D2C2D Lab Workbook

Device to Cloud to Device - a workshop for learning about Windows 10 Core IoT device development, Azure IoT Hub, Stream Analytics and automating Azure using PowerShell

|  |  |
| --- | --- |
| Step | Details |
| 0 | Requirements   * Azure Account - <https://azure.microsoft.com/en-us/> * Visual Studio 2015 - <https://www.visualstudio.com/en-us/products/vs-2015-product-editions.aspx> * Azure SDK 2.8 and Azure PowerShell 1.2.1 - <https://azure.microsoft.com/en-us/downloads/> * Windows 10 Core IoT Templates - <https://visualstudiogallery.msdn.microsoft.com/55b357e1-a533-43ad-82a5-a88ac4b01dec> |
| 1 | Download the lab materials from Git Hub   * [add github location here] * Add the local *NuGets* folder to your Visual Studio environment * Follow this menu path in Visual Studio: Tools Menu 🡪 NuGet Package Manager 🡪 Manage NuGet Packages for this solution. The NuGet Package Manager Dashboard will display.        * Click the Options button in the upper right corner of the manager window to bring up the options dialog        * To add the NuGet Packages location for this repo, click the + icon to add an additional package location * Change the name to something meaningful (D2C2D Packages, for example). * Use the … button to navigate to the *NuGets* folder at the top level of the repo. * Select the folder, click Update, and then OK. Now you can switch between the online NuGet catalogs and this local NuGet catalog when making NuGet package references. Referencing shared NuGet packages is now fully integrated into your development environment. |
| 2 | Provision Service bus |
|  | * Run PowerShell console as Administrator * Navigate to the *Automation* folder of the D2C2D repo * Login to Azure using ‘Classic’ mode using your Azure credentials * Add-AzureAccount * Run the 01-Provision-ServiceBus.ps1 script. Enter the parameters as you are prompted: * .\01-Provision-ServiceBus   **Subscription**: [the name of your subscription]  **Azure Location**: [East US for example]  **Prefix**: [a unique prefix to be used in the naming of service components]  **Suffix**: [dev | tst | stg | prd]  The script will create a Service Bus Namespace called [Prefix]sbname[Suffix]. This naming format is used to help achieve a unique name.  Validate that the service bus was created by viewing the Resource Groups in the Azure Portal. Look for a resource group called ‘Default-ServiceBus-EastUS’ or similar based on the region you selected. Click through to see that your service bus was created. |
| 3 | Provision IoT Hub and Stream Analytics Job   * Login to Azure using the ‘ARM’ mode using your Azure credentials * Login-AzureRMAccount * Run the 02-Provision-IoTHub.ps1 script. Enter the parameters as you are prompted * .\02-Provision-IoTHub.ps1   **Subscription**: [the name of your subscription]  **ResourceGroup**: [the name of your resource group, d2c2d for example]  **Azure Location**: [East US for example]  **Prefix**: [a unique prefix to be used in the naming of service components]  **Suffix**: [dev | tst | stg | prd]   * Note the name of the IoT Hub and the Primary Key that are output on the screen. Copy those two values and update the ConnectionStrings.ps1 file in the Automation folder. |
|  | * Run the 03-Provision-SAJon.ps1 Script and provide the parameters as prompted: |
|  | * .\03-Provision-IoTHub.ps1   **Subscription**: [the name of your subscription]  **ResourceGroup**: [the name of your resource group, d2c2d for example]  **Azure Location**: [East US for example]  **Prefix**: [a unique prefix to be used in the naming of service components]  **Suffix**: [dev | tst | stg | prd]   * Validate that the script provision the IoTHub and Stream Analytics job by navigating to the Azure Portal Resource Groups screen. Click on the Resource Group that you created: |
| 4 | Provision a Device   * Each device that connects to IoT Hub requires a unique Id and IoT Hub needs to know that unique Id. This application will generate a unique id for yoru device and register it with IoT Hub. * Start Visual Studio and open the solution tools\IoTHubProvision * Update the App.Config setting for the setting called IoTHubConnStr. You can get the IoT Hub connection string from the Azure Portal. Use the connection associated with the *iothubowner* policy.      * Run the application. It will connect to your IoT Hub and provision a device with a unique id. If you run the app multiple times it will register multiple devices. * Open the DeviceInfo.txt file in the tools\IoTHubProvision\IoTHubProvision\bin\Debug folder. It will contain the generated Id for your device along a unique device key. These values will be used by the Device Client to securely connect to IoT Hub. |
| 5 | Create an IoT Hub Listener |
| 6 | Create a Windows 10 Universal App for your Windows 10 Core IoT device |
| 7 | Create a Windows 10 Universal App that sends commands and receives telemetry from your Windows 10 IoT Device   * Go to the Bin Maps Portal, sign in and request a developer key - <https://www.bingmapsportal.com> |
| 8 | Next Steps |