

Connecting AutomationDirect Do-more! BRX PLC to Microsoft Azure IoT Platform as a Device

This guide describes the steps required to connect the Do-more! BRX PLC to the Microsoft Azure Platform using MQTTs.

Introduction

Industrial IoT has enabled a new surge of manufacturing productivity with Industry 4.0. Do-more! BRX PLCs are stackable programmable logic controllers that integrate traditional industrial automation technologies and IoT technologies to achieve Industry 4.0 compliance.

Prerequisites

Software Requirements

1. Microsoft Azure Cloud account: <https://azure.microsoft.com/en-us/>
2. An instance of the Azure IoT Hub
3. An instance of the Azure Device Provisioning Service (DPS)
4. Microsoft Azure IoT Explorer: <https://github.com/Azure/azure-iot-explorer/releases>
5. Do-more! Designer 2.8 +: <https://support.automationdirect.com/products/domore.html>
- 6.

Hardware Requirements

- Do-more! BRX BX-DM1E-XXX-X (Ethernet Enabled)

Provision your device over DPS

1. Sign in to the Azure portal, select the **All resources** button on the left-hand menu and open your Device Provisioning service.
2. Select the **Manage enrollments** tab, and then select the **Add individual enrollment** button at the top.
3. In the **Add Enrollment** panel, enter the following information, and press the **Save** button.
 - **Mechanism:** Select **Symmetric Key** as the identity attestation *Mechanism*.
 - **Auto-generate keys:** Check this box.
 - **Registration ID:** Enter a registration ID to identify the enrollment. Use only lowercase alphanumeric and dash ('-') characters. For example, **symm-key-brx-01879**.
 - **IoT Hub Device ID:** Enter a device identifier. For example, **brx-01879**.

- Once you have saved your enrollment, the **Primary Key** and **Secondary Key** will be generated and added to the enrollment entry. Your symmetric key device enrollment appears as **symm-key-brx-01879** under the *Registration ID* column in the *Individual Enrollments* tab.

Open the enrollment and copy the value of your generated **Primary Key**.

Registering the BRX PLC in Device Provisioning Service

The Device Provisioning Service uses security tokens to authenticate services to avoid sending keys on the wire. Additionally, security tokens are limited in time validity and scope. Some scenarios do require you to generate and use security tokens directly. Such scenarios include the direct use of the HTTP surface. We will be using HTTP to register our BRX PLC.

Creating a Security token

You use security tokens to grant time-bounded access for services to specific functionality in IoT Device Provisioning Service. To get authorization to connect to the provisioning service, services must send security tokens signed with either a shared access or symmetric key.

A token signed with a shared access key grants access to all the functionality associated with the shared access policy permissions.

The security token has the following format:

SharedAccessSignature sig={signature}&se={expiry}&skn={policyName}&sr={URL-encoded-resourceURI}

VALUE	DESCRIPTION
{signature}	An HMAC-SHA256 signature string of the form: {URL-encoded-resourceURI} + "\n" + expiry. Important: The key is decoded from base64 and used as key to perform the HMAC-SHA256 computation.
{expiry}	UTF8 strings for number of seconds since the epoch 00:00:00 UTC on 1 January 1970.
{URL-encoded-resourceURI}	Lower case URL-encoding of the lower-case resource URI. URI prefix (by segment) of the endpoints that can be accessed with this token, starting with host name of the IoT Device Provisioning Service (no protocol). For example, mydps.azure-devices-provisioning.net.
{policyName}	The name of the shared access policy to which this token refers.

The following Python snippet shows a function called `generate_Sas-Token` that computes the token from the inputs `resourceUri`, `signingKey`, `policyName`, `expiresInMins`. Microsoft Visual Studio Code can run this function with the Python plugin.

Using Python

```
from base64 import b64encode, b64decode
from hashlib import sha256
from time import time
from urllib.parse import quote_plus, urlencode
from hmac import HMAC

def generate_sas_token(uri, key, policy_name, expiry=3600):
    ttl = time() + expiry
    sign_key = "%s\n%d" % ((quote_plus(uri)), int(ttl))
    signature = b64encode(HMAC(b64decode(key), sign_key.encode(), sha256)
        .digest())

    rawtoken = {
        "sr" : uri,
        "sig": signature,
        "se" : str(int(ttl))
    }

    if policy_name is not None:
        rawtoken["skn"] = policy_name

    return "SharedAccessSignature " + urlencode(rawtoken)

uri = "scopeId/registrations/registrationId"
key = "key"
expiry = "Time to Live"
```

```
policy="registration"

print(generate_sas_token(uri, key, policy, expiry))
```

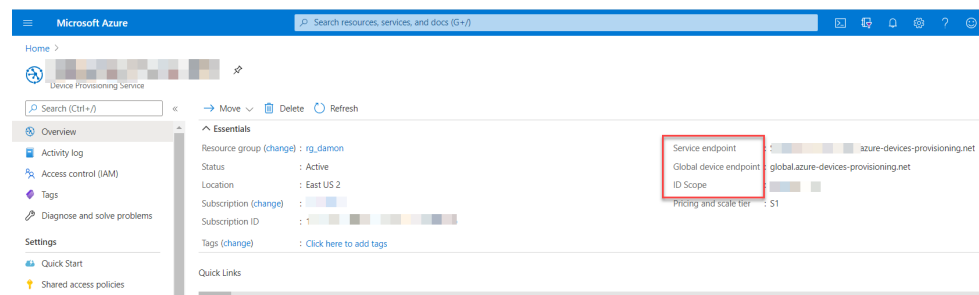
Input Parameters

URI - Global Provisional Service uses the follow format:

scopelD/registrations/registrationId

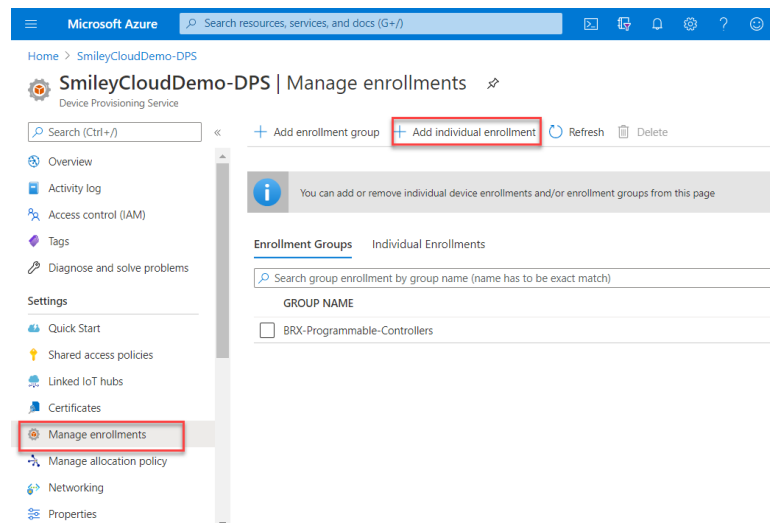
scopelD – ID Scope in the DPS Overview

registrationId – User defined ID



Key – The key depends on whether you are enrolling an individual device or enrolling a device in a group which uses a derived key. This document will focus on individual enrollment. How to create a derived key can be found at [How to provision devices using symmetric key enrollment groups](#).

Individual Enrollment



1. Navigate to the “Manage enrollments” menu from the left nav underneath your DPS instance and click “Add Individual Enrollment”.

- On the 'Add Enrollment' dialog, for Mechanism, choose "Symmetric Key". Select "Auto-generate keys". In "Registration ID" provide a unique ID for the device. In "IoT Hub Device ID" provide a unique device ID. Select "false" for "IoT Edge Device".

Home > SmileyCloudDemo-DPS >

Add Enrollment

Save

Mechanism * ⓘ
Symmetric Key

Auto-generate keys ⓘ
☒

Primary Key ⓘ
Enter your primary key

Secondary Key ⓘ
Enter your secondary key

Registration ID * ⓘ
Individual enrollment registration id

✖ The value must not be empty.
✖ Registration ID must only contain alphanumeric and hyphen.

IoT Hub Device ID ⓘ
Device ID

IoT Edge device ⓘ
☐ True ☒ False

- Click "Save" to create the enrollment.
- Select Individual Enrollments>[Your Registration ID]
- The key is the Primary Key

Enrollment Details

Save Refresh Regenerate keys

Registration Status
Status: unassigned
Assigned hub: -
Device ID: -
Last assigned: -

Authentication Type
Mechanism: Symmetric Key

Primary Key
nrOgGJRtQitu00V9uLGpd9eWeBqyczRnVp52DHoSztUrf0qWnhZ1v+5vOo+4UEs.

Secondary Key

Policy – "registration"

Expire – Unix Epoch time

Preparing your Device

Programming Connection Options

The BRX MPU allows several built-in programming interface options. Choose from one of the following suggested communication methods for this short step-by-step introduction to the BRX platform. You should have at least one of these programming options ready and available.

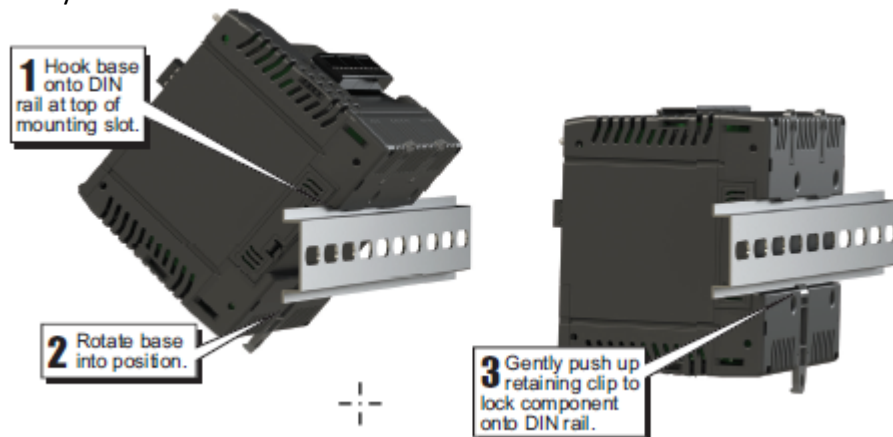
- Ethernet port: A standard Ethernet patch cable Ethernet Cable C5E-STPxx-xx
- Serial port: Using the 3-pin serial connection on the MPU Serial Cable ZL-DB9F-CBL-2P
- POM Slot Programming Options – USB POM and cable assembly

Preparing the Hardware

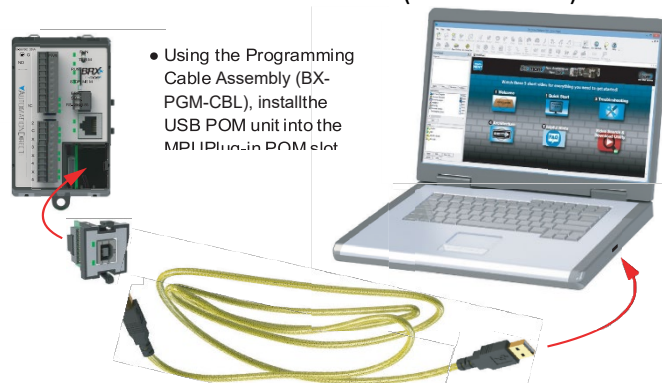
The “Installation and Wiring” chapter of the BRX User Manual (BX-USER-M) contains detailed information for your specific model.

The following summary explains the basic steps needed to get the BRX MPU ready to establish a link so you can program and run a project.

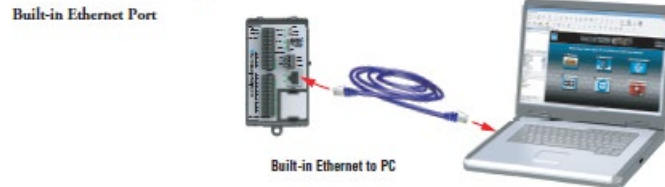
1. Installing the BRX MPU on DIN rail:
 - a. Hook rear upper tabs over flange of DIN rail.
 - b. Rotate the unit down toward the DIN rail, pressing firmly until the BRX MPU snaps securely to the DIN rail.



2. There are three ways to connect to the BRX MPU for programming:
 - a. Using the USB POM with a USB programming cable. For USB connectivity the BX-PGM-CBL assembly is required. This assembly includes one USB POM module (BX-P-USB-B) and one 6ft USB-A to USB-B cable (USB-CBL-AB6).



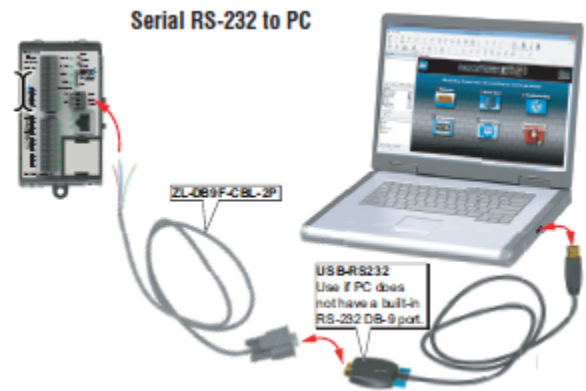
- b. Using Ethernet cable with built-in Ethernet port. There are two options for connecting the BRX via Ethernet to a PC. One option is to use the built-in Ethernet port and connect to the PC via a Cat5 Ethernet cable. An alternate connection is available by using the optional Ethernet POM (BX-P-ECOMEX) module that plugs into the POM slot on the MPU. Both Ethernet options support 10/100Base-T via the RJ45 port. The Ethernet port supports Auto-Crossover, which allows the use of a Cat5 patch or crossover cable.



- c. Using RJ12 Serial POM module and RJ12 programming cable or via the built-in 3-pin serial port connector with user supplied cabling. It is possible to connect to the BRX using the CPU's built-in serial port.

There are two options to consider for connecting the BRX built-in serial port to a PC. The first option uses a ZL-DB9F-CBL-2P to a PC USB port using a USB to Serial adapter (USB-RS232, as shown below). If the PC has a 9-pin serial port connection, you do not need to use the USB to Serial adapter. The second option (not shown) is to use a user-built cable to connect the built-in 3-pin serial port to the PC serial port or USB to Serial adapter.

BRX 3-Pin Serial Port				
ZL-DB9F-CBL-2P			To	3-pin Terminal
Pin #	Term	Color		
2	RXD	Brown	TXD
3	TXD	Red	RXD
5	GND	Yellow	GND
7	RTS	Blue	Jumper	
8	CTS	Violet		



Apply Power to PLC

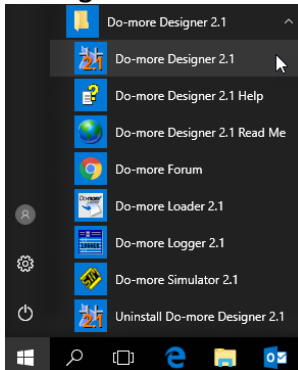
Once all the power wiring has been completed and verified, connect the appropriate voltage source to the power supply and power up the system. The BRX MPU will perform a self-evaluation once power is applied.

The following summary explains the basic steps needed to get the BRX MPU ready to establish a link for programming and running a project using USB.

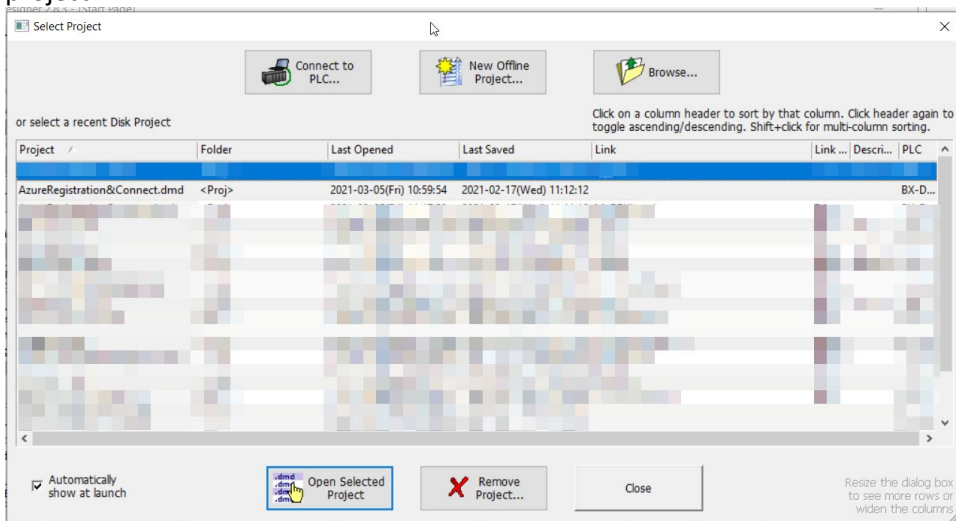
Run Sample

1. Download the PLC Project from GitHub at the following URL: <https://github.com/dpervis/BRX-Azure-Conn>. Save the project to the following directory: C:\Users\Public\Documents\Do-more\DesignerX_X\Projects

2. Launch Do-more! Designer software from your PC Start menu or All Programs menu. If the software link is not embedded in the Start menu, use the path: **Start > All Programs > Do-more > Designer x.x > Do-more! Designer x.x** to launch the software.

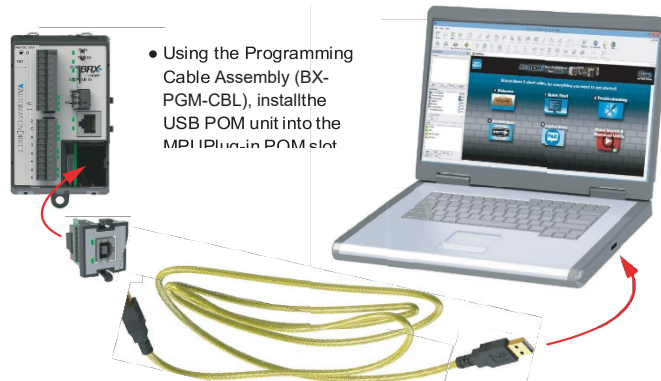


3. Select Browse and Select the project that was downloaded in Step 1. Click “Open” to load project.

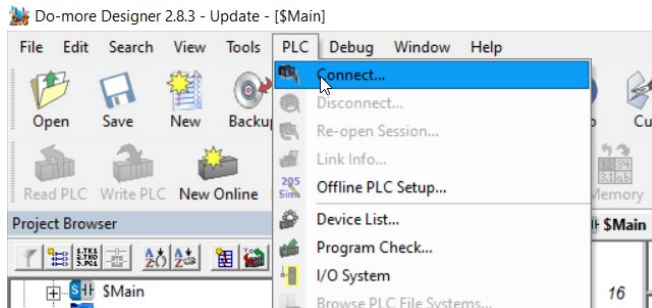


Loading the PLC Project

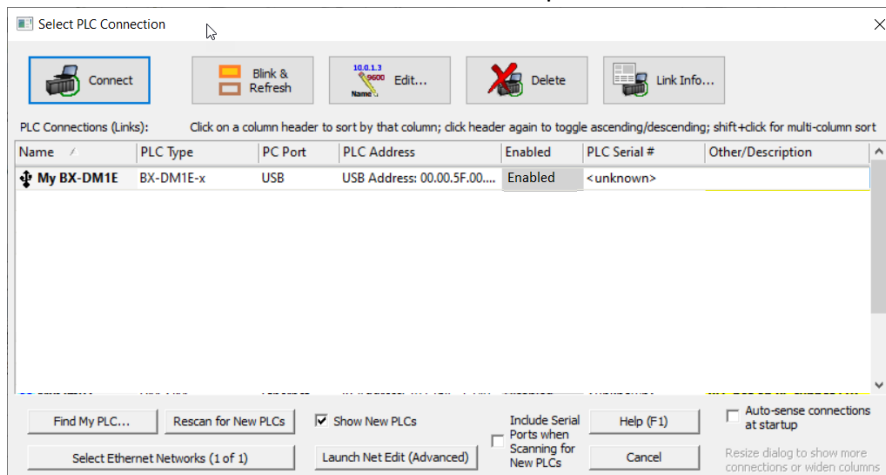
1. Plug USB cable into the powered PLC.



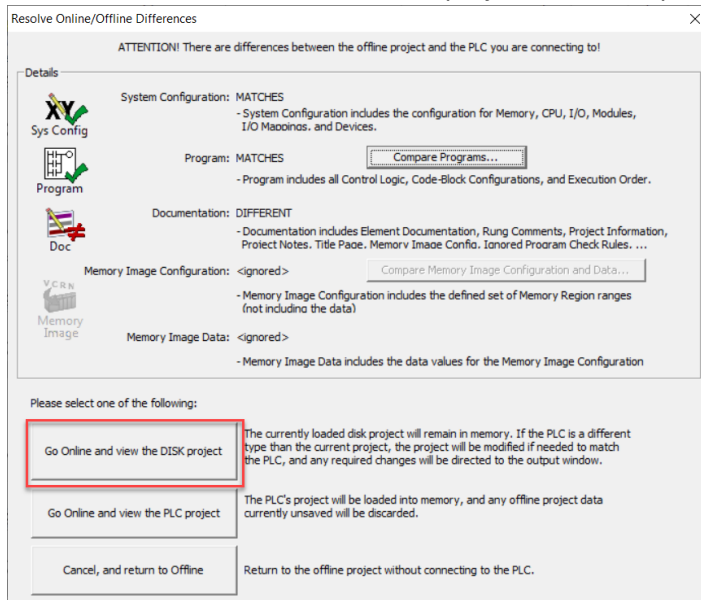
2. In the Do-more! Designer software select “**PLC>Connect**”



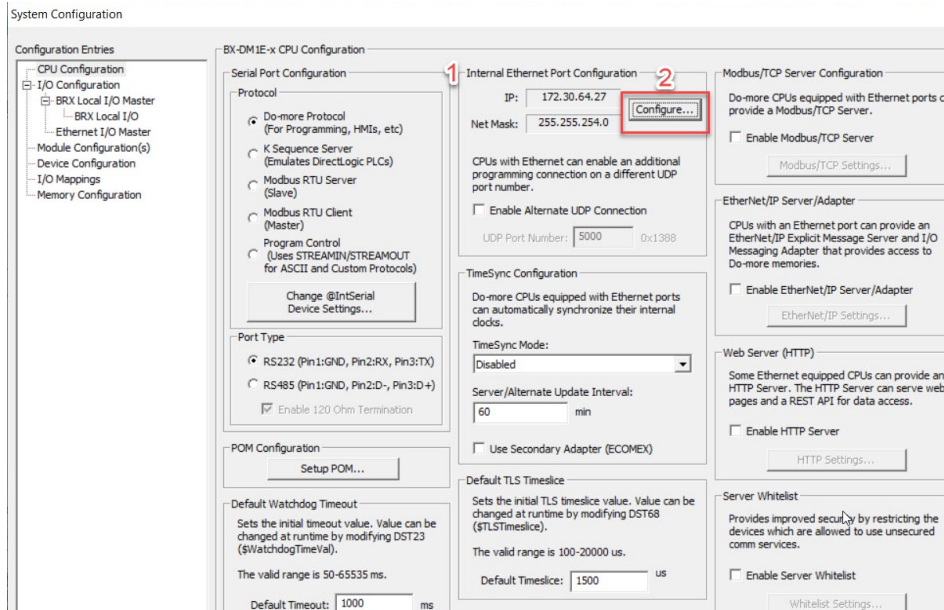
3. Select the PLC that is attached to the PC USB port and click “Connect”.



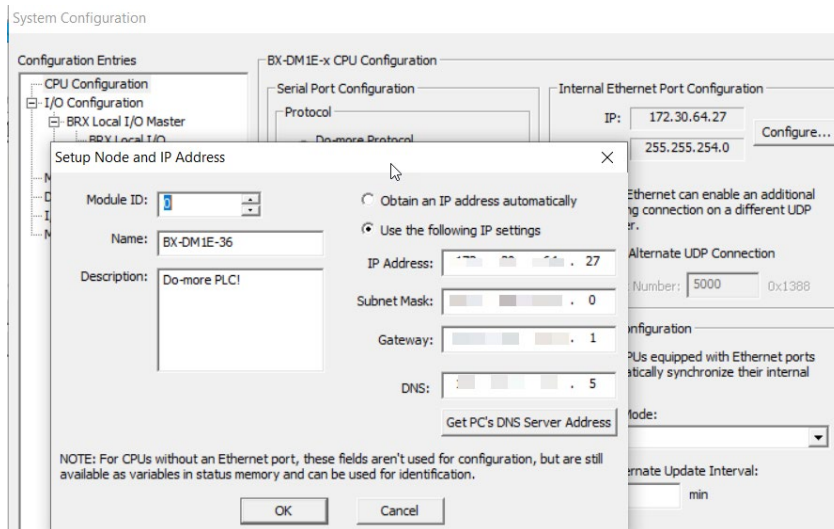
4. Select “Go Online and view the DISK project” from the proceeding dialog.



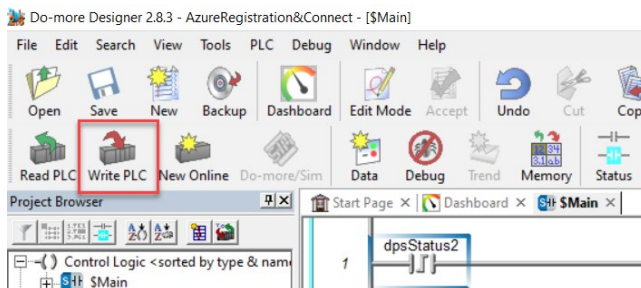
5. Select “PLC>System Configuration”. We need to set a valid IP Address that has internet access. The PLC is set to DHCP by default. Plug the PLC ethernet port in to a Ethernet switch or router.
6. From the “System Configuration” dialog, select “Configure” under Internal Ethernet Port Configuration. If you have a DHCP server, the PLC IP Address will be assigned automatically.



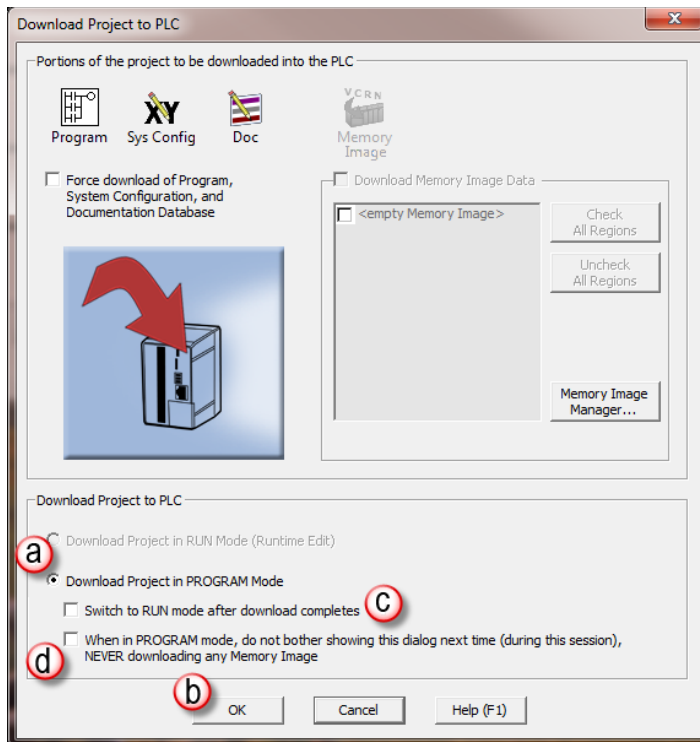
7. If using a static IP Address, Specify a unique IP address, a subnet mask, gateway IP Address, and DNS IP Address. If you are unsure of these addresses, please contact your IT department. Click “Ok” on both dialogs to accept changes. We can now write the project and configuration to the PLC.



8. From the toolbar, click “Write PLC”.



9. When the “Write PLC” button is selected and the MPU is in program mode, the Download Project to PLC pop-up window seen here appears.

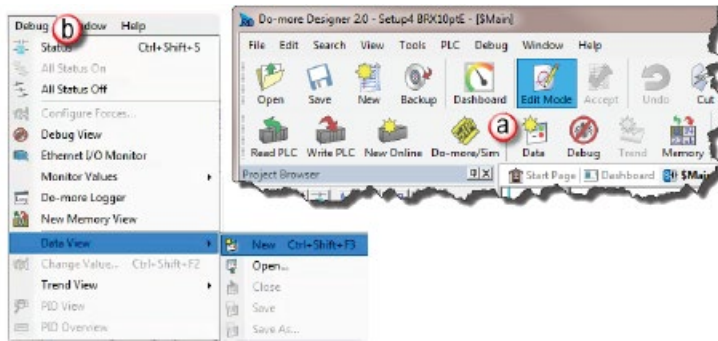


When the BRX MPU is in (a) Program mode and you click on the (b) OK button, the project is written to the MPU. When the download is complete, the MPU remains in program mode. In this window, you can select (c) to switch the MPU to RUN mode after the download is complete. You can also select (d) not to display the Download Project to PLC window if the MPU is in program mode, keeping in mind that selecting this option will not download any Memory Image. Please view the Do-more! Designer HELP topics for details on the features available in the Download Project to PLC window. **The Physical Switch on the front of the hardware must be set to TERM to be able to change PLC modes from the software.**



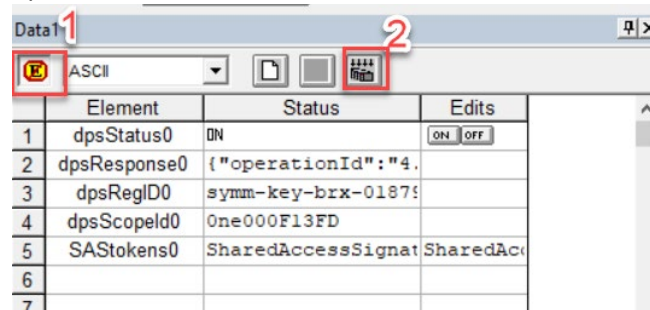
Adding the DPS SAS Token to the PLC Program

1. Start by opening a new Data window. To do this, click on the (a) Data icon on the Project Toolbar, or (b) select menu item Debug > Data View > New (Ctrl+Shift+F3).



10. In a Dataview add the following tags:

- dpsRegID0
- dpsScopeID0
- SASToken0
- dpsResponse0
- dpsStatus0

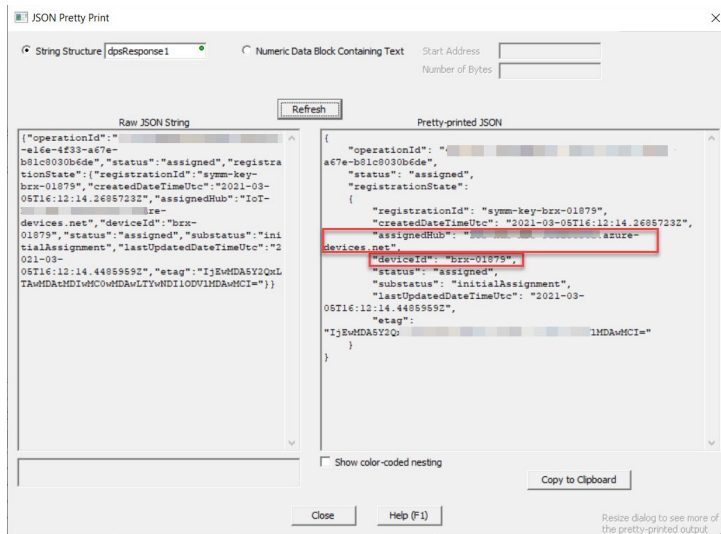


Click on (1) to enable edits.

11. Enter the Device Registration ID (dpsRegID0), Scope ID (dpsScopeID0) and the SAS Token (SASTokens0) generated and Click (2) write to PLC.
12. Select "On" for dpsStatus0 and (2) write to PLC.
13. If the device is registered, the dpsResponse0 will include an operationID and status. Ex.



14. After the device is registered, the software waits 1 minutes before rechecking the assignment. The response will include the the deviceId and IoTHub assignment. Record these values for use in creating a Device SAS Token.
15. In Do-more! Designer software select **Debug>JSON Pretty Print**. Then enter "dpsResponse1" as the String Structure and click "Refresh".



Creating a Device SAS Token

1. Using the Python code, enter the following parameters:

- URI – {IoTHubAssignment}/devices/{DeviceId}

Example: funco.azure-devices.net/devices/brx-01879

- Key – This is the same key used for the DPS registration.
- Expire – 360000 or User Define
- Policy – None. This is not needed for the Device SAS Token.

Output:


SharedAccessSignature sr=funco.azure-devices.net%2Fdevices%brx-01879&sig=GBYcRqDedY%2BBbaV41%2F1wHswpxcv%2BC6n7DyWzX5MNug%3D&se=1613937230

Adding the Device SAS Token to the PLC Program

1. In a Dataview add the following tags:

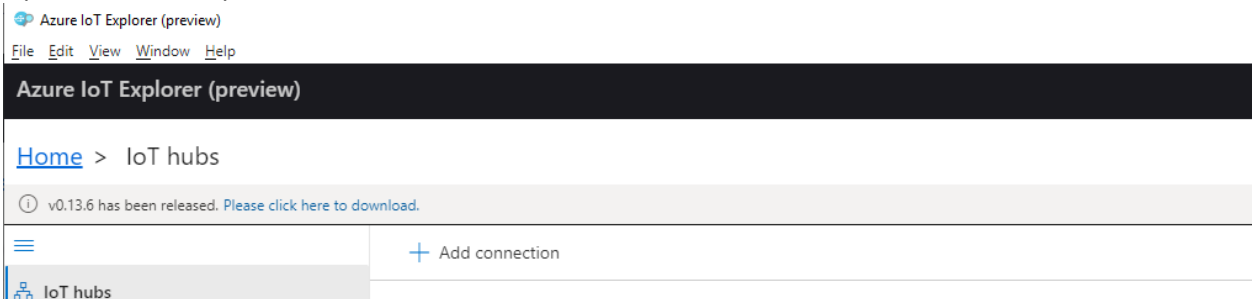
- SASToken1
- lotPUBPLO

	Element	Status	Edits
33	SAStokens1	SharedAccessSignat	
34	iotPubPL0	{"message": "hello"}	
35			
36			
37			
38			
39			
40			
41			
42			
43			

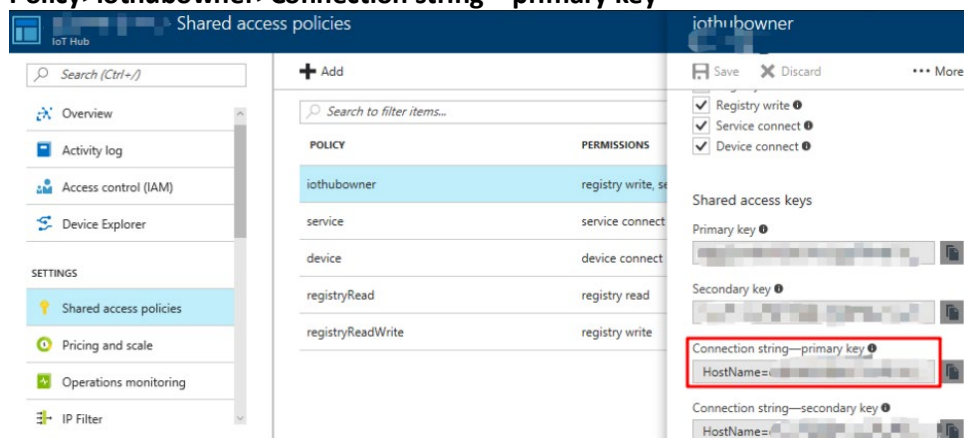
2. Enter the Device SAS Token & MQTT Publish Payload and Click  write to PLC.
3. You are Connected.

Integration with Azure IoT Explorer

1. Go to [Azure IoT Explorer](#) releases and expand the list of assets for the most recent release. Download and install the most recent version of the application.
2. Open Azure IoT Explore and Click “Add Connection”.



- a. Your IoT Hub Connection string can be found in your instance of **IoT Hub>Share Access Policy>iotHubowner>Connection string – primary key**



- b. Copy Connection String and Paste into the Azure IoT Explorer “Add connection string” dialog, and click “Save”.

×

Add connection string

Connection string *

HostName=funco.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=

Where do I get an IoT hub connection string?
Please do not save your hub connection string to any unsafe locations

Host name

funco.azure-devices.net

Shared access policy name

iothubowner

Shared access policy key

.....

- Under the newly created connection, click **“View devices in this hub”**.

Azure IoT Explorer (preview)

File Edit View Window Help

Azure IoT Explorer (preview)

Home > IoT hubs

v0.13.6 has been released. [Please click here to download.](#)

IoT hubs

IoT Plug and Play Settings

+ Add connection

Test1

✎ 🗑

Host name

1.azure-devices.net

Shared access policy name

iothubowner

Shared access policy key

.....

Connection String

.....

→ View devices in this hub

- If the device is properly configured, the device “Connection Status” should be “Connected”. Select the device that was provisioned. In this example, we will select **BRX-01879**.

Azure IoT Explorer (preview)

File Edit View Window Help

Azure IoT Explorer (preview)

Home > Devices

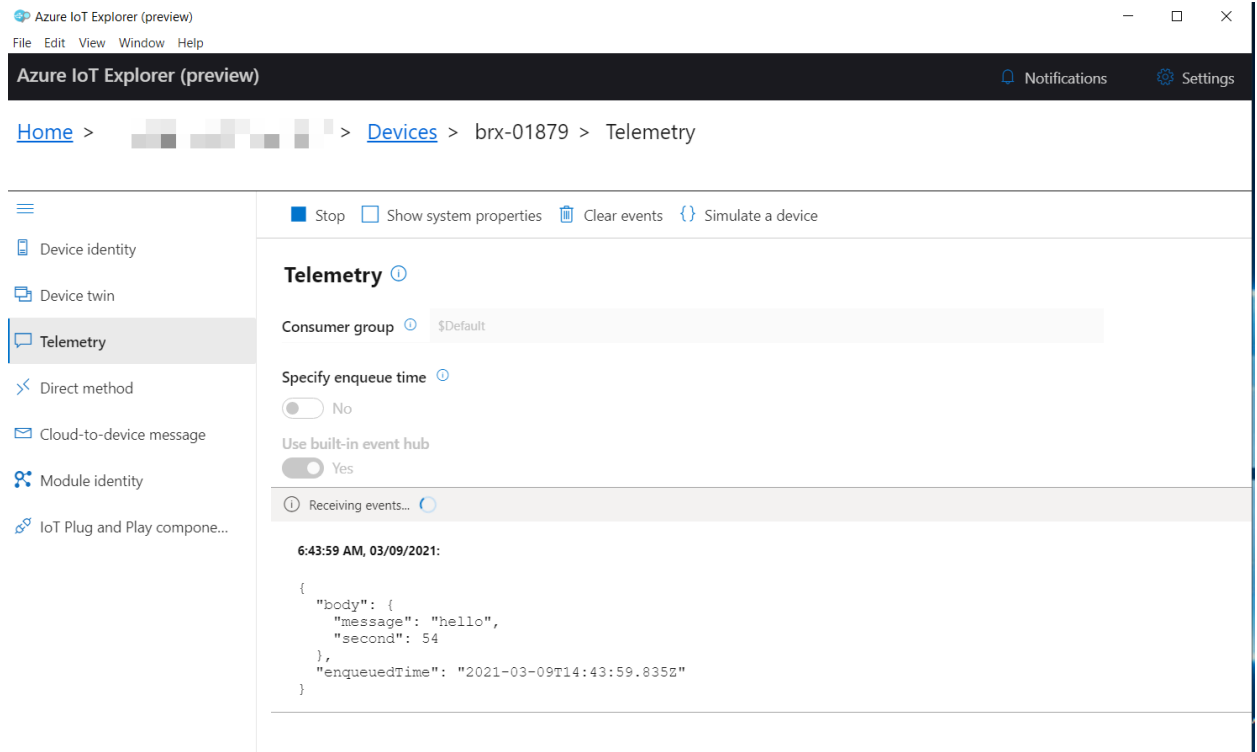
New Refresh Delete

brx

Add query parameter

Device ID	Status	Connection st...	Authentication
brx-01879	Enabled	Connected	Sas
brx-01729	Enabled	Disconnected	Sas

- In the device configuration screen, select **“Telemetry”** and click the **Start** button. This allow you to see the data feed from the remote device.



Conclusion

In this guide you learned how to format and publish data from a Do-more BRX PLC into the Azure IoT Platform. Now that the data is flowing into the Azure Cloud, there is a whole catalog of microservices. This is the foundation of building a variety of cloud-based applications. Opening the door to your organization's Industrial 4.0 journey. This journey includes technologies such as AI (machine learning (predictive maintenance, predictive control & process optimization)), remote monitoring & decentralized data storage.