
# Advanced Live Training

## Variables

Revision 1

June 20, 2024

# Contents

Variables in Nerdio Manager ..... 2
Hard-Coded Variables ..... 2
Nerdio Manager Built-in Variables ..... 2
Secure, Inherited, and Environment Variables ..... 3
Runtime Variables (Parameters) ..... 8

# Variables in Nerdio Manager

There are four methods to get, set, or use variables in Nerdio Manager scripted actions:

- Hard-coded variables
- Built-in Nerdio environment variables
- Secure, inherited, and environment variables
- Runtime variables, also known as parameters

## Hard-Coded Variables

You define hard-coded variables in your script.

```
Example: $MyVariable = 'MyValue'
```

### Consider the following notes and best practices:

- Hard-coded variables are typically defined at the beginning of your script. This makes them easier to edit when needed.
- Every time you run the script, the same value is used.

  You can define new values in one of the following ways:

  - By editing the variable in the original script.
  - By cloning the original script, and then defining new values in the script copy.
- Hard-coded variables are best for items like a download URL, which rarely need to be changed. For other use cases, consider using inherited variables instead.

## Nerdio Manager Built-in Variables

Nerdio Manager can pre-define some variables for you. These variables become available to your script when the script is run.

### Consider the following notes:

- You do not need to define the built-in variables ahead of time.
- The built-in variables are defined based on the specific context in which the script is run. For example, when running a script on a virtual machine (VM), the

`$AzureVMName` variable acquires the value that is the name of the VM running the script.

### The Nerdio Manager-defined variables are the following:

- `$HostPoolId`
- `$HostPoolName`
- `$AzureSubscriptionId`
- `$AzureSubscriptionName`
- `$AzureResourceGroupName`
- `$AzureRegionName`
- `$AzureVMName`
- `$ADUsername` (when passing the AD credentials)
- `$ADPassword` (when passing the AD credentials)
- `$DesktopUser` (when using with personal host pools)

## Secure, Inherited, and Environment Variables

### Secure Variables

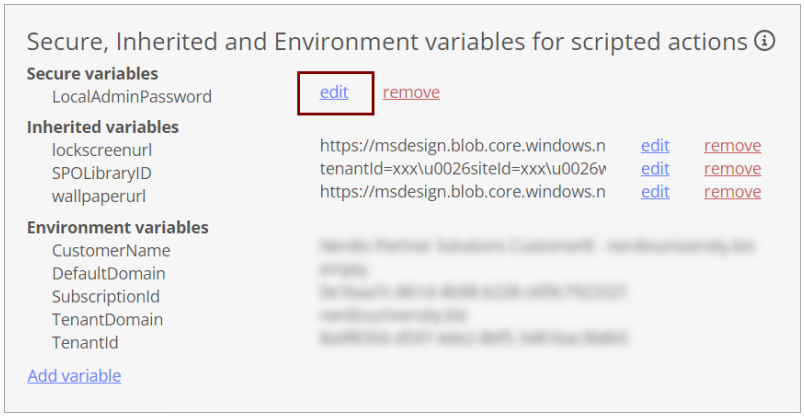
Secure variables provide a way of passing sensitive information, such as API keys, passwords, or other secrets, to your scripts securely.

Secure variables are provided in a hashtable available to your script. To retrieve the value of a secure variable from within a script, use the following format:

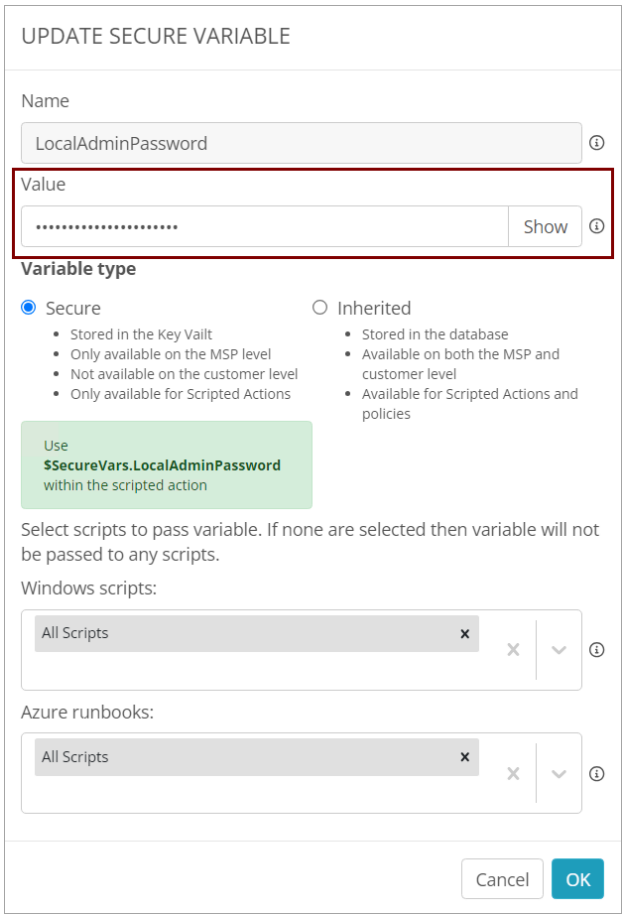
```
$SecureVars.VariableName
```

To define or edit the value of a secure variable:

1. In Nerdio Manager, at the MSP level, navigate to **Settings** > **Integrations**.
2. In the **Secure, Inherited and Environment variables for scripted actions** section, next to the variable whose value you want to modify, select **edit**.




3. In the new dialog box, in the **Value** field, edit the variable value as required.



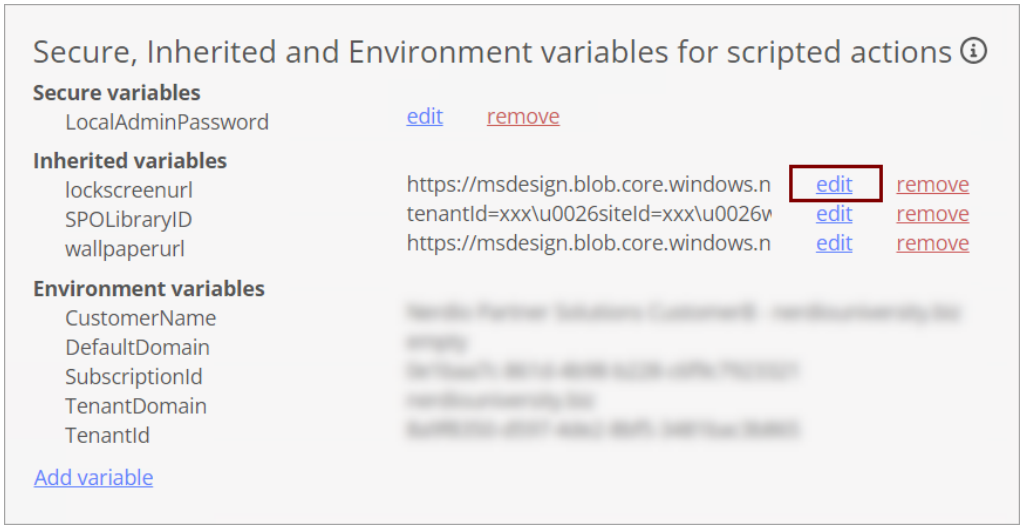

4. Select **OK**.

## Inherited Variables

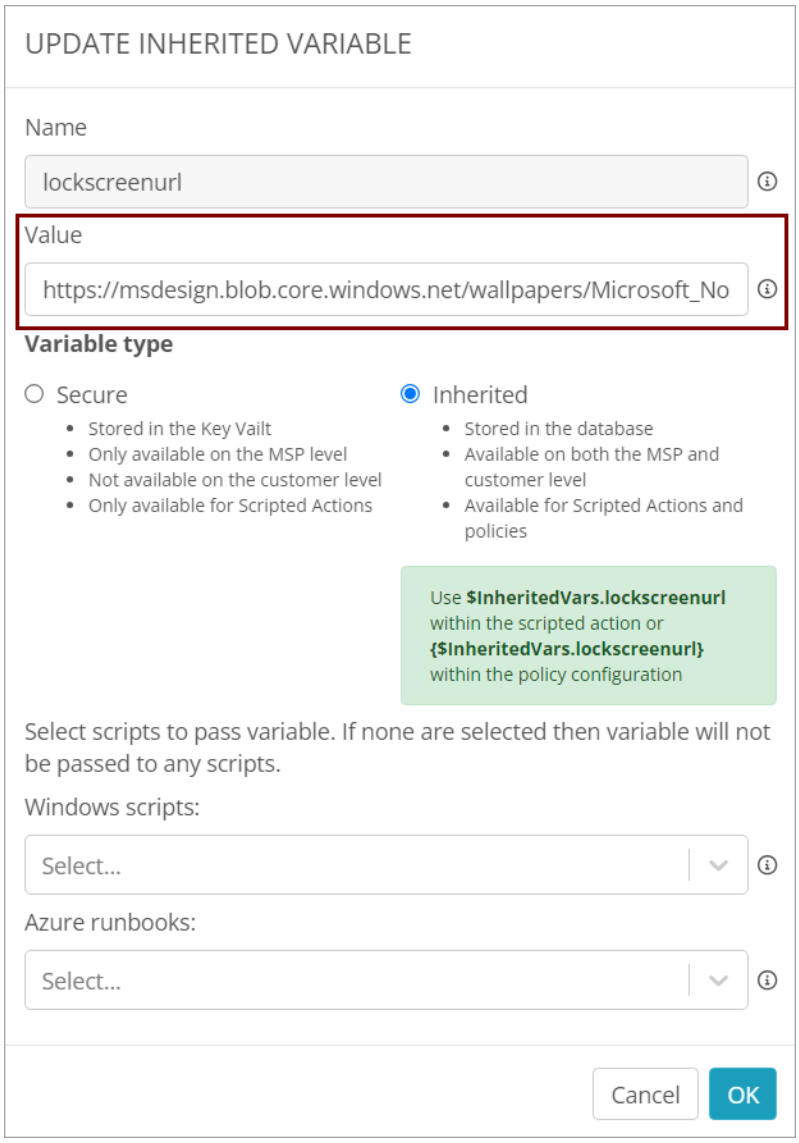
Inherited variables are defined in Nerdio Manager at the MSP level, and can be inherited at the account level. They can optionally be overridden at the account level.

**To define or edit the value of an inherited variable:**

1. In Nerdio Manager, at the MSP level or Account level, navigate to **Settings** > **Integrations**.
2. In the **Secure, Inherited and Environment variables for scripted actions** section, next to the variable whose value you want to modify, select **edit**.



3. In the new dialog box, in the **Value** field, edit the variable value as required.




4. Select **OK**.

Similar to secure variables, inherited variables are also defined in a hashtable available to your script. To use the value of an inherited variable inside a script, use the following format:

```
$InheritedVars.VariableName
```

### Environment Variables

Environment variables provide information specific to the environment where the script is running, whether at the MSP level or an account level.

The following environment variables are available:

- `CustomerName`
- `DefaultDomain`
- `SubscriptionId`
- `TenantDomain`
- `TenantId`

For an environment variable, use the following hashtable notation:

```
$EnvironmentalVars.VariableName
```

**Note:** For more details about secure, inherited, and environment variables, see:

- Scripted Actions - MSP-Level Variables
- Scripted Actions - Account-Level Variables

## Runtime Variables (Parameters)

Runtime variables are currently available only in Azure runbooks. These variables allow you to create a script that expects specific parameters to be passed when the script is run.

You define runtime variables in a `json` notation within a comment block named `Variables`.

The variables definition block looks as follows:

```
<# Variables:
{
  "Param1": {
    "Description": "The first parameter",
    "IsRequired": true
  },
  "Param2": {
    "Description": "An optional parameter",
    "IsRequired": false
  }
}
#>
```