Parameters in ARM templates

Resource Manager resolves parameter values before starting the deployment operations. Wherever the parameter is used in the template, Resource Manager replaces it with the resolved value.

Secure parameters

You can mark string or object parameters as secure. The value of a secure parameter isn't saved to the deployment history and isn't logged.

Each parameter must be set to one of the [data types](#).

```json
"parameters": {
  "demoString": {
    "type": "string"
    "type": "int"
    "type": "bool"
    "type": "object"
    "type": "array"
    "type": "secureObject"
    "type": "secureString"
  }
}
```

Allowed values

provide the allowed values in an array. The deployment fails during validation if a value is passed in for the parameter that isn't one of the allowed values.

```json
"parameters": {
  "demoEnum": {
    "type": "string",
    "allowedValues": [
      "one",
      "two"
    ]
```

Length constraints

specify minimum and maximum lengths for string and array parameters. You can set one or both constraints. For strings, the length indicates the number of characters. For arrays, the length indicates the number of items in the array

```json
"parameters": {
  "storageAccountName": {
    "type": "string",
    "minLength": 3,
    "maxLength": 24
  },
```

Integer constraints

```json
"parameters": {
  "month": {
    "type": "int",
    "minValue": 1,
    "maxValue": 12
  }
```

Variables in ARM templates

don't specify a [data type](#) for the variable. Instead provide a value or template expression.

```
"variables": {
  "stringVar": "myVariable",
  "concatToVar": "[concat(variables('stringVar'), '-addtovar') ]",
  "concatToParam": "[concat(parameters('inputValue'), '-addtoparam')]"
}
```

```
"variables": {
  "storageName": "[concat(toLower(parameters('storageNamePrefix')),
uniqueString(resourceGroup().id))]"
},
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "name": "[variables('storageName')]",
    ...
  }
```