



Microsoft Cloud Workshop

사물인터넷(IoT)

핸즈온 랩 세부 단계

2018 년 7 월

URL 및 기타 인터넷 웹 사이트 참조를 포함하는 본 문서의 내용은 예고 없이 변경될 수 있습니다. 달리 명시되지 않는 한 본 문서에 예시로 나오는 회사, 기관, 제품, 도메인 이름, 이메일 주소, 로고, 인물, 장소 및 사건은 허구이며 실제 회사, 기관, 제품, 도메인 이름, 이메일 주소, 로고, 인물, 장소 또는 사건과의 연관성을 의도하거나 유추해서는 안 됩니다. 모든 해당 저작권법을 준수하는 것은 사용자의 책임입니다. 저작권에 따른 권리를 제한하지 않는 한 본 문서의 어떤 부분도 복사되거나 검색 시스템에 저장 또는 도입될 수 없으며, Microsoft Corporation 의 서면 허가 없이 어떤 형식이나 수단(전자적, 기계적, 복사, 녹음/녹화 등) 또는 어떤 목적으로든지 전송될 수 없습니다.

Microsoft 는 본 문서의 주제와 관련된 특허, 특허 출원, 상표, 저작권 또는 기타 지적 재산권을 보유할 수 있습니다. Microsoft 의 서면 사용권 계약에 명시적으로 규정된 경우를 제외하고 본 문서가 제공되더라도 해당 특허, 상표, 저작권 또는 기타 지적 재산권에 대한 사용권을 제공하는 것은 아닙니다.

제조업체, 제품 또는 URL 이름은 정보 제공의 목적으로만 제공되며, Microsoft 는 해당 제조업체 또는 모든 Microsoft 기술과 함께 해당 제품의 사용에 대해 명시적, 묵시적 또는 법적 보증이나 보장을 하지 않습니다. 제조업체 또는 제품의 포함이 해당 제조업체 또는 제품에 대한 Microsoft 의 지지를 의미하지는 않습니다. 링크는 타사 사이트에 제공될 수 있습니다. 이러한 사이트는 Microsoft 에서 통제하지 않으며, Microsoft 는 연결된 모든 사이트의 콘텐츠나 연결된 사이트 내의 링크 또는 해당 사이트에 대한 모든 변경 사항이나 업데이트에 대한 책임을 지지 않습니다. Microsoft 는 연결된 모든 사이트에서 수신하는 웹캐스팅 또는 다른 모든 형태의 전송에 대한 책임을 지지 않습니다. Microsoft 는 사용자의 편의를 위해서만 이러한 링크를 제공하며, 해당 링크의 포함이 여기에 포함된 사이트 또는 제품에 대한 Microsoft 의 지지를 의미하지는 않습니다.

© 2017 Microsoft Corporation. All rights reserved.

<https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/General.aspx> 에 나열된 Microsoft 및 상표는 Microsoft 그룹 계열사의 상표입니다. 다른 모든 상표는 해당 소유자의 자산입니다.

목차

사물인터넷 핸즈온 랩 세부 단계	1
요약 및 학습 목표.....	1
개요	1
요구 사항.....	1
핸즈온 랩 실습 전.....	2
작업 1: Azure Pass 활성화.....	2
작업 2: Power BI 프로비저닝	2
작업 3: Blob Storage 프로비저닝	3
실습 1: 환경 설정.....	5
작업 1: SmartMeterSimulator 프로젝트 다운로드 및 열기	5
작업 2: Device Explorer 다운로드 및 설치	5
실습 2: IoT Hub 프로비저닝	6
작업 1: IoT Hub 프로비저닝	6
작업 2: SmartMeterSimulator 구성	8
실습 3: SmartMeterSimulator 구성 완료	11
작업 1: IoT Hub 에 장치 관리 구현.....	11
작업 2: IoT Hub 에서 원격 분석 통신 구현	15
작업 3: 장치 등록 및 원격 분석 확인	18
작업 4: Device Explorer 통한 메시지 수신 확인	21
실습 4: Stream Analytics 를 통한 핫 패스 데이터 처리.....	Error! Bookmark not defined.
작업 1: Power BI 에 대한 핫 패스 처리를 위해 Stream Analytics 작업 생성	Error! Bookmark not defined.
작업 2: Power BI 를 통한 핫 데이터 시각화	Error! Bookmark not defined.
실습 5: Stream Analytics 를 통한 콜드 패스 데이터 처리	Error! Bookmark not defined.
작업 1: 콜드 패스 처리를 위한 Stream Analytics 작업 생성	Error! Bookmark not defined.
작업 2: Blob Storage 에서 CSV 파일 확인	Error! Bookmark not defined.
핸즈온 랩 실습 후.....	Error! Bookmark not defined.
작업 1: 리소스 그룹 삭제	Error! Bookmark not defined.

Appendix 22

 Device Explorer 22

 Microsoft Azure Storage Explorer 23

사물인터넷 핸즈온 랩

세부 단계

요약 및 학습 목표

이 패키지는 스마트 미터에서 방출되어 Azure 에서 분석되는 고속 데이터를 시뮬레이션하는 포괄적 IoT 솔루션을 구현하는 과정을 안내하도록 설계되었습니다. 이 세션에서는 핫 패스에서 실시간 시각화를 위해 원격 분석 데이터의 하위 세트를 필터링하고 콜드 패스에서 장기 보관을 위해 모든 데이터를 저장하는 아키텍처를 설계해 보겠습니다. 패키지 구성이 완료되고 나면 훨씬 손쉽게 IoT Hub 레지스트리에서 장치를 등록하고 Power BI 를 통해 핫 데이터를 시각화할 수 있습니다.

학습 목표:

- 스마트 미터에서 원격 분석 데이터를 전송하는 시뮬레이터 구현
- Stream Analytics 및 Blob Storage 를 통해 데이터를 수집 및 처리
- Power BI 를 통한 핫 데이터 시각화

개요

Fabrikam 은 기업 에너지(전원) 관리를 위한 서비스 및 스마트 미터를 제공합니다. "You-Left-The-Light-On" 서비스는 기업이 에너지 소비량을 파악할 수 있도록 해줍니다.

이 핸즈온 랩 실습에서는 장치 관리, 원격 분석 데이터 수집, 핫/콜드 패스 처리, 보고 등을 포함하여 IoT 시나리오를 위한 포괄적 솔루션을 구현해보겠습니다.

요구 사항

1. Microsoft Azure 구독은 종량제 또는 MSDN 이어야 합니다.
 - a. 평가판 구독은 사용할 수 없습니다.
2. 가상 컴퓨터는 다음으로 구성됩니다.
 - a. Visual Studio Community 2017 이상
 - b. Azure SDK 2.9 이상(Visual Studio 2017 에 포함)

핸즈온 랩 실습 전

소요 시간: 30 분

이 실습에서는 나머지 핸즈온 랩을 사용하기 위한 환경을 설정합니다. 핸즈온 랩에 참석하기 *전에* 환경을 준비하려면 핸즈온 랩 실습 전 섹션에 있는 모든 단계를 따라야 합니다.

작업 1: Azure Pass 활성화

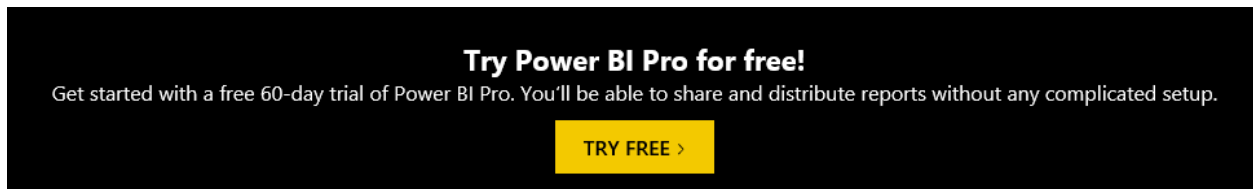
Azure Pass 는 Microsoft Azure 서비스를 1 개월 동안 평가해 보도록 화폐성 차변을 제공하는 특별 행사입니다. 기존에 Azure 계정이 없으신 분은 아래 절차에 따라 Azure Pass 를 활성화 해주시기 바랍니다.

1. live.com 에 회원 가입
2. 해당 계정으로 <https://portal.azure.com> 으로 로그인
3. 그 브라우저에서 추가 탭을 열고, <https://www.microsoftazurepass.com/> 에 접속
4. 이후 마법사를 따라 진행하다가 코드 입력
(코드는 개별 배포)

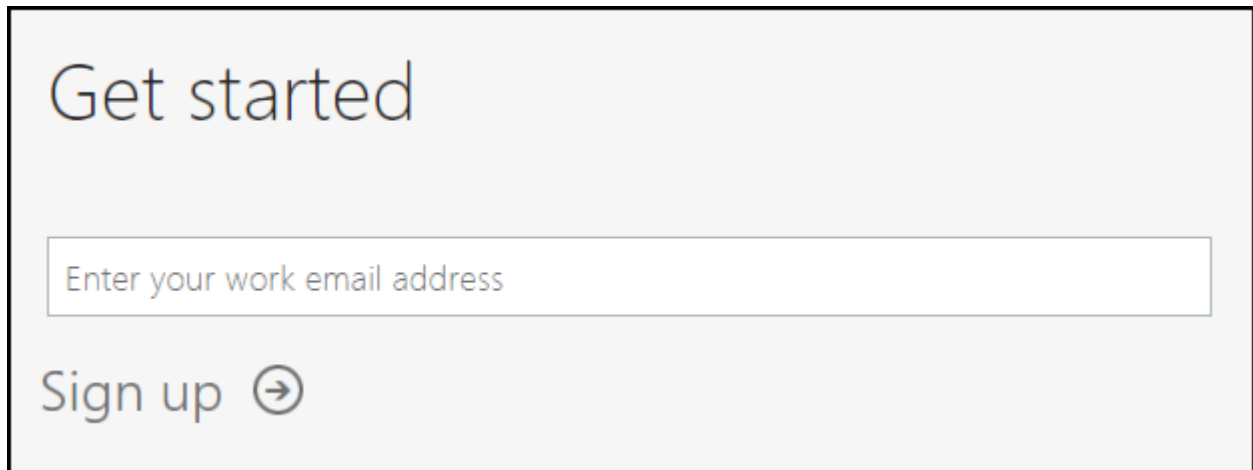
작업 2: Power BI 프로비저닝

Power BI 계정이 없는 경우:

1. <https://powerbi.microsoft.com/features/>로 이동합니다.
2. **Power BI 무료 체험!** 섹션이 보일 때까지 아래로 스크롤한 다음 **무료 체험>** 버튼을 클릭합니다.



3. 해당 페이지에서 업무 이메일 주소(Azure 구독 계정과 동일해야 함)를 입력하고 **등록**을 선택합니다.

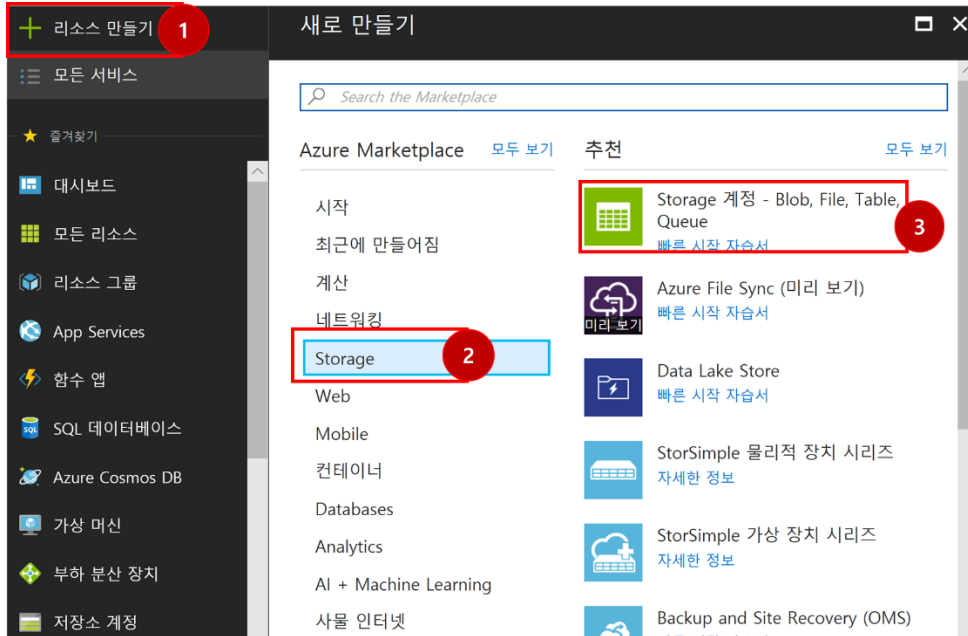


- 화면상의 지시를 따르면 몇 분 내에 Power BI 환경이 준비됩니다. <https://app.powerbi.com/>을 통해 언제나 해당 화면으로 돌아갈 수 있습니다.

작업 3: Blob Storage 프로비저닝

Azure Portal 을 사용하여 새로운 Blob Storage 를 프로비저닝합니다.

- 브라우저를 열고 Azure Portal(<https://portal.azure.com>)로 이동합니다.
- +새로 만들기**를 선택하고 **Storage(저장소)** 와 **스토리지 계정**을 차례로 선택합니다.



- 저장소 계정 만들기에서:
 - 구독: Azure Pass 를 선택합니다
 - 리소스 그룹 : `iot-hol` 을 선택합니다.
 - 이름: `smartmeterblobXX` 로 입력합니다. (Unique 해야함)
 - 위치: 다른 리소스와 동일한 위치를 선택합니다. (한국중부)
 - 성능: 표준으로 설정을 둡니다.
 - 계정 종류 : `StorageV2(범용 v2)`으로 설정합니다.
 - 복제: `LRS(로컬 중복 스토리지)`를 선택합니다.
 - 엑세스 계층(기본값) : `핫`으로 설정을 둡니다.
 - 검초 + 만들기를 클릭하고 리뷰 후 만들기를 클릭합니다.

[기본 사항](#) [고급](#) [태그](#) [검토 + 만들기](#)

Azure Storage는 가용성, 보안, 내구성, 확장성 및 중복성이 뛰어난 클라우드 스토리지를 제공하는 Microsoft 관리 서비스입니다. Azure Storage는 Azure Blob(개체), Azure Data Lake Storage Gen2, Azure Files, Azure 큐 및 Azure 테이블을 포함합니다. 스토리지 계정의 비용은 사용량 및 아래에서 선택한 옵션에 따라 다릅니다. [자세한 정보](#)

프로젝트 정보

배포된 리소스와 비용을 관리할 구독을 선택합니다. 폴더 같은 리소스 그룹을 사용하여 모든 리소스를 정리 및 관리합니다.

* 구독

* 리소스 그룹 [새로 만들기](#)

인스턴스 정보

기본 배포 모델은 최신 Azure 기능을 지원하는 Resource Manager입니다. 대신 클래식 배포 모델을 사용하여 배포하도록 선택할 수 있습니다. [클래식 배포 모델 선택](#)

* 스토리지 계정 이름 ✓

* 위치

성능 ☒ 표준 ☐ 프리미엄


계정 종류

복제

액세스 계층(기본값) ☐ 클 ☒ 핫

4. 서비스에서 Blob 선택

서비스



Blob

구조화되지 않은 데이터를 위한 REST 기반의 개체 저장소

[CORS 규칙 구성](#)

[사용자 지정 도메인 설정](#)

[메트릭 보기](#)

5. +컨테이너 추가

[+ 컨테이너](#) [새로 고침](#) [삭제](#) [액세스 수준 변경](#)

새 컨테이너

* 이름

공용 액세스 수준

[확인](#) [취소](#)

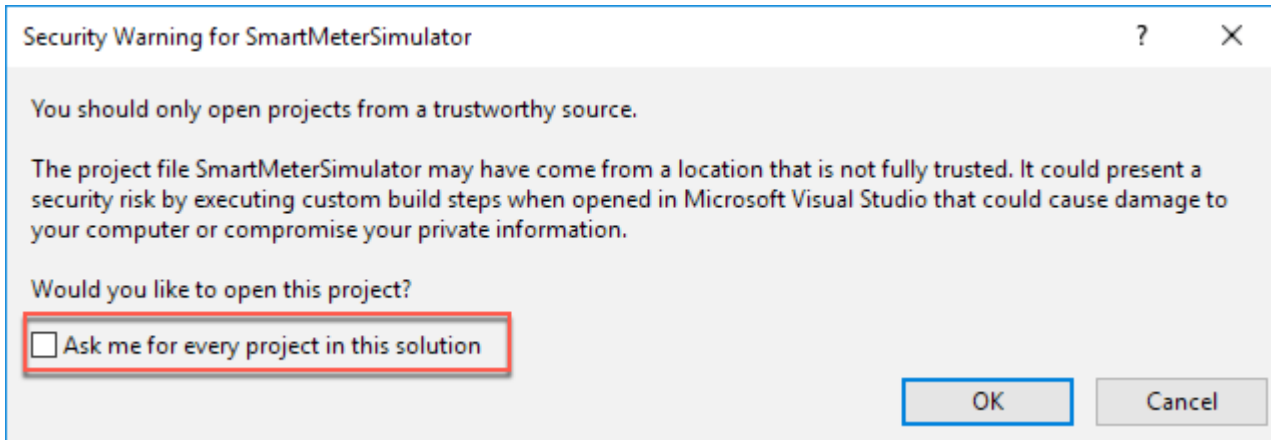
실습 1: 환경 설정

소요 시간: 10 분

Fabrikam 은 장치 등록과 원격 분석 데이터의 생성 및 전송을 시뮬레이션하는 데 사용할 SmartMeterSimulator 를 제공했습니다. Azure 에 스마트 미터를 통합하기 위한 출발점으로 이를 사용하도록 요청했습니다.

작업 1: SmartMeterSimulator 프로젝트 다운로드 및 열기

1. 핸드온 랩 실습 전, 작업 4 에 자세히 설명된 대로 랩 VM 에 연결합니다.
2. 랩 VM 에서 다음 URL 을 통해 SmartMeterSimulator 스타터 솔루션을 다운로드합니다. <https://bit.ly/2wMSwsH> (참고: URL 은 대/소문자 구분)
3. 콘텐츠를 C:\SmartMeter\ 폴더에 압축 해제합니다.
4. Visual Studio 2017 에서 SmartMeterSimulator.sln 을 엽니다.
5. 메시지가 나타나면 Visual Studio 에 로그인하거나 계정을 생성합니다.
6. SmartMeterSimulator 를 위한 보안 경고 창이 나타나면 *이 솔루션의 모든 프로젝트에 대해 물어보기*를 선택 취소한 다음 확인을 선택합니다.



참고: 이 시점에서 솔루션을 구축하려고 할 경우 많은 오류가 발생합니다. 이는 의도적입니다. 다음 실습에서 이러한 빌드 오류를 수정합니다.

작업 2: Device Explorer 다운로드 및 설치

<https://github.com/Azure/azure-iot-sdks/releases>

실습 2: IoT Hub 프로비저닝

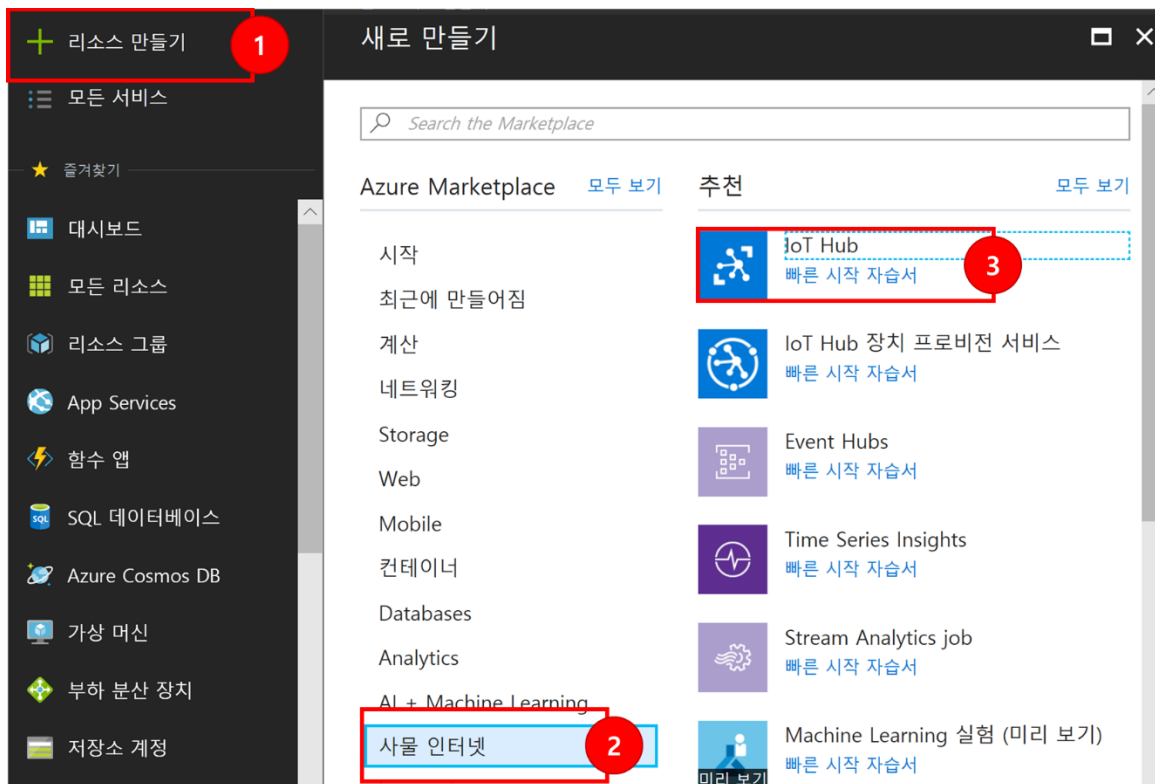
소요 시간: 20 분

Fabrikam 의 아키텍처 설계 세션에서는 SmartMeterSimulator 에서 장치 등록 및 원격 분석 데이터 수집을 모두 관리하기 위해 Azure IoT Hub 를 사용한다는 것에 동의했습니다. 또한 여러분은 Fabrikam 이 IoT Hub 레지스트리에서 장치의 목록 및 상태를 확인하는 데 사용할 수 있는 Microsoft 제공 장치 탐색기 프로젝트를 확인했습니다.

작업 1: IoT Hub 프로비저닝

이 단계에서는 IoT Hub 의 인스턴스를 프로비저닝하겠습니다.

1. 브라우저를 열고 Azure Portal(<https://portal.azure.com>)로 이동합니다.
2. +새로 만들기를 선택한 다음 사물인터넷과 IoT Hub 를 차례로 선택합니다.



3. **IoT Hub** 블레이드에서 다음을 입력합니다.
 - a. Subscription: Azure Pass 를 선택합니다.
 - b. Resource Group: 새로만들기에서 **iot-hol** 리소스 그룹을 입력합니다.
 - c. Region: 한국 중부를 선택합니다.
 - d. IoT Hub Name: smartmeter-hub 와 같이 새 IoT Hub 의 이름을 지정합니다

기본 사항 크기 및 배율 검토 + 만들기

수많은 IoT 자산을 연결, 모니터링 및 관리하도록 도와주는 IoT Hub를 만듭니다. [자세히](#)

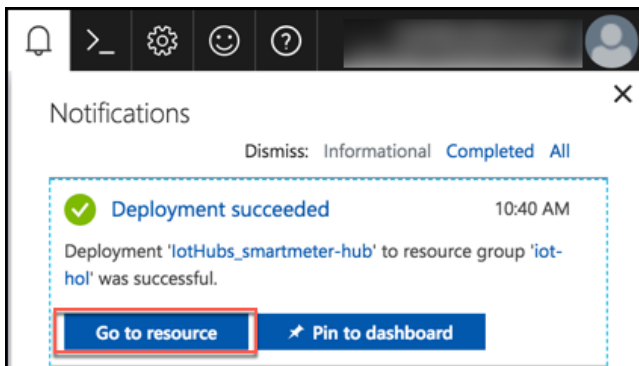
프로젝트 세부 정보

배포된 리소스와 비용을 관리할 구독을 선택합니다. 폴더 같은 리소스 그룹을 사용하여 모든 리소스를 정리 및 관리합니다.

* 구독 ⓘ	Microsoft Azure Internal Consumption
* 리소스 그룹 ⓘ	iot-hol 새로 만들기
* 영역 ⓘ	한국 중부
* IoT Hub 이름 ⓘ	smartmeter-hub ✓

- e. 검토+만들기를 선택합니다.
- f. 리뷰 후 만들기를 선택합니다.

4. IoT Hub 배포가 완료되면 Azure 포털에서 알림 메시지를 수신하게 됩니다. 알림 메시지에서 리소스로 이동을 선택하여 새 IoT Hub 를 탐색합니다.



5. IoT Hub 의 개요 블레이드에서 왼쪽 메뉴의 설정 아래에 있는 공유 액세스 정책을 선택합니다.

설정

- 🔑 공유 액세스 정책
- 🕒 가격 및 크기
- 🏠 IP 필터

6. **iothubowner** 정책을 선택합니다.

POLICY	PERMISSIONS
iothubowner	registry write, service connect, device connect
service	service connect
device	device connect
registryRead	registry read
registryReadWrite	registry write

7. iotHubowner 블레이드에서 **연결 문자열 - 기본 키** 필드 오른쪽의 **복사** 버튼을 선택합니다. 다음 작업에서는 TextBox의 텍스트 속성에 연결 문자열 값을 붙여넣어 보겠습니다.

iotHubowner
smartmeter-hub

저장 취소 기타

액세스 정책 이름
iotHubowner

사용 권한

- ☒ 레지스트리 읽기 ⓘ
- ☒ 레지스트리 쓰기 ⓘ
- ☒ 서비스 연결 ⓘ
- ☒ 디바이스 연결 ⓘ

공유 액세스 키

기본 키 ⓘ

보조 키 ⓘ

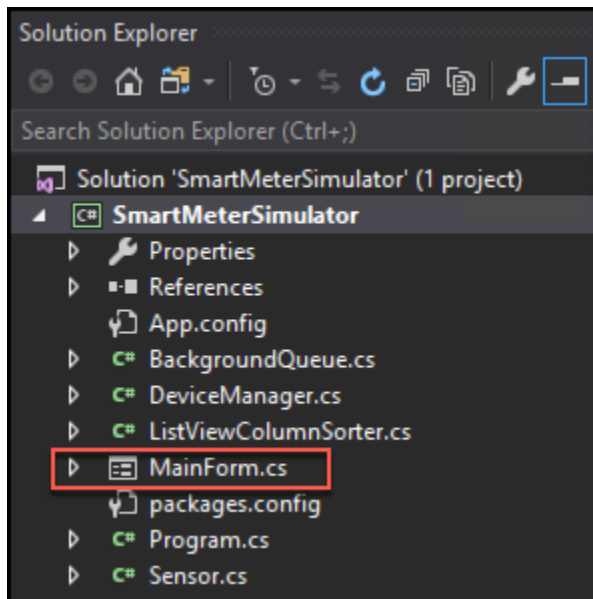
연결 문자열—기본 키 ⓘ

연결 문자열—보조 키 ⓘ

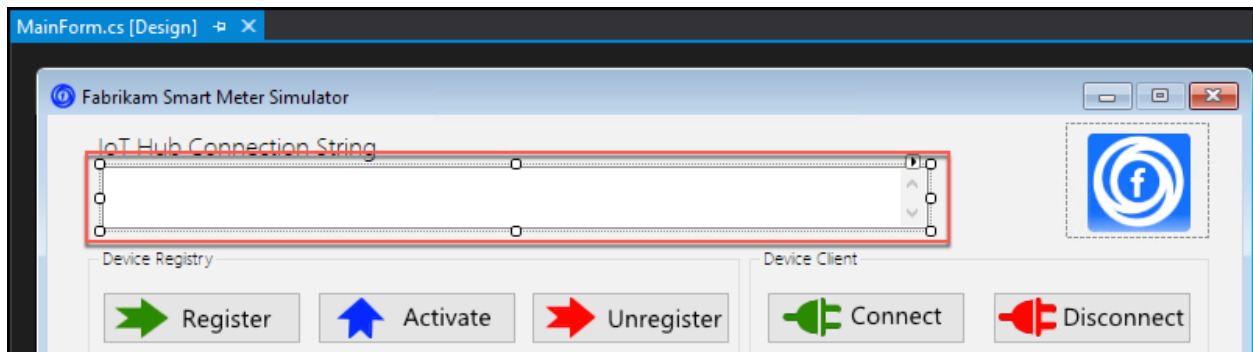
작업 2: SmartMeterSimulator 구성

프로젝트에서 이 연결 문자열을 저장하고 싶은 경우(디버깅을 중지한 경우나 시뮬레이터를 닫은 경우)에는 이를 텍스트 상자를 위한 기본 텍스트로 설정할 수 있습니다. 다음 단계에 따라 연결 문자열을 구성합니다.

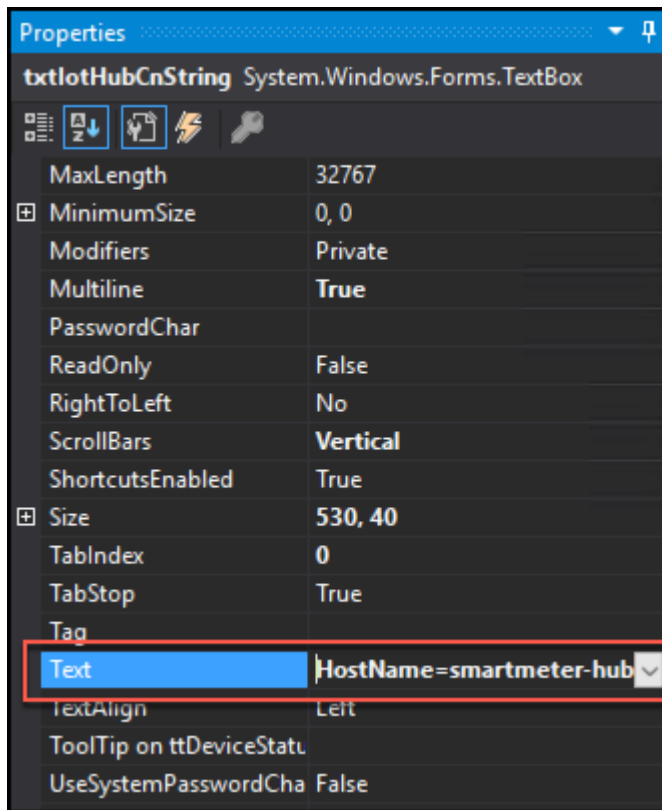
1. Visual Studio 로 돌아갑니다.
2. 솔루션 탐색기 (Solution Explorer)에서 **MainForm.cs** 를 두 번 클릭하여 엽니다. (Visual Studio 인스턴스의 왼쪽 모서리에 솔루션 탐색기가 없는 경우에는 Visual Studio 의 보기 메뉴 아래에서 찾아볼 수 있음)



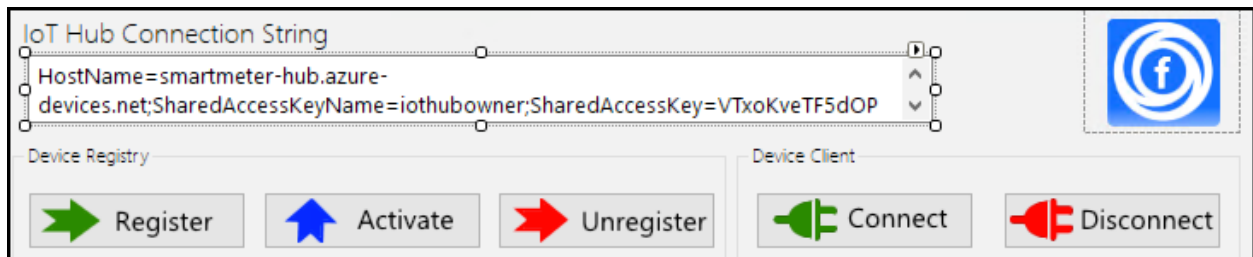
3. Windows Forms 설계 표면에 IoT Hub 연결 문자열 TextBox 를 클릭해서 이를 선택합니다.



4. 속성 (Properties) 패널에서 텍스트 (Text) 속성이 나타날 때까지 스크롤 합니다. 이전 작업의 7 단계에서 복사한 IoT Hub 연결 문자열을 텍스트 속성값에 붙여넣습니다. (솔루션 탐색기 아래에 속성 창이 보이지 않는 경우에는 TextBox 를 마우스 오른쪽 단추로 클릭하여 속성을 선택)



5. SmartMeterSimulator 를 실행할 때마다 연결 문자열이 나타나야 합니다.



실습 3: SmartMeterSimulator 구성 완료

소요 시간: 90 분

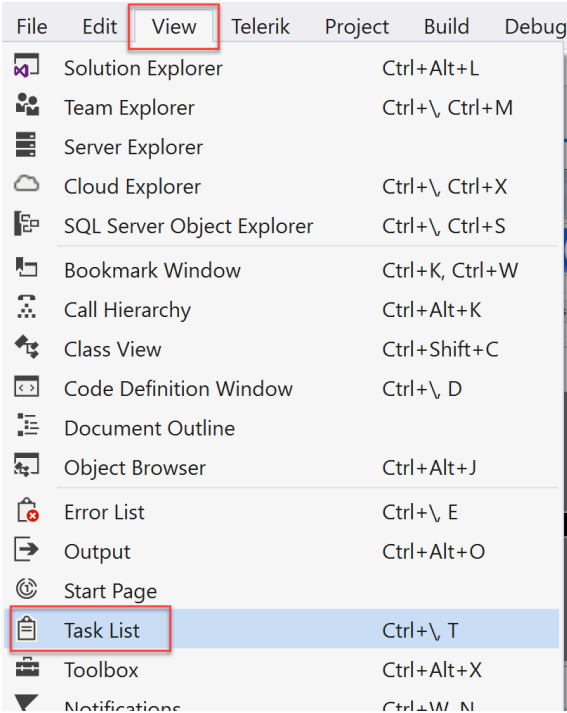
Fabrikam 은 SmartMeterSimulator 솔루션 형태로 부분 완료된 샘플을 남겨뒀습니다. 장치 등록 관리 및 장치 원격 분석 데이터 전송 시 IoT Hub 와 통신하면서 발생한 코드의 누락된 라인을 완료해야 합니다.

* **Nuget Package Install : Microsoft.Azure.Devices / Microsoft.Azure.Devices.Client Install**

(참고: Solution Explorer 의 SmartMeterSimulator -> 오른쪽마우스 클릭 -> Manage Nuget Packages -> Browse)

작업 1: IoT Hub 에 장치 관리 구현

1. Visual Studio 에서 솔루션 탐색기를 사용하여 **DeviceManager.cs** 파일을 엽니다.
2. Visual Studio **보기** 메뉴에서 **작업 목록**을 클릭합니다.



3. 여기에서 모든 TODO 작업에 대한 목록을 확인할 수 있습니다(각 작업은 완료해야 하는 한 코드 라인을 표시).
뒤따라 완료된 코드의 해당 TODO 항목 아래에 **굵은 글씨**로 코드를 복사합니다.
4. 다음 코드는 DeviceManager.cs 에서 완료된 작업을 나타냅니다.

```
class DeviceManager
{
    static string connectionString;
    ///RegistryManager : Device create, remove, update, delete operation
    static RegistryManager registryManager;

    public static string HostName { get; set; }
```

```
public static void IotHubConnect(string cnString)
{
    connectionString = cnString;

    //TODO: 1. connectionString에서 RegistryManager의 인스턴스 생성
    registryManager = RegistryManager.CreateFromConnectionString(connectionString);

    var builder = IotHubConnectionStringBuilder.Create(cnString);

    HostName = builder.HostName;
}

/// <summary>
/// IoT hub에 단일 장치를 등록합니다. 장치는 초기에
/// 비활성 상태로 등록됩니다.
/// </summary>
/// <param name="connectionString"></param>
/// <param name="deviceId"></param>
/// <returns></returns>
public async static Task<string> RegisterDevicesAsync(string connectionString, string
deviceId)
{
    //연결되었는지 확인합니다.
    if (registryManager == null)
        IotHubConnect(connectionString);

    //TODO: 2. 새 장치 생성
    Device device = new Device(deviceId);

    //TODO: 3. 비활성 상태로 장치 초기화
    //후속 단계에서 활성화
    device.Status = DeviceStatus.Disabled;

    try
    {
        //TODO: 4. 새 장치 등록
        device = await registryManager.AddDeviceAsync(device);
    }
    catch (Exception ex)
    {
        if (ex is DeviceAlreadyExistsException ||
            ex.Message.Contains("DeviceAlreadyExists"))
        {
            //TODO: 5. 장치가 이미 존재할 경우 등록된 장치 가져오기
            device = await registryManager.GetDeviceAsync(deviceId);

            //TODO: 6. 추후 활성화가 될 때까지 장치를 비활성화
            device.Status = DeviceStatus.Disabled;
        }
    }
}
```



```

        //TODO: 7. 장치 상태 변경을 통해 IoT Hub 업데이트
        await registryManager.UpdateDeviceAsync(device);
    }
    else
    {
        MessageBox.Show($"An error occurred while registering one or more
devices:\r\n{ex.Message}");
    }
}

//return the device key
return device.Authentication.SymmetricKey.PrimaryKey;
}

/// <summary>
/// 상태를 활성화로 변경하여 이미 등록된 장치를 활성화합니다.
/// </summary>
/// <param name="connectionString"></param>
/// <param name="deviceId"></param>
/// <param name="deviceKey"></param>
/// <returns></returns>
public async static Task<bool> ActivateDeviceAsync(string connectionString, string
deviceId, string deviceKey)
{
    //로컬 설치된 이후에 IoT Hub에 연결되도록
    //프로비저닝된 장치를 활성화하기 위한 서버 측 관리 기능입니다.
    //장치 ID 장치 키가 유효하면 장치를 활성화합니다.

    //연결되었는지 확인합니다.
    if (registryManager == null)
        IotHubConnect(connectionString);

    bool success = false;
    Device device;

    try
    {
        //TODO: 8. 장치 인출
        device = await registryManager.GetDeviceAsync(deviceId);

        //TODO: 9. 장치 키가 일치하는지 확인
        if (deviceKey == device.Authentication.SymmetricKey.PrimaryKey)
        {
            //TODO: 10. 장치 활성화
            device.Status = DeviceStatus.Enabled;

            //TODO: 11. IoT Hub 업데이트

```

```
        await registryManager.UpdateDeviceAsync(device);

        success = true;
    }
}
catch(Exception)
{
    success = false;
}

return success;
}

/// <summary>
/// IoT Hub 레지스트리에서 단일 장치를 비활성화합니다.
/// </summary>
/// <param name="connectionString"></param>
/// <param name="deviceId"></param>
/// <returns></returns>
public async static Task<bool> DeactivateDeviceAsync(string connectionString, string
deviceId)
{
    //연결되었는지 확인합니다.
    if (registryManager == null)
        IotHubConnect(connectionString);

    bool success = false;
    Device device;

    try
    {
        //TODO: 12. 장치 ID로 레지스트리에서 장치 조회
        device = await registryManager.GetDeviceAsync(deviceId);

        //TODO: 13. 장치 비활성화
        device.Status = DeviceStatus.Disabled;

        //TODO: 14. 레지스트리 업데이트
        await registryManager.UpdateDeviceAsync(device);

        success = true;
    }
    catch(Exception)
    {
        success = false;
    }

    return success;
}
```

```

    /// <summary>
    /// IoT Hub 레지스트리에서 단일 장치의 등록을 해제합니다.
    /// </summary>
    /// <param name="connectionString"></param>
    /// <param name="deviceId"></param>
    /// <returns></returns>
    public async static Task UnregisterDevicesAsync(string connectionString, string
deviceId)
    {
        //연결되었는지 확인합니다.
        if (registryManager == null)
            IoTHubConnect(connectionString);

        //TODO: 15. 레지스트리에서 장치 제거
        await registryManager.RemoveDeviceAsync(deviceId);
    }

    /// <summary>
    /// SmartMeterSimulator가 관리하는 모든 장치의 등록을 해제합니다.
    /// </summary>
    /// <param name="connectionString"></param>
    /// <returns></returns>
    public async static Task UnregisterAllDevicesAsync(string connectionString)
    {
        //연결되었는지 확인합니다.
        if (registryManager == null)
            IoTHubConnect(connectionString);

        for(int i = 0; i <= 9; i++)
        {
            string deviceId = "Device" + i.ToString();

            //TODO: 16. 레지스트리에서 장치 제거
            await registryManager.RemoveDeviceAsync(deviceId);
        }
    }
}

```

5. DeviceManager.cs 를 저장합니다.

작업 2: IoT Hub 에서 원격 분석 통신 구현

1. 솔루션 탐색기에서 Sensor.cs 를 열고 IoT Hub 에 원격 분석 데이터를 전송하는 역할을 하는 코드 내에 표시된 TODO 항목들을 완료합니다.
2. 아래 코드는 완료된 결과를 나타냅니다.

```

class Sensor
{
    private DeviceClient _DeviceClient;
    private string _IotHubUri { get; set; }
    public string DeviceId { get; set; }
    public string DeviceKey { get; set; }
    public DeviceState State { get; set; }
    public string StatusWindow { get; set; }
    public double CurrentTemperature
    {
        get
        {
            double avgTemperature = 70;
            Random rand = new Random();

            double currentTemperature = avgTemperature + rand.Next(-6, 6);

            if(currentTemperature <= 68)
                TemperatureIndicator = SensorState.Cold;
            else if(currentTemperature > 68 && currentTemperature < 72)
                TemperatureIndicator = SensorState.Normal;
            else if(currentTemperature >= 72)
                TemperatureIndicator = SensorState.Hot;

            return currentTemperature;
        }
    }

    public SensorState TemperatureIndicator { get; set; }

    public Sensor(string iotHubUri, string deviceId, string deviceKey)
    {
        _IotHubUri = iotHubUri;
        DeviceId = deviceId;
        DeviceKey = deviceKey;
        State = DeviceState.Registered;
    }

    public void InstallDevice(string statusWindow)
    {
        StatusWindow = statusWindow;
        State = DeviceState.Installed;
    }

    /// <summary>
    /// ID와 키로 해당 장치에서 DeviceClient를 인스턴스화하여 IoT Hub에 장치를 연결합니다.
    /// </summary>
    public void ConnectDevice()
    {
        //TODO: 17. DeviceClient의 인스턴스를 생성하여 Iot Hub에 장치를 연결합니다.
    }
}

```

```

        DeviceClient = DeviceClient.Create(_IotHubUri, new
DeviceAuthenticationWithRegistrySymmetricKey(DeviceId, DeviceKey));

        //장치 상태를 준비로 설정합니다.
        State = DeviceState.Ready;
    }

    public void DisconnectDevice()
    {
        //로컬 장치 클라이언트를 삭제합니다.
        _DeviceClient = null;

        //장치 상태를 활성화로 설정합니다.
        State = DeviceState.Activated;
    }

    /// <summary>
    /// 스마트 미터 장치에서 IoT Hub로 메시지를 전송합니다.
    /// </summary>
    public async void SendMessageAsync()
    {
        var telemetryDataPoint = new
        {
            id = DeviceId,
            time = DateTime.UtcNow.ToString("o"),
            temp = CurrentTemperature
        };

        //TODO: 18. telemetryDataPoint를 JSON으로 직렬화
        var messageString = JsonConvert.SerializeObject(telemetryDataPoint);

        //TODO: 19. ASCII에 대한 JSON 문자열을 바이트로 인코딩하고 바이트로 새 메시지 생성
        var message = new Message(Encoding.ASCII.GetBytes(messageString));

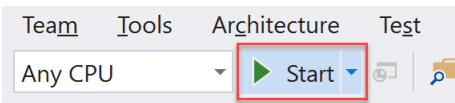
        //TODO: 20. IoT Hub에 메시지 전송
        var sendEventAsync = _DeviceClient?.SendEventAsync(message);
        if (sendEventAsync != null) await sendEventAsync;
    }
}

```

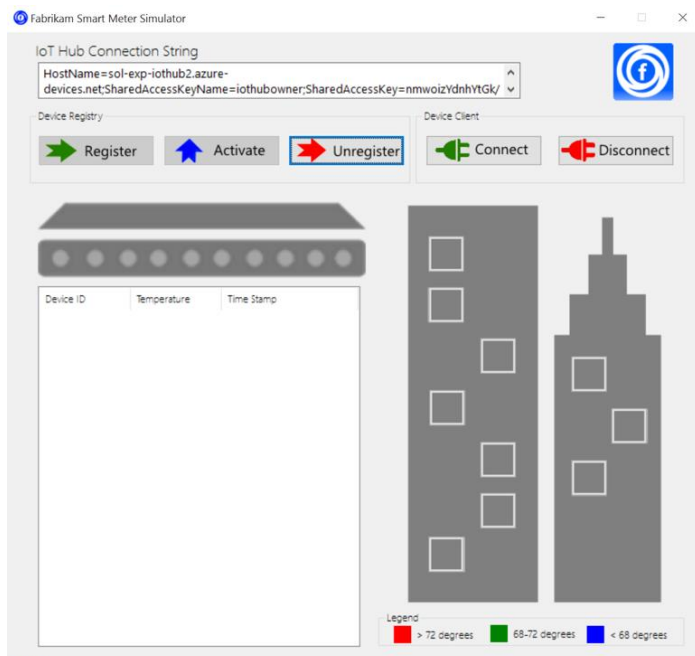
3. Sensor.cs 저장

작업 3: 장치 등록 및 원격 분석 확인

1. Visual Studio 메뉴 모음에서 녹색 시작 버튼을 클릭하여 SmartMeterSimulator 를 실행합니다.



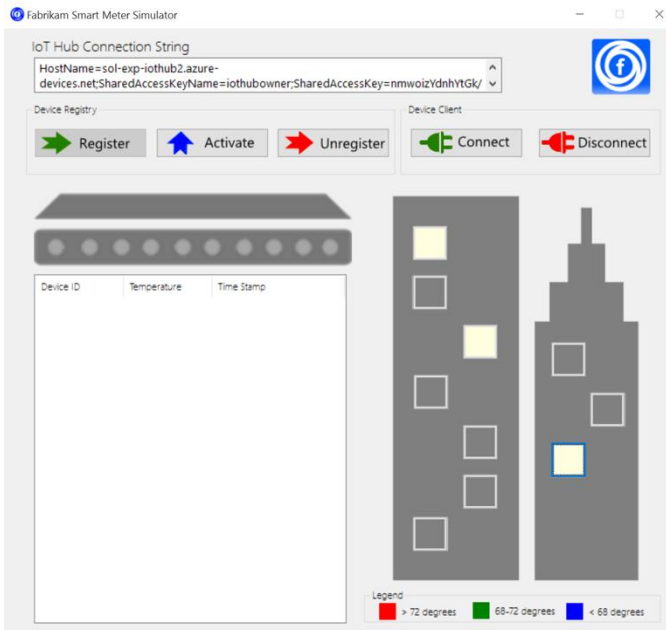
2. 등록 (Register)을 클릭합니다. 빌딩 내의 창이 검은색에서 회색으로 바뀌어야 합니다.



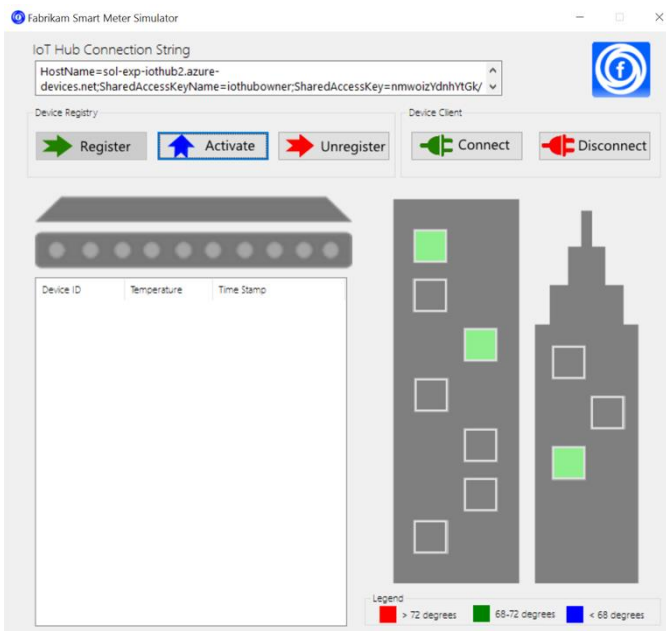
3. Portal 에서 위에서 생성한 IoT Hub 의 IoT 디바이스 메뉴를 선택합니다. 10 개의 디바이스가 Disabled 상태로 추가된 것을 확인할 수 있습니다.

<input type="checkbox"/> 디바이스 ID	상태	마지막 작업	마지막 상태 업데이트...	인증 유형	클라우드-디바이스
Device2	Disabled			Sas	0
Device7	Disabled			Sas	0
Device3	Disabled			Sas	0
Device1	Disabled			Sas	0
Device9	Disabled			Sas	0
Device0	Disabled			Sas	0
Device8	Disabled			Sas	0
Device4	Disabled			Sas	0
Device6	Disabled			Sas	0
Device5	Disabled			Sas	0

4. 몇 개의 창을 클릭합니다. 각각은 장치 설치를 시뮬레이션하려는 장치를 나타냅니다. 선택된 창이 노란색으로 바뀌어야 합니다.



5. **활성화 (Activate)**를 클릭하여 IoT Hub 레지스트리에서의 장치 상태 변경(비활성화에서 활성화로)을 시뮬레이션합니다. 선택된 창이 녹색으로 바뀌어야 합니다.

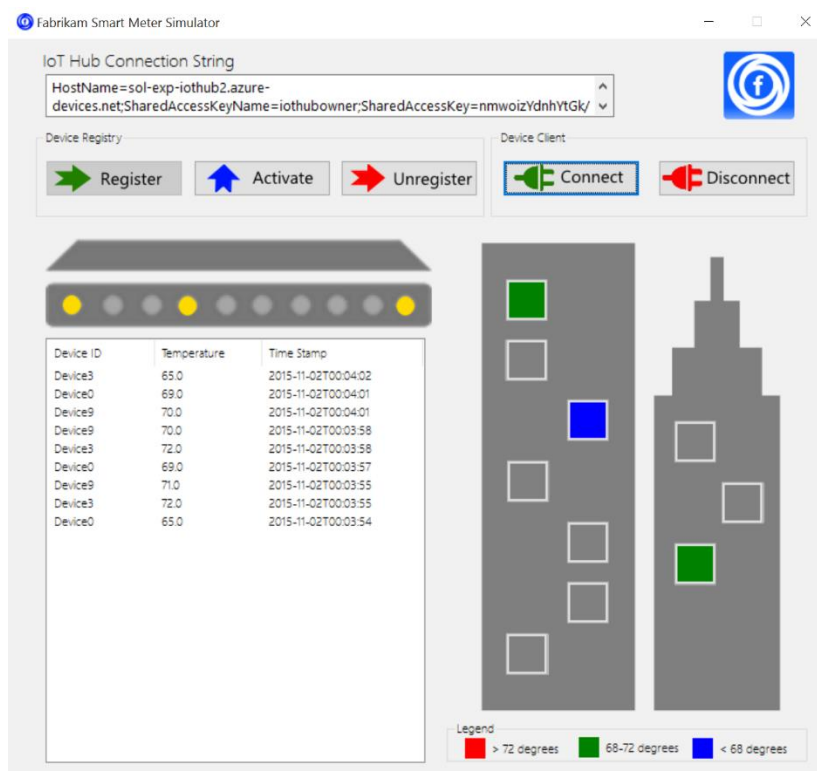


6. 이 시점에서는 10 개의 장치(회색 창)가 등록되었지만 선택한 창(녹색)만 활성화가 되었습니다. 확인을 위하여 **IoT 디바이스** 목록을 새로 고침 합니다.

7. 10 개의 디바이스 목록에서 **활성화(enabled)**상태인 장치가 모두 표시되어야 합니다.

DEVICE ID	STATUS
Device0	enabled
Device1	enabled
Device2	disabled
Device3	enabled
Device4	disabled
Device5	disabled
Device6	disabled
Device7	disabled
Device8	enabled
Device9	enabled

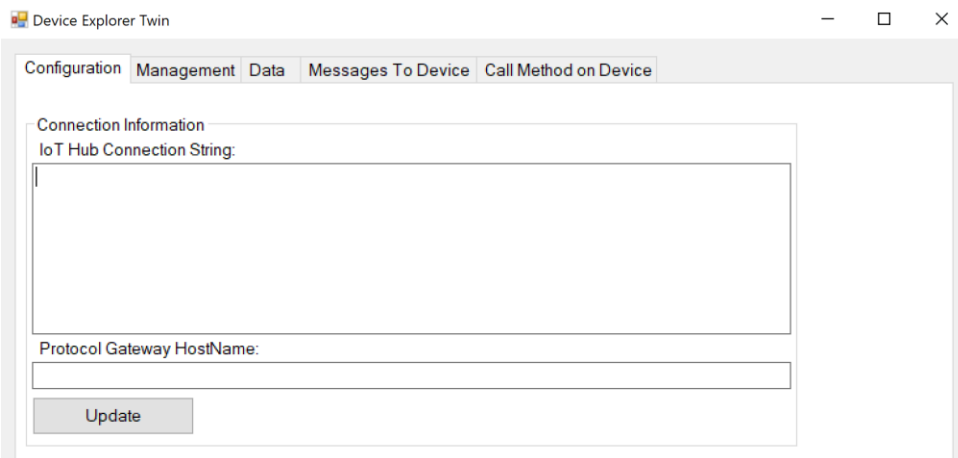
8. **SmartMeterSimulator** 로 돌아갑니다.
9. **Connect** 을 클릭합니다. 창 색깔이 변해서 스마트미터가 원격 분석 데이터를 전송 중임을 나타내면 활동을 확인 할 수 있습니다. 왼쪽의 그리드에는 전송된 각각의 원격 분석 메시지와 시뮬레이션된 온도 값이 나열됩니다.



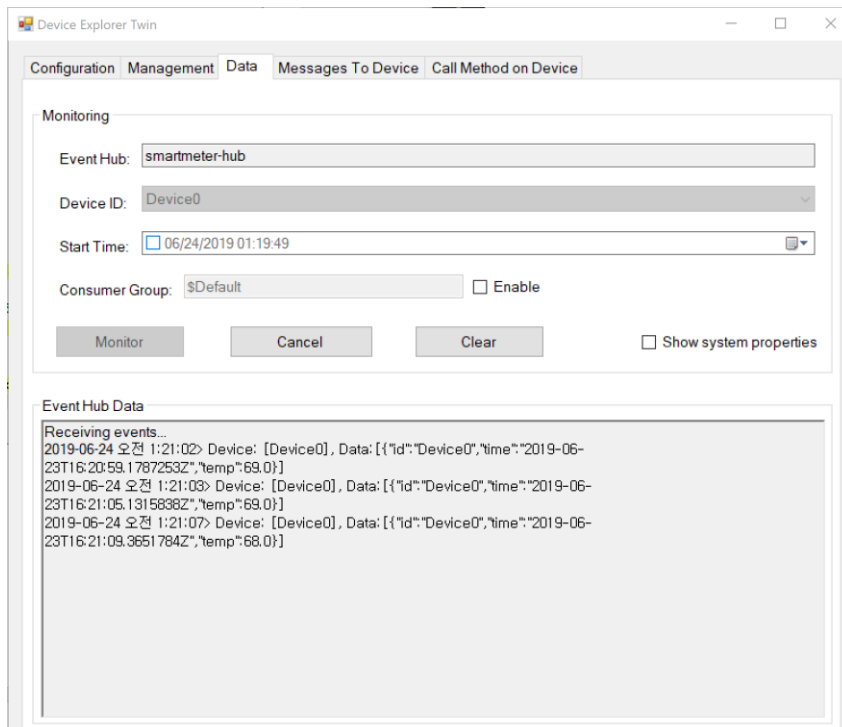
10. 스마트 미터가 계속 실행되도록 허용합니다. (원격 분석 데이터의 전송을 중지하고 싶을 때마다 **Disconnect** 버튼을 클릭).

작업 4: Device Explorer 통한 메세지 수신 확인

1. Device Explorer 를 실행하고 IoT Hub 의 Connection String 을 Update 합니다.



2. Data 탭으로 이동하여 Enabled 되어있는 디바이스 중의 하나를 선택하여 모니터링을 시작한다.



Appendix

Device Explorer

1. <https://github.com/Azure/azure-iot-sdks/releases>
2. Download and run **Device Explorer**. ([https://github.com/Azure/azure-iot-sdks/blob/master/tools/DeviceExplorer/doc/how to use device explorer.md](https://github.com/Azure/azure-iot-sdks/blob/master/tools/DeviceExplorer/doc/how%20to%20use%20device%20explorer.md))
3. After running the installer, the *DeviceExplorer.exe* can be found at **C:\Program Files (x86)\Microsoft\DeviceExplorer**.
4. Run *DeviceExplorer.exe*.
5. Open the *Configuration* tab.
6. Paste the *iothubowner* connection string from the previous section of this lab in the *IoT Hub Connection String* field.
7. Click Update

Device Explorer

Configuration Management Data Messages To Device

Connection Information

IoT Hub Connection String:

HostName=rickgrimes-iot-labs.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=YOURUNIQUEIOTPRIMARYKEY=

Protocol Gateway HostName:

Update

Shared Access Signature

Key Name: iothubowner

Key Value: YOURUNIQUEIOTPRIMARYKEY=

Target: rickgrimes-iot-labs.azure-devices.net

TTL (Days): 365

Generate SAS

Microsoft Azure Storage Explorer

<https://azure.microsoft.com/en-us/features/storage-explorer/>