[#tokens-string] == Working with string-encoded tokens

String-encoded tokens look like the following.

== [source,none,subs=“verbatim,attributes”]

## ${TOKEN[Bucket.Name.1234]}

They can be passed around like regular strings, and can be concatenated, as shown in the following example.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const functionName = bucket.bucketName + ‘Function’; —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const functionName = bucket.bucketName + ‘Function’; —

Python:: + [source,python,subs=“verbatim,attributes”] — function\_name = bucket.bucket\_name + “Function” —

Java:: + [source,java,subs=“verbatim,attributes”] — String functionName = bucket.getBucketName().concat(“Function”); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — string functionName = bucket.BucketName + “Function”; —

Go:: + [source,go,subs=“verbatim,attributes”] — functionName := \*bucket.BucketName() + “Function” — ====

You can also use string interpolation, if your language supports it, as shown in the following example.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const functionName = ${bucket.bucketName}Function; —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const functionName = ${bucket.bucketName}Function; —

Python:: + [source,python,subs=“verbatim,attributes”] — function\_name = f”{bucket.bucket\_name}Function” —

Java:: + [source,java,subs=“verbatim,attributes”] — String functionName = String.format(“%sFunction”. bucket.getBucketName()); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — string functionName = $"${bucket.bucketName}Function”; —

Go:: Use fmt.Sprintf for similar functionality: + [source,go,subs=“verbatim,attributes”] — functionName := fmt.Sprintf(“%sFunction”, \*bucket.BucketName()) — ====

Avoid manipulating the string in other ways. For example, taking a substring of a string is likely to break the string token.