다룬다.

1.1 View의 종류 및 사용방법

- Alarm관련 View는 다음과 같다.
 - Alarm Description View - Alarm Description을 관리하는 화면
각 아이콘은 순서대로 다시로드, 저장, 내보내기, 가져오기, Detail 폴더열기이다
UI에서 Description을 추가하고자 할 때는 ALARM CODE항목을 클릭하고 추가할 코드번호를 넣으면 된다.
UI에서는 간단한 수정만 하고 대량 수정을 위해서는 Excel로 파일을 열어서 수정한다.
(주의 : Excel에서는 Row추가 혹은 수정만 하고 Column은 손대지 않도록 하자.)



- Excel File
Detail폴더를 열어서 한 Depth 올라가면 Alarm Description엑셀파일이 있다.
열면 아래와 같이 나오고 Row를 대량으로 추가 및 수정이 가능하다.
(주의 : Excel file이 열려있는 경우 로딩 실패가 발생되며 팝업으로 알려준다. 이 경우 장비를 껐다 켜거나 Alarm Description View에서 reload해야 한다.)

	A	B	C	D	E	F	G
1	Code	Unit	Level	No	Name	Description	Solution
2	0	System	H	-1	Alarm 0	This is Alarm 0	Do Solution 0
3	1	System	H	-1	Alarm 0	This is Alarm 1	Do Solution 1
4	2	System	H	-1	Alarm 0	This is Alarm 2	Do Solution 2
- Alarm History View - 당일에 발생한 Alarm 로그 뷰
Alarm이 발생되거나 Reset될 때 기록되며 File에 저장된다.(지정한 DataPath의 _logAlarm 폴더에 csv로 저장)
Application을 껐다 켤 경우 당일 기록된 Alarm를 모두 표시 해 준다.

State	Set Time	Reset Time	Alarm code	Level	Unit	Name
RESET	2024-09-11 16:05:24.720	2024-09-11 16:05:27.202	58	H	Unknown	
RESET	2024-09-11 16:05:24.720	2024-09-11 16:05:27.202	57	H	Unknown	
RESET	2024-09-11 16:05:24.720	2024-09-11 16:05:27.201	56	H	Unknown	
RESET	2024-09-11 16:05:24.720	2024-09-11 16:05:27.201	55	H	Unknown	
RESET	2024-09-11 16:05:24.720	2024-09-11 16:05:27.201	54	H	Unknown	
RESET	2024-09-11 16:05:24.720	2024-09-11 16:05:27.201	53	H	Unknown	
RESET	2024-09-11 16:05:24.720	2024-09-11 16:05:27.201	52	H	Unknown	
RESET	2024-09-11 16:05:24.720	2024-09-11 16:05:27.201	51	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.201	50	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.200	49	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.200	48	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.200	47	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.200	46	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.200	45	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.200	44	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.200	43	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.200	42	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.199	41	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.199	40	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.199	39	H	Unknown	
RESET	2024-09-11 16:05:24.719	2024-09-11 16:05:27.199	38	H	Unknown	
RESET	2024-09-11 16:05:24.718	2024-09-11 16:05:27.199	37	H	Unknown	
RESET	2024-09-11 16:05:24.718	2024-09-11 16:05:27.199	36	H	Unknown	
RESET	2024-09-11 16:05:24.718	2024-09-11 16:05:27.199	35	H	Unknown	

- Alarm History Popup View - 단일 이외의 Alarm을 보기위한 뷰
Alarm History View의 View버튼 클릭 시 Popup형식으로 표시되며 보고서 하는 Alarm의 기간을 설정하여 볼 수 있다.

STATE	SET TIME	RESET TIME	ALARM CODE	LEVEL	UNIT	NAME
SET	2024-09-11 16:04:33.042		0	H	System	Alarm 0
RESET	2024-09-11 16:04:33.042	2024-09-11 16:04:34.674	0	H	System	Alarm 0
SET	2024-09-11 16:04:34.640		1	H	System	Alarm 1
RESET	2024-09-11 16:04:34.640	2024-09-11 16:04:38.339	1	H	System	Alarm 1
SET	2024-09-11 16:05:00.303		1	H	System	Alarm 1
RESET	2024-09-11 16:05:00.303	2024-09-11 16:05:07.936	1	H	System	Alarm 1
SET	2024-09-11 16:05:09.767		0	H	System	Alarm 0
RESET	2024-09-11 16:05:09.767	2024-09-11 16:05:12.533	0	H	System	Alarm 0
SET	2024-09-11 16:05:24.715		0	H	System	Alarm 0
SET	2024-09-11 16:05:24.715		1	H	System	Alarm 1
SET	2024-09-11 16:05:24.715		2	H	System	Alarm 1
SET	2024-09-11 16:05:24.715		3	H	Unknown	
SET	2024-09-11 16:05:24.715		4	H	Unknown	
SET	2024-09-11 16:05:24.715		5	H	Unknown	
SET	2024-09-11 16:05:24.715		6	H	Unknown	
SET	2024-09-11 16:05:24.714		7	H	Unknown	
SET	2024-09-11 16:05:24.714		8	H	Unknown	
SET	2024-09-11 16:05:24.714		9	H	Unknown	
SET	2024-09-11 16:05:24.714		10	H	Unknown	
SET	2024-09-11 16:05:24.714		11	H	Unknown	

- Alarm Popup View - Alarm발생 시 발생하는 Popup으로 현재 발생된 Alarm 리스트와 상세보기 등을 제공하며 Alarm Reset을 할 수 있음

Alarm Popup

ERROR / ALARM REPORT

Name

Alarm 0

Level

H

Desc

This is Alarm 0

Solution

Do Solution 0

STATE	SET TIME	ALARM CODE	LEVEL	UNIT	NAME	CLEAR
SET	2024-01-30 15:13:32.214	0	H	System	Alarm 0	Clear
SET	2024-01-30 15:16:11.132	1	H	System	Alarm 0	Clear

All Clear

Close

- IEquipmentSetup의 RegisterViews에서 IEventManager를 통해 아래와 같이 추가한다.

```
// Alarm
MainMenu Alarm = new MainMenu(ViewIds.Alarm, (DrawingImage)IconDictionary["AlarmBell"], PermissionLevel.NONE)
.AddSubMenu("History", new AlarmHistoryView())
.AddSubMenu("Description", new AlarmDescView());

manager.RegisterMainMenu(Alarm);
```

2. Alarm Description 관리

- Alarm Description의 설정값 사용, 다국어 사용

2.1 설정값

- Equipment 설정에 Alarm관련 항목

Equipment	Equipment name	Location	Line no	Machine No
	Aurora Test Equipment		1	1

Operate mode	Simulation Mode	Sequence mode
	☒ Normal	☐ Dynan
	☐ Transfer	

Log setting	Button logger name
	Aurora

ETC
Automatically connect Device ☐
Automatically connect service ☐
Alarm config

Alarm config
Description column ☐
Solution column ☐
Unuse column ☐
Use detail ☐
History view limit 100

- Description column - Alarm Description에서 Description column을 사용할지 여부
- Solution column - Alarm Description에서 Solution column을 사용할지 여부
- Unuse column - Alarm Description에서 use/un-use를 선택할 수 있는 column을 사용할지 여부
- Use detail - 알람 상세보기 기능을 사용할지 여부
(알람 상세보기란 사용자가 특정 Alarm관련 내용을 이미지나 PDF로 첨부해 줬다가 알람 발생 시 쉽게 열 수 있는 기능)

- 아래와 같이 각각의 설정값을 변경 시 Column에 Description과 Use 항목이 사라지고 Detail 항목이 생긴다.

Description column	☐
Solution column	☒
Unuse column	☐
Use detail	☒
History view limit	100

Detail은 이미지 등을 추가할 수 있는 Upload버튼과 삭제버튼, 보기버튼으로 구성된다.(Upload해야지 삭제/보기 버튼 활성화 됨)

Alarm code	Unit	Level	Name	Solution	Detail
0	System	H	Alarm 0		  
1	System	H	Alarm 1		
2	System	H	Alarm 0		

해당 이미지나 PDF는 Alarm 발생 시 Popup에서 바로 볼 수 있다.

AlarmPopup

ERROR / ALARM REPORT

Name

Alarm 0

Level

H

Solution

Do Solution 0

Detail

STATE	SET TIME	ALARM CODE	LEVEL	UNIT	NAME	CLEAR
SET	2024-01-30 15:36:43.745	0	H	System	Alarm 0	CLEAR

All Clear

Close

| 2.2 다국어 처리

- Alarm은 Aurora의 기본적인 다국어 사용과 다른 방법으로 다국어를 처리한다.
- 기본적으로 Name, Description, Solution 3개 항목에 대해서만 다국어를 지원한다.
- 일반적으로 정의된 Alarm Description을 Excel파일 등으로 관리하기 때문에 Alarm 다국어도 동일하게 Excel에 Column을 추가하는 방식으로 처리한다.
아래와 같이 Excel파일에 다국어 Column을 추가한다. column header에 "언어코드" (ex. Name_ko)를 추가하기 구성하며, 그냥 Name일 경우는 영어(en)이다.
여러 언어가 필요할 시 column을 늘려서 사용할 수 있다

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Code	Unit	Level	No	Name	Description	Solution	Name_ko	Description_ko	Solution_ko	Name_vi	Description_vi	Solution_vi
2		0 System	H	-1	Alarm 0	This is Alarm 0	Do Solution 0	알람 0	이것은 알람 0이다.	솔루션 0을 수행	보통서 0	đây là báo động 0	làm giải pháp 0
3		1 System	H	-1	Alarm 0	This is Alarm 1	Do Solution 1	알람 1	이것은 알람 1이다.	솔루션 1을 수행	보통서 1	đây là báo động 1	làm giải pháp 1
4		2 System	H	-1	Alarm 0	This is Alarm 2	Do Solution 2	알람 2	이것은 알람 2이다.	솔루션 2을 수행	보통서 2	đây là báo động 2	làm giải pháp 2

The image displays three screenshots of the Aurora Test Equipment software interface, demonstrating different localization settings. Each screenshot shows a top header bar with the application title, status indicators, and a right sidebar with navigation and user controls.

Screenshot 1 (Top): The interface is in English. The title bar reads "Aurora Test Equipment". The sidebar shows "All settings" and "Basic settings". The main content area displays a table with columns: Alarm code, Unit, Level, Name, Desc., Solution, and Detail. The table contains three rows of alarm data.

Alarm code	Unit	Level	Name	Desc.	Solution	Detail
0	System	H	Alarm 0	This is Alarm 0	Do Solution 0	[Icons]
1	System	H	Alarm 1	This is Alarm 1	Do Solution 1	[Icons]
2	System	H	Alarm 0	This is Alarm 2	Do Solution 2	[Icons]

Screenshot 2 (Middle): The interface is in Korean. The title bar reads "Aurora Test Equipment". The sidebar shows "All settings" and "Basic setting". The main content area displays a table with columns: 알람번호 (Alarm No.), 유닛 (Unit), 레벨 (Level), 이름 (Name), 내용 (Content), 해결방안 (Solution), and 상세보기 (Detail View). The table contains three rows of alarm data.

알람번호	유닛	레벨	이름	내용	해결방안	상세보기
0	System	H	알람 0	이것은 알람 0이다	솔루션 0을 수행	[Icons]
1	System	H	알람 1	이것은 알람 1이다	솔루션 1을 수행	[Icons]
2	System	H	알람 2	이것은 알람 2이다	솔루션 2을 수행	[Icons]

Screenshot 3 (Bottom): The interface is in Vietnamese. The title bar reads "Aurora Test Equipment". The sidebar shows "All settings" and "Basic settings". The main content area displays a table with columns: Alarm code, Unit, Level, Name, Desc., Solution, and Detail. The table contains three rows of alarm data.

Alarm code	Unit	Level	Name	Desc.	Solution	Detail
0	System	H	bảo động 0	Đây là bảo động 0	Thực hiện giải pháp 0	[Icons]
1	System	H	bảo động 1	Đây là bảo động 1	Thực hiện giải pháp 1	[Icons]
2	System	H	bảo động 2	Đây là bảo động 2	Thực hiện giải pháp 2	[Icons]

3. Alarm Set / Event

- AlarmService에서 각종 API를 제공하며 SetAlarm(int code)과 OnSetAlarm, OnResetAlarm Event가 주요 내용이다.

| 3.1 AlarmService

- `IAuroraEnvironment`와 `SequenceContext`에서 `AlarmService`에 접근할 수 있다.

```
public interface IAlarmService
{
    bool Import(string path);

    bool Export(string path);

    bool SetAlarm(int code, bool useReport = true);

    bool ResetAlarm(int code);

    bool ClearAllAlarm();

    AlarmDescription GetAlarmDescription(int code);

    bool ReloadAlarmDescriptions();

    Dictionary<int, AlarmDescription> GetAlarmDescriptionList();
}
```

```
void SetAlarmDescriptionList(Dictionary<int, AlarmDescription> alarmDescList);

List<int> GetCurrentAlarmCodes();
}
```

- Import / Export / ReloadAlarmDescription - AlarmDescription View에서와 동일한 기능이다. Excel파일로 Import/Export할 수 있고 Reload할 수 있다.
- SetAlarm - 가장 많이 사용하는 기능으로 Alarm을 발생 시킨다.
 - int code - 발생시킬 알람의 code로 Alarm Description에서 사용하는 Code이다.
 - useReport - 발생시키는 쪽에서 AlarmDescription 객체에 bool flag를 세팅할 수 있고 Event로 받아서 Alarm을 처리하는 쪽에서 확인하여 해당 알람에 대한 조건을 처리할 수 있다.
- ResetAlarm / ClearAllAlarm - AlarmPopup에서와 동일한 기능으로 Alarm을 Reset할 수 있다.
- GetAlarmDescription(int code) - 장비 DLL에서는 기본적으로 code값만 주고 받으며 Description전체가 필요할 시 code로 Get할 수 있는 API이다.
- GetAlarmDescriptionList / SetAlarmDescriptionList - AlarmDescription 전체를 Get/Set할 수 있다.
- GetCurrentAlarmCode - 현재 발생돼 있는 Alarm List를 받을 수 있다.

| 3.2 Alarm Event

- Alarm Event를 받는 방법을 기술한다.
- IEquipmentSetup의 다음 Delegate를 통해 Event를 처리한다.
- 이 때 AlarmDescription을 받아오려면 AlarmService의 GetAlarmDescription(int code)를 사용한다.

```
public void OnSetAlarm(int code, bool isHeavy)
{

}

public void OnResetAlarm(List<int> codeList)
{

}
```

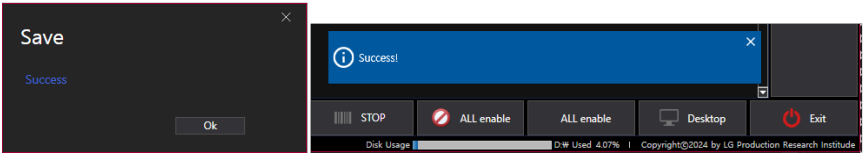
[16] MessageUtil

0. Overview

- MessageUtil의 Pop-up 기능들을 사용하는 방법에 대해 설명한다.

1. MessageUtil API

- MessageUtil에는 UI상의 알림을 위한 API를 모아두었고, 크게 MessageBox와 Toast API가 있다.



* 좌측은 MsgBox 알림 우측은 Toast 알림의 예시

- **MessageBox API**
 - (void) ShowSaveSuccessDialog() : 저장 이 성공 되었음을 알리기 위해 사용
 - (void) ShowSaveFailedDialog() : 저장 이 실패 하였음을 알리기 위해 사용
 - (void) CreateAndShowMessageDialog(title, message) : 좌측 상단의 title을 표시 및 하단에 Message를 표기
 - (bool) ShowYesNoMessageDialog(title, message) : 좌측 상단의 title을 표시 및 하단에 Message를 표기 + 추가적으로 Ok/Cancel 버튼으로 Return값을 반환함

- **Toast API**
 - (void) ShowSuccessToast() : Toast 정보 메세지 표시창에서 Success 메세지를 표기
 - (void) ShowSuccessToast(message, Log = null) : Toast 정상 메세지(정록색) 표시창에 Message를 표시
 - (void) ShowSavedToast() : Tosat 정보 메세지 표시창에서 Saved 메세지를 표기
 - (void) ShowToast(message, Log = null) : Toast 정보 메세지 표시창(파란색)에 Message를 표기
 - (void) ShowErrorToast(message, Log = null) : Toast 에러 메세지 표시창(빨간색)에 Message를 표기
 - (void) ShowWarningToast(message, Log = null) : Toast 메세지 표시창(주황색)에 Message를 표기

→ Logger를 파라미터를 받는 Toast API에 넣어 주면 Toast가 실행될 때마다, Log를 발생시킴(Success : Information로그, Error : Error로그, Warning: Warning 로그를 발생 시킴)

CF) Toast Message같은 경우 UI에서 중복해서 쌓이고, 시간이 지나면 사라지게 됨. 만약 Toast Message를 바로 사라지게 하고 싶은 경우 변동부에서 SYS_REQ_TOAST_MESSAGE_CLEAR를 Publish 하면 됨

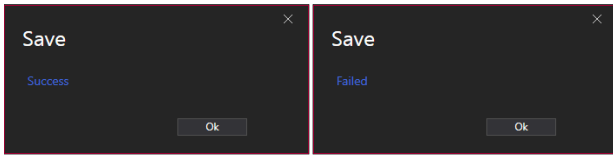
2. 사용 예제

- 사용

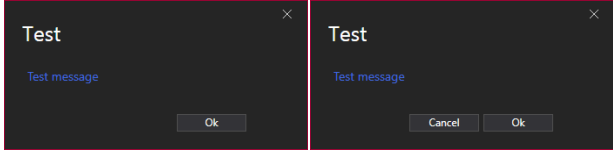
```
//MessageUtil.(MessageUtil API)
MessageUtil.ShowSaveSuccessDialog();
```

• UI상 출력

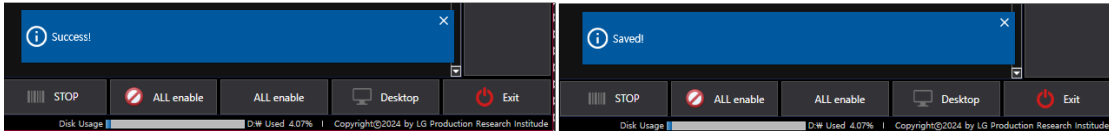
- ShowSaveSuccessDialog / ShowSaveFailedDialog의 UI상 표시



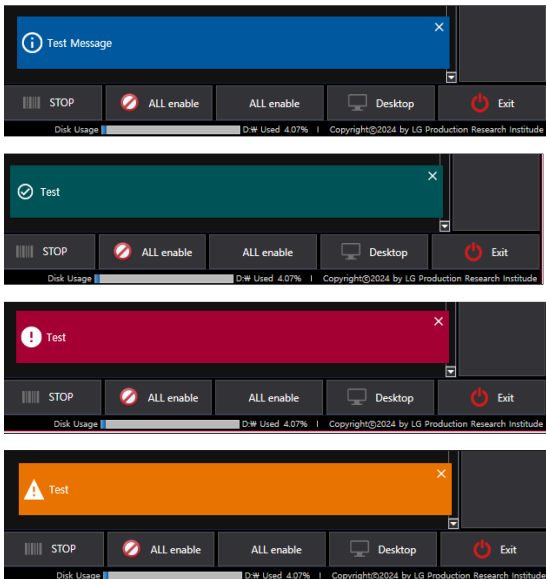
- CreatAndShowMessageDilog(title, message) / ShowYesNoMessageDialog(tittle, message)의 UI상 표시



- ShowSuccessToast / ShowSavedToast의 UI상 표시



- showToast(message,Log =null) / ShowSuccessToast(message, Log = null) / ShowErrorToast(message, Log = null)/ShowWarningToast(message, Log = null)



CF) ToastAPI에 파라메타로 Logger를 넣기

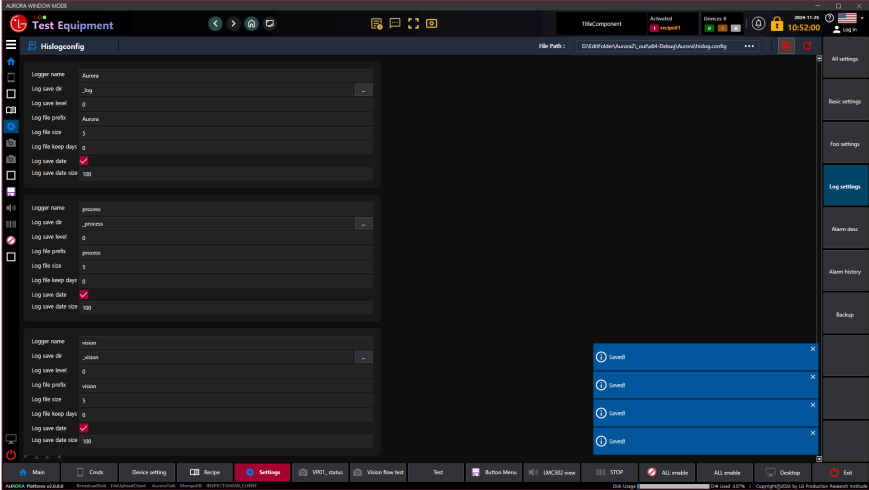
- 아래와 같은 코드를 작성하고 Toast를 발생시키면

```
Logger log = new Logger("LogSettingView");
MessageUtil.ShowToast("Test",log);
```

- Log View에서 Toast에서 발생시킨 로그를 확인 가능

```
2024-11-25 13:53:44.588 D Button      : Save Clicked!
2024-11-25 13:53:44.588 I LogSettingView: Test
2024-11-25 13:53:44.758 D Button      : Save Clicked!
2024-11-25 13:53:44.758 I LogSettingView: Test
```

CF) Toast Message는 아래와 같이 중복해서 쌓이고, 시간이 지나면 사라짐



[17] DataBase

0. Overview

- Aurora에서 기본 제공하는 DataBase에 대한 내용입니다.
- DataHub 서비스를 통해 제공되며 MongoDB를 사용합니다.
- 하나의 item은 16MB까지 가능하며, 그 이상의 대용량 데이터의 경우는 grid.fs를 이용하여 처리 가능합니다.
- 아래 예시들은 sync함수위주로 작성하였으나 거의 모든 API가 async함수도 함께 지원한다.

1. Service 등록

- EquipmentSetup의 RegisterServices()에서 아래와 같이 등록

```
public void RegisterServices(IServiceRegistry registry)
{
    registry.Register("DataBase",
        () => new DataHubService<DBSampleItem2>(true),
        DataHubService<DBSampleItem2>.Config.GetDefaultConfig(),
        (svc) => new DataHubServiceView() { DataContext = new DataHubServiceViewModel<DBSampleItem2>(svc, AuroraEnv)});
}
```

public DataHubService(bool useLargeData = false, string collection = null);	DataHubService 생성자
useLargeData	특정 아이템이 16MB가 넘는 경우 true, 아닌경우 false, 기본값 false
collection	Table 이름, 기본값은 데이터 class 이름

2. Class 생성

- 서비스 등록 시 DBSampleItem Class를 사용하였는데 만드는 방법은 다음과 같습니다.

```
internal class DBSampleItem2 : IMongoItemEntity
{
    [Browsable(false)]
    public string _id { get; set; } // _id는 장비에서 사용하지 않으므로 Browsable(false)로 한다.
    public DateTime? TimeStamp { get; set; }

    public string Name { get; set; }
    public double Width { get; set; }
    public double Height { get; set; }
    public int NumOfDefect { get; set; }
    public int RetryCnt { get; set; }
    public string ImgId { get; set; } // Img와 같이 큰 데이터(16MB이상)를 저장할 경우 실제 데이터가 아닌 데이터의 id를 저장할 수 있는 string 타입으로 선언한다.
}
```

- IMongoItemEntity를 상속받은 Class type만 사용이 가능합니다.

```
public interface IMongoItemEntity
{
    [BsonId]
    [BsonRepresentation(/*Could not decode attribute arguments.*/)]
    string _id { get; set; }

    DateTime? TimeStamp { get; set; }
}
```

- _id는 MongoDB에서 id로 사용되며 자동생성되므로 값을 할당하지 않도록 합니다.
- TimeStamp는 DateTime 정보를 의미하며 값을 넣지않아 null이 들어갈 경우 데이터를 삽입한 시간으로 설정됩니다.

3. Service 사용

3-1. DB Write

- ServiceManager를 통해 Service instance를 받습니다.

```
var dbSvc = AuroraEnv.ServiceManager[ServiceIds.DataHub] as DataHubService<DBSampleItem2>;
```

- 일반 테이블 데이터 쓰기 - 16MB이상의 Large 데이터가 없는 경우

```
DBSampleItem2 item = new DBSampleItem2(); // 아이템을 생성하고
dbSvc.AddItem(item); // 아이템을 DB에 쓴다.
```

- Large 데이터 쓰기 - 16MB이상의 Large 데이터가 있는 경우

```
var id = dbSvc.PushLargeData(imgByteArr, "Img20240923_120121.298_NG.png"); // Large 데이터를 먼저 DB에 쓰고 id를 반환받는다.
var item = new DBSampleItem2(); // 아이템을 생성한다.
item.ImgId = id; // 생성한 아이템의 Large 데이터 ID를 보관할 Property에 받아온 id를 쓴다.
dbSvc.AddItem(item); // 아이템을 DB에 쓴다.
```

- Large 데이터 종류별 API

데이터 종류	API(Read/Write)	설명
byte[]	byte[] GetLargeData(string id)	id를 넣으면 byte[]로 리턴한다. 실패 시 return null
	string PushLargeData(byte[] source, string filename = "")	byte[]를 넣으면 id를 리턴한다.
object	T GetLargeObject<T>(string id)	id를 넣으면 T로 반환한다. T를 명시해야만 한다. 실패 시 return null
	string PushLargeObject(object obj, string filename = "")	instance를 넣으면 id를 리턴한다.
file	bool GetLargeFile(string id, Stream stream)	id와 Stream을 넣으면 Stream에 데이터를 넣어준다. 실패 시 return false
	string PushLargeFile(string filename)	file경로를 넣으면 id를 리턴한다.

(Push의 공통 parameter인 filename은 각각의 데이터의 이름을 의미하며 임의의 값을 써도 무방함)

3-2. DB Read

- 읽기 방법에 따른 API

방법	API	설명
마지막 N개	IEnumerable<T> GetData(int? limit)	마지막 N개의 아이템을 시간 역순으로 읽는다. limit이 null이면 DB 전체를 읽으므로 주의.
특정 column의 값	IEnumerable<T> GetByCol(string col, object val)	특정 column(class의 Property Name)의 값이 val인 아이템을 시간역순으로 읽는다. 없는경우 null 반환.
특정 column의 범위	IEnumerable<T> GetDataByRange(string col, object gteVal, object ltVal)	특정 column의 범위가 gteVal보다 크거나 같고, ltVal보다 작은 아이템을 시간역순으로 읽는다. 없는경우 null 반환.

- Example

```
Data = await dbSvc.GetDataByRangeAsync(nameof(IMongoItemEntity.TimeStamp), StartTime.Value, EndTime.Value.AddMilliseconds(1));
```

(TimeStamp의 값을 범위로 읽는다.)

- Large 데이터가 있는 Table 읽기

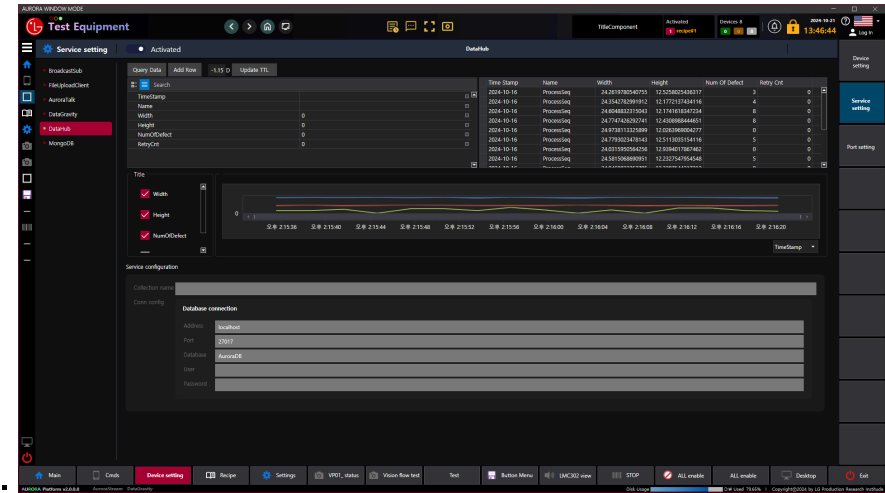
```
var item = dbSvc.GetData(1).First(); // 아이템을 읽는다. 예시로 마지막 1개의 아이템을 읽어옴
byte[] img = dbSvc.GetLargeData(item.ImgId); // 아이템에서 저장된 Large 데이터의 id를 이용하여 Large 데이터를 읽는다.
```

3-3. TTL(Time To Live) 사용

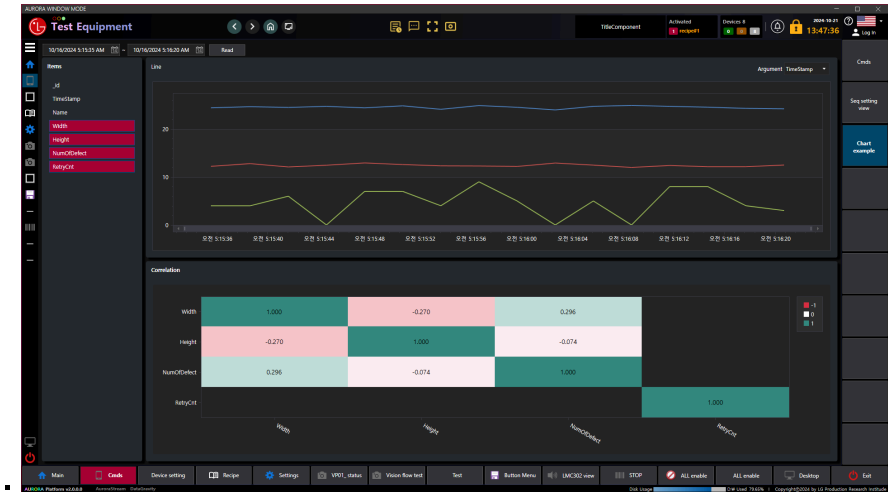
- 아이템이 Add된 시점으로부터 해당 시간을 경과할 경우 자동 삭제됨
- 설정값은 DB에 저장되어 있으므로 App에서는 DB의 값을 읽어서 UI에 표시하고 수정 시 DB의 값을 업데이트 해야 함
- 값은 double타입이며 단위는 Day - ex> 0.01day는 864초. 일반적으로 1개월 이상으로 설정함
- API
 - double GetTTL() - TTL값을 읽는다.
 - bool UpdateTTL(double day) - TTL값을 쓴다.

4. View

4-1. Default View



4-2 Custom View



5. 전체 API

```
public string AddItem(T item);
public Task<string> AddItemAsync(T item);
public IList<string> AddMany(IEnumerable<T> items);
public Task<IList<string>> AddManyAsync(IEnumerable<T> items);
public bool DeleteItem(string id);
public Task<bool> DeleteItemAsync(string id);
public IEnumerable<T> GetByCol(string col, object val);
public Task<IEnumerable<T>> GetByColAsync(string col, object val);
public IEnumerable<T> GetData(int? limit);
public Task<IEnumerable<T>> GetDataAsync(int? limit);
public IEnumerable<T> GetDataByRange(string col, object gteVal, object ltVal);
public Task<IEnumerable<T>> GetDataByRangeAsync(string col, object gteVal, object ltVal);
public byte[] GetLargeData(string id);
public Task<byte[]> GetLargeDataAsync(string id);
public bool GetLargeFile<TT>(string id, Stream stream);
public Task<bool> GetLargeFileAsync<TT>(string id, Stream stream);
public TT GetLargeObject<TT>(string id);
public Task<TT> GetLargeObjectAsync<TT>(string id);
public double GetTTL();
public string PushLargeData(byte[] source, string filename = "");
public Task<string> PushLargeDataAsync(byte[] source, string filename = "");
public string PushLargeFile(string filename);
public Task<string> PushLargeFileAsync(string filename);
public string PushLargeObject(object obj, string filename = "");
public Task<string> PushLargeObjectAsync(object obj, string filename = "");
public bool ReplaceItem(T entity);
public Task<bool> ReplaceItemAsync(T entity);
public bool UpdateField<TT>(string id, string col, TT val);
public Task<bool> UpdateFieldAsync<TT>(string id, string col, TT val);
public bool UpdateTTL(double day);
```

[17][01] MongoDB/Compass 설치 및 사용법

0. Overview

- MongoDB 다운로드 및 설치 가이드입니다.
- MongoDB Browser인 MongoDBCompass 설치 가이드입니다.
- MongoDBCompass로 기본 데이터 조회 및 Command 사용법을 설명합니다.

1. MongoDB 설치

- 설치경로 - <https://www.mongodb.com/try/download/community-kubernetes-operator>
 - 7.0.14버전 권장

Version

7.0.14

▼

Platform

Windows x64

▼

Package

msi

▼

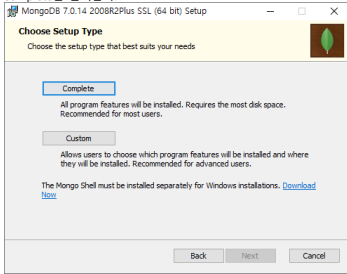
Download

Copy link

More Options

- 설치순서

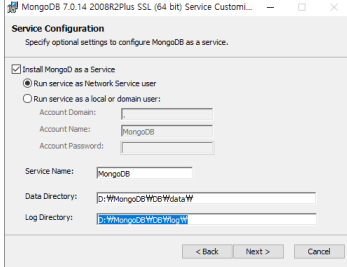
- Complete를 선택한다.



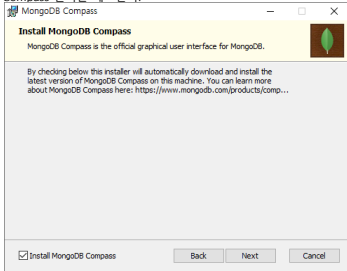
- 아래와 같이 설정한다.(권장사항)

Data Directory : D:\MongoDB\WDB\data\W

Log Directory : D:\MongoDB\WDB\log\W



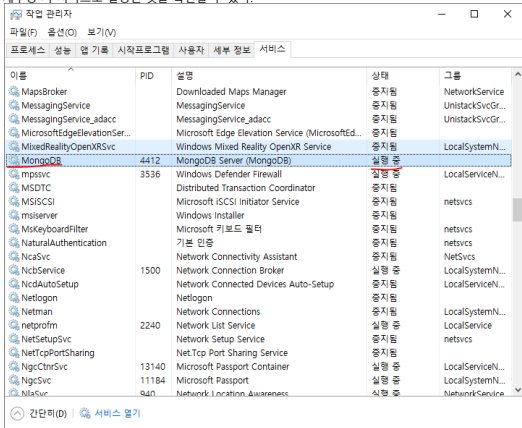
- Compass 설치를 체크한다.



언제크 시 다음 경로를 통해 다운로드 할 수 있다.

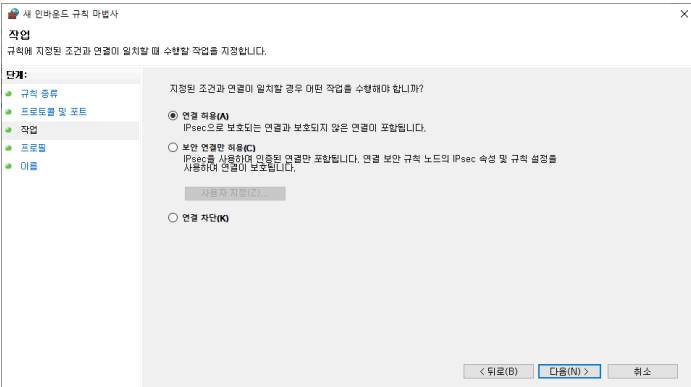
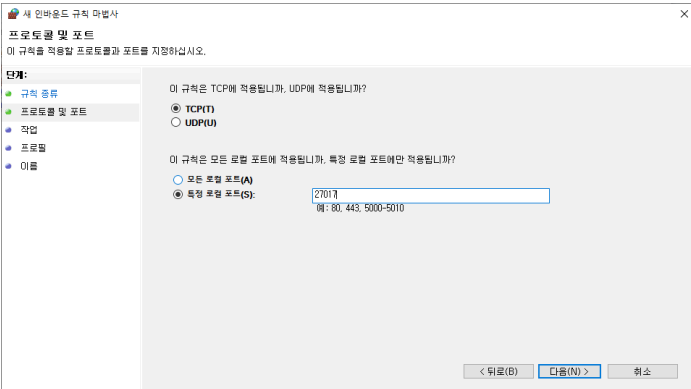
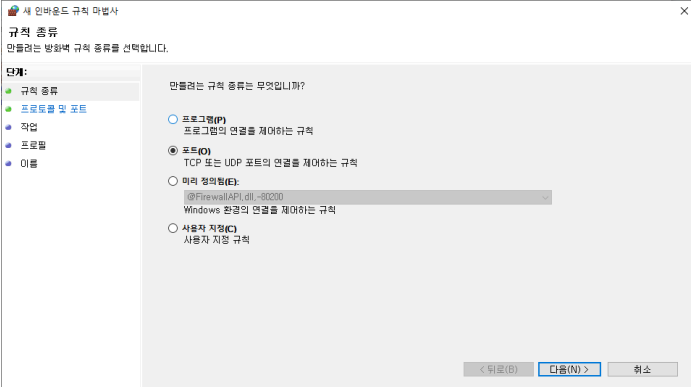
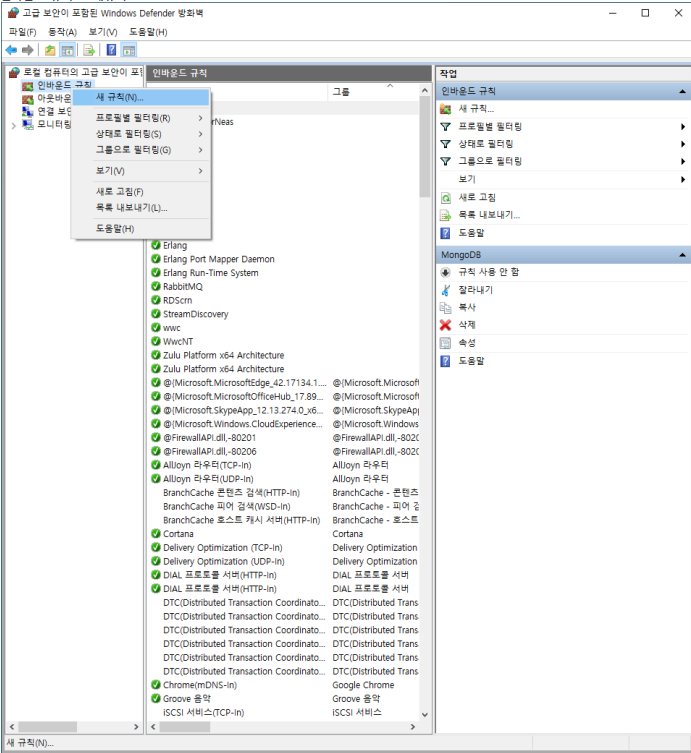
(<https://www.mongodb.com/try/download/compass>)

- 재부팅 시 서비스로 실행된 것을 확인할 수 있다.



- 네트워크로 DB 접속 시 방화벽 설정
 - localhost로 접속시는 불필요하나 외부IP에서 접속 시에는 방화벽 인바운드 허용을 해 줘야 한다.
 - 고급 보안이 포함된 Windows Defender 방화벽

■ 인바운드 규칙 → 새규칙



2. MongoDBComapss 사용방법

- MongoDBCompass는 MongoDB에 접속하여 기본 데이터를 조회할 수 있는 Browser이다.
- 대표적인 기능은 다음과 같다.
 - 데이터 탐색 및 쿼리 작성
 - 데이터베이스 및 컬렉션 관리
 - 성능 모니터링
 - 연결관리

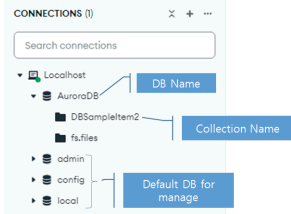
- Connect
 - Add new connection 클릭

- 접속정보 입력
local DB의 경우 localhost, 원격 DB의 경우 localhost 자리에 IP주소를 입력한다.
Port는 기본적으로 27017이다.

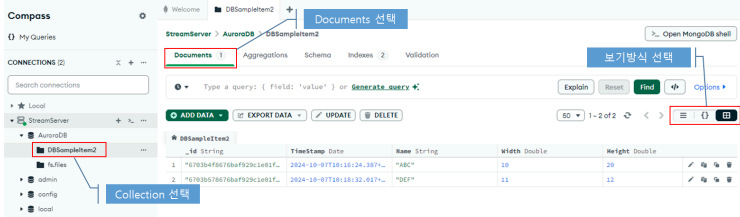
The screenshot shows the 'New Connection' dialog in MongoDB Compass. The 'URI' field contains 'mongodb://localhost:27017/'. The 'Name' field is 'localhost' and the 'Color' is 'No Color'. There are two informational boxes on the right: 'How do I find my connection string in Atlas?' and 'How do I format my connection string?'. At the bottom, there are 'Cancel', 'Save', and 'Connect' buttons.

- 접속 시 DB와 Collection의 구조
 - AuroraDB - 데이터 저장을 위해 만든 DB공간
 - DBSampleItem2 - 데이터가 저장되는 Collection. 일반적으로 Table에 해당됨

- admin, config, local - mongoDB를 관리하기 위해 기본적으로 만들어지는 DB공간



- 데이터 조회
 - Collection → Documents 선택 시 Collection의 Document를 볼 수 있다. 보기방식은 3가지를 제공하며 3번째가 Table 형식으로 보기이다.



3. Replica 설정

MongoDB는 데이터 삽입, 수정, 삭제에 대한 이벤트를 제공하며 Change Stream이라고 부른다.

Change Stream을 사용하기 위해서는 replication 설정을 해야 한다.

- mongod.cfg 파일 수정
 - 위치 : C:\Program Files\MongoDB\Server\7.0\bin\mongod.cfg
 - 아래 붉은 부분을 추가 혹은 수정한다.

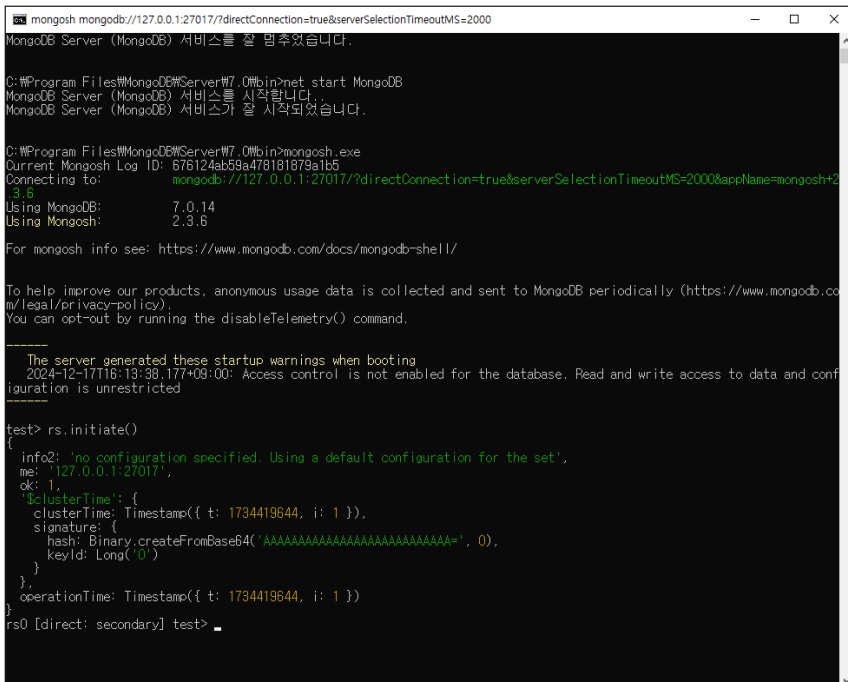
```
storage:
  dbPath: D:\MongoDB\DB\data

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: D:\MongoDB\DB\log\mongod.log

# network interfaces
net:
  port: 27017
  bindIp: 0.0.0.0

replication:
  replSetName: rs0
```

- rs.initiate() 를 수행
 - bin.zip 파일을 받아서 C:\Program Files\MongoDB\Server\7.0\bin\ net start MongoDB
 - 해당 폴더에서 cmd창을 연다.
 - MongoDB 서비스를 멈춘다.
 - net stop MongoDB
 - MongoDB 서비스를 시작한다.
 - net start MongoDB
 - cmd창에서 mongosh.exe 실행
 - rs.initiate() 실행



0. Overview

- C/C++용 MongoDB 드라이버 및 예제입니다.
- MongoDB는 데이터 기본 단위가 json 이라서 데이터 혹은 아이템을 문서라고 명명합니다.

1. 사용 가이드

- Connection
 - MongoDB에 접속
 - uri_string : mongodb 서비스가 설치된 PC의 주소(local PC의 경우 localhost)
 - db_name : AuroraDB로 설정 시 Aurora와 동일한 DB명으로 사용할 수 있다. 다른 이름을 사용할 경우 해당 이름으로 DB영역이 생성된다.
 - collection_name : DB의 Table 이름. 다른 App에서 동일 이름을 사용할 경우 같은 Table을 사용하는 점 주의.

```
const char* uri_string = "mongodb://localhost:27017";
const char* db_name = "AuroraDB";
const char* collection_name = "testcollection";

// MongoDB에 연결
if (!connect_to_mongo(uri_string, db_name, collection_name)) {
    std::cerr << "Failed to connect to MongoDB" << std::endl;
    return 1;
}
```

- TTL (Time To Live) - DB에 문서를 Write할 경우 해당 시간이 지나면 자동 삭제됨.
 - 단위는 sec

```
// TTL 읽고 쓰기
int idx = get_ttl_index();
set_ttl_index(30 * 24 * 60 * 60); // 30일
```

- 문서 삽입
 - bson_t 타입을 사용
 - Person이라는 구조체가 있을 경우 fields의 타입에 따라 bson_t를 만들어서 삽입한다.

```
// 문서 삽입
Person p;
p.name = "dkdk";
p.age = 55;
p.city = "ccc";
p.height = 170;
p.time = time(NULL);

bson_t docPerson;
bson_init(&docPerson);
BSON_APPEND_TIME_T(&docPerson, "time", p.time);
BSON_APPEND_UTF8(&docPerson, "name", p.name.c_str());
BSON_APPEND_INT32(&docPerson, "age", p.age);
BSON_APPEND_UTF8(&docPerson, "city", p.city.c_str());
BSON_APPEND_DOUBLE(&docPerson, "height", p.height);

if (!insert_document(&docPerson)) {
    std::cerr << "Failed to insert document" << std::endl;
}
else {
    std::cout << "Document inserted successfully" << std::endl;
}
```

- 배열 삽입

docPerson에 배열 삽입 시 배열을 먼저 bson_t 문서로 만들어서 해당 문서를 docPerson에 삽입한다.

```
bson_t array;
bson_init(&array);
for (int i = 0; i < 10; i++)
{
    string idx = to_string(i);

    BSON_APPEND_DOUBLE(&array, idx.c_str(), i + 0.3);
}

BSON_APPEND_ARRAY(&docPerson, "data", &array);
```

- 결과

```
{
  "_id": {
    "$oid": "6715cc382bb8e41d960e99a1"
  },
  "time": {
    "$date": "2024-10-21T03:36:24.000Z"
  },
  "name": "abc",
  "age": 55,
  "city": "ccc",
  "height": 170,
  "data": [
    0.3,
    1.3,
    2.3,
    3.3,
    4.3,
    5.3,
    6.3,
    7.3,
    8.3,
    9.3
  ]
}
```
- 문서 조회
 - json filter로 특정 문서를 조회할 수 있다.
 - 내부에서 생성한 동적 메모리를 정리하는 것에 주의.
 - bson_t를 조회하는 부분은 bson_to_person 참고
 - 배열 조회 시 bson_t를 한번 더 꺼내는 점 주의.

```
// 문서 조회
const char* filter_json = R"({"name": "abc"})";
vector<bson_t*> documents;
documents = find_documents(filter_json, documents);

vector<Person> people;
for (auto doc : documents) {
    Person p;
    bson_to_person(doc, p);
}
```

```

people.push_back(p);

// 메모리 정리
delete(doc);
}

for (const auto& p : people)
{
    char timeStr[26];
    ctime_s(timeStr, sizeof(timeStr), &p.time);
    std::cout << "Time" << timeStr << endl
                << "Name" << p.name << endl
                << "Age" << p.age << endl
                << "Height" << p.height << endl
                << "City" << p.city << endl;
}

```

```

bool bson_to_person(const bson_t* bson, Person& person) {
    if (bson == NULL) {
        // err log: bson is NULL
        return false;
    }

    bson_iter_t iter;
    bool init = bson_iter_init(&iter, bson);
    if (init == false) {
        return false;
    }

    while (bson_iter_next(&iter)) {
        const char* key = bson_iter_key(&iter);
        if (strcmp(key, "time") == 0 && BSON_ITER_HOLDS_DATE_TIME(&iter)) {
            person.time = bson_iter_time_t(&iter);
        }
        if (strcmp(key, "name") == 0 && BSON_ITER_HOLDS_UTF8(&iter)) {
            person.name = bson_iter_utf8(&iter, NULL);
        }
        else if (strcmp(key, "city") == 0 && BSON_ITER_HOLDS_UTF8(&iter)) {
            person.city = bson_iter_utf8(&iter, NULL);
        }
        else if (strcmp(key, "age") == 0 && BSON_ITER_HOLDS_INT32(&iter)) {
            person.age = bson_iter_int32(&iter);
        }
        else if (strcmp(key, "height") == 0 && BSON_ITER_HOLDS_DOUBLE(&iter)) {
            person.height = bson_iter_double(&iter);
        }
        else if (strcmp(key, "data") == 0 && BSON_ITER_HOLDS_ARRAY(&iter)) {
            const uint8_t *arr = NULL;
            uint32_t len;
            bson_t array;
            bson_iter_array(&iter, &len, &array);
            bson_init_static(&array, arr, len);

            bson_iter_t array_iter;
            int idx = 0;
            if (bson_iter_init(&array_iter, &array)) {
                while (bson_iter_next(&array_iter)) {
                    person.data[idx++] = bson_iter_double(&array_iter);
                }
            }
            else {
                // err log
            }
        }
    }
    return true;
}

```

- 문서 삭제
 - 조회시 사용하는 방식과 동일하게 json filter를 사용하여 원하는 문서를 삭제할 수 있다.

```

// 문서 삭제
if (!delete_document(filter_json)) {
    std::cerr << "Failed to delete document" << std::endl;
}
else {
    std::cout << "Document deleted successfully" << std::endl;
}

```

- 문서 업데이트
 - 값을 변경하는 기능
 - 조회 시 사용한 json filter로 원하는 문서를 찾고 해당 문서의 특정 필드 값을 변경할 수 있다.

```

// 문서 업데이트
const char* update_json = R"({"set": {"age": 45}})";
if (!update_document(filter_json, update_json)) {
    std::cerr << "Failed to update document" << std::endl;
}
else {
    std::cout << "Document updated successfully" << std::endl;
}

```

- Disconnection
 - 접속해제

```

// MongoDB 연결 해제
disconnect_from_mongo();

```

2. Release

LIB	CDataHub.lib	구현 시 참조 라이브러리
Header	MongoRepo.h	구현 시 Header
DLL	CDataHub.dll bson-1.0.dll mongoc-1.0.dll utf8proc.dll zlibd1.dll	runtime에 사용
zip	DataHubCpp_Release_241106.zip	위 파일 첨부
MongoC	MongoCpp.zip	MongoC Library에서 필요한 DLL 과 Header파일들

EVALUATION LICENSE Are you enjoying Confluence? Please consider purchasing it today.
You have **24 days left** in your **Gliffy Confluence Plugin** free trial

