## Microsoft Cloud Solution Toolkit – Azure Application Survey Scenarios

Version: 0.1

Author: [pegrimsd@microsoft.com](mailto:pegrimsd@microsoft.com)

Date: 29/02/2024

**Change Log**

|  |  |
| --- | --- |
| **Date** | **Change Detail** |
| 29/02/2024 | Initial Version |
|  |  |
|  |  |

# VS Code Extension Scenarios

# Scenario 1: Create a New Azure Application Project

Instructions:

1. Install the Cloud Solution Toolkit VS Code extension into VS Code from the provided .vsix file.
2. Open the Cloud Solution Toolkit extension and click “Create a new project” from the command palette.
3. Follow the guided steps to select Marketplace Offer, Azure Application and then provide a project location and project name.

Expected Results:

* A new Azure Application project folder is created in the selected location and VS Code opens in this folder.
* The new project folder contains the sample templates in the “templates” folder
* Opening the Cloud Solution Toolkit extension will now display the Cloud Solution Toolkit side-bar menus.

# Scenario 2: Validate Azure Application

Instructions:

1. Open the Cloud Solution Toolkit extension in VS Code
2. Open an existing Azure Application Project or a Create a New Project as per scenario 1 instructions.
3. From the Cloud Solution Toolkit **Utility** menu select “**Validate Azure Application**”

Expected Results:

* The extension will build the bicep templates in the templates folder.
* The extension will run the ARM-TTK validation activity against the built templates.
* The user will be presented with the output in the VS Code Output window detailing PASS, FAIL and WARNINGS from the template validation.

# Scenario 3: Launch Azure Custom Deployment

Instructions:

1. Open the Cloud Solution Toolkit extension in VS Code
2. Open an existing Azure Application Project or a Create a New Project as per scenario 1 instructions.
3. From the Cloud Solution Toolkit **Utility** menu select “**Launch Custom Deployment**”

Expected Results:

* The user can follow a guided experience to connect to an Azure subscription if not already connected.
* The user can follow a guided experience to create a storage account if required.
* The extension will build the templates and upload them to the storage account.
* The Azure portal is launched, and the custom deployment UI is shown to the user.
* The user can complete the required input parameters and trigger a deployment of the resources to Azure.
* If the resource deployment is valid resources should be deployed as expected.

# Scenario 4: Generate a CreateUiDefinition.json from a set of existing bicep files

Instructions:

1. Open the Cloud Solution Toolkit extension in VS Code
2. Open an existing Azure Application Project or a Create a New Project as per scenario 1 instructions.
3. From the Solution Explorer view navigate to the templates folder
4. Right-click on the main.bicep file and from the context menu select “Generate createUiDefintion.json”

Expected Results:

* The extension will build the bicep templates.
* If a createUiDefinition.json exists user will be presented with a message to overwrite the file.
* If no existing file or overwrite existing file is selected a new createUiDefinition.json file is created
* The file should contain an entry for all parameters in the main.bicep file
  + UI element entry
  + Entry in the outputs section of the createUiDefintion.json

# Scenario 5: Create Marketplace-ready package

Instructions:

1. Open the Cloud Solution Toolkit extension in VS Code
2. Open an existing Azure Application Project or a Create a New Project as per scenario 1 instructions.
3. From the Cloud Solution Toolkit **Utility** menu select “**Create Marketplace-ready package(zip)**”

Expected Results:

* The extension will build the bicep templates in the templates folder.
* The extension will run the ARM-TTK validation activity against the built templates.
* A package.zip file will be created.
* The user will be presented with a message indicating the location of the file.
* The contents of the zip file contain the generated mainTemplate.json, the createUiDefiniton.json and the artifacts folder/files if these were present.

# Scenario 6: Connect to Partner Center and Publish Azure Application

Pre-requisites:

* Service Principal setup to allow connection to Partner Center
  + Browse to the [App management](https://aka.ms/accountexp/appMgmt) feature of Partner Center
  + Select Add new web app to create a new Microsoft Entra application.
  + Be sure to document the App ID, Account ID\*, and Key values because they'll be used in the steps below
* Create an Azure Application Offer and Plan in Partner Center

Instructions:

1. Open the Cloud Solution Toolkit extension in VS Code
2. Open an existing Azure Application project or create a new one.
3. Under the “**Connections**” pane select “**Add credentials to Partner Center**”
4. Follow the guided experience to enter the credentials for connection to Partner Center using the Service Principal configured in Partner Center.
5. Under the “**Partner Center**” pane select “**Configure Marketplace offer**”
6. Select the pre-created Offer from the list which is presented.
7. Select the required plan
8. Upload your marketplace package using the upload icon
9. Publish your offer for Preview in Partner Center using the “Publish for Preview” icon

Expected Results:

* User can connect to Partner Center and list existing Offers and Plans.
* User can upload their marketplace package to an existing offer/plan.
* The Offer status is moved to Preview status in Partner Center

# Azure CLI Extension Scenarios

# Scenario 1: Create a New Azure Application Project

Instructions:

1. Install the Cloud Solution Toolkit Azure CLI extension from the provided .whl file
   1. **az extension add –source “<path to .whl file>”**
2. Validate the installation by running:
   1. **az cloudsolution -h**
3. Create a new Azure Application project the azure-application create command
   1. **az cloudsolution azure-application create -n <projectName> -p <path where project is created>**

Expected Results:

* A new Azure Application project folder is created in the selected location.
* The new project folder contains the sample templates in the “templates” folder.

# Scenario 2: Validate Azure Application

Instructions:

1. Open the Cloud Solution Toolkit extension in VS Code
2. Open an existing Azure Application Project or a Create a New Project as per scenario 1 instructions.
3. From the Cloud Solution Toolkit **Utility** menu select “**Validate Azure Application**”

Expected Results:

* The extension will build the bicep templates in the templates folder.
* The extension will run the ARM-TTK validation activity against the built templates.
* The user will be presented with the output in the VS Code Output window detailing PASS, FAIL and WARNINGS from the template validation.

# Additional Scenarios

# Scenario 1: Open Existing Azure Application Project

Instructions:

1. Create an Azure Application project with the az cli extension if one has not already been created.
2. Use the VS Code extension to open the Azure Application project.

Expected Results:

* The project can be opened in VS Code and the Cloud Solution Toolkit sidebar menu is opened as expected.

# Scenario 2: Validate Azure Application

# (a)

Instructions:

1. Create an Azure Application project using the Cloud Solution Toolkit VS Code extension.
2. Open the project folder in the az cli environment.
3. Run the validate command on the Azure Application project.
   1. **az cloudsolution azure-application validate.**

Expected Results:

* The Cloud Solution Toolkit will build the relevant templates and run the ARM-TTK validation.
* The results from the validation are presented to the user.

# (b)

Instructions:

1. Create an Azure Application project using the Cloud Solution Toolkit Azure CLI extension.
2. Open the project folder using the Cloud Solution Toolkit VS Code extension.
3. Run the “**validate Azure Application**” command from the Cloud Solution Toolkit Utility Menu

Expected Results:

* The Azure Application project folder is opened successfully in VS Code.
* The Cloud Solution Toolkit extension side-bar menus are displayed.
* After selecting the “Validate Azure Application” command the Cloud Solution Toolkit will build the relevant templates and run the ARM-TTK validation.
* The results from the validation are presented to the user.

# Scenario 2: Use existing Bicep and createUiDefinition templates

Instructions:

1. Create or open and existing Azure Application Project in VS Code
2. From Solution explorer view navigate to the templates directory
3. Replace the sample main.bicep and createUiDefiniton.json file with your existing Azure Application infrastructure template files.
4. From the Cloud Solution Toolkit **Utility** menu select “**Validate Azure Application**”
5. From the Cloud Solution Toolkit **Utility** menu select “**Launch Custom Deployment**”

Expected Results:

* The infrastructure templates should be validated by ARM-TTK and results presented to the user in the output pane
* The Azure portal with the custom deployment window should be launched contained the users customized UI definition allowing the user to test a deployment.