Extensions

CEL extensions are a related set of constants, functions, macros, or other features which may not be covered by the core CEL spec.

Encoders

Encoding utilies for marshalling data into standardized representations.

Base64.Decode

Decodes base64-encoded string to bytes.

This function will return an error if the string input is not base64-encoded.

```
base64.decode(<string>) -> <bytes>
```

Examples:

```
base64.decode('aGVsbG8=')  // return b'hello'
base64.decode('aGVsbG8')   // error
```

Base64.Encode

Encodes bytes to a base64-encoded string.

```
base64.encode(<bytes>)  -> <string>
```

Example:

```
base64.encode(b'hello') // return 'aGVsbG8='
```

Strings

Extended functions for string manipulation. As a general note, all indices are zero-based.

CharAt

Returns the character at the given position. If the position is negative, or greater than the length of the string, the function will produce an error:

```
<string>.charAt(<int>) -> <string>
```

Examples:

```
'hello'.charAt(4)  // return 'o'
'hello'.charAt(5)  // return ''
'hello'.charAt(-1) // error
```

IndexOf

Returns the integer index of the first occurrence of the search string. If the search string is not found the function returns -1.

The function also accepts an optional position from which to begin the substring search. If the substring is the empty string, the index where the search starts is returned (zero or custom).

```
<string>.indexOf(<string>) -> <int>
<string>.indexOf(<string>, <int>) -> <int>
```

Examples:

```
'hello mellow'.indexOf('')         // returns 0
'hello mellow'.indexOf('ello')     // returns 1
'hello mellow'.indexOf('jello')    // returns -1
'hello mellow'.indexOf('', 2)      // returns 2
'hello mellow'.indexOf('ello', 2)  // returns 7
'hello mellow'.indexOf('ello', 20) // error
```

LastIndexOf

Returns the integer index of the last occurrence of the search string. If the search string is not found the function returns -1.

The function also accepts an optional position which represents the last index to be considered as the beginning of the substring match. If the substring is the empty string, the index where the search starts is returned (string length or custom).

```
<string>.lastIndexOf(<string>) -> <int>
<string>.lastIndexOf(<string>, <int>) -> <int>
```

Examples:

```
'hello mellow'.lastIndexOf('')         // returns 12
'hello mellow'.lastIndexOf('ello')     // returns 7
'hello mellow'.lastIndexOf('jello')    // returns -1
'hello mellow'.lastIndexOf('ello', 6)  // returns 1
'hello mellow'.lastIndexOf('ello', -1) // error
```

LowerAscii

Returns a new string where all ASCII characters are lower-cased.

This function does not perform Unicode case-mapping for characters outside the ASCII range.

```
 <string>.lowerAscii() -> <string>
```

Examples:

```
'TacoCat'.lowerAscii()      // returns 'tacocat'
'TacoCÆt Xii'.lowerAscii()  // returns 'tacocÆt xii'
```

Replace

Returns a new string based on the target, which replaces the occurrences of a search string with a replacement string if present. The function accepts an optional limit on the number of substring replacements to be made.

When the replacement limit is 0, the result is the original string. When the limit is a negative number, the function behaves the same as replace all.

```
<string>.replace(<string>, <string>) -> <string>
<string>.replace(<string>, <string>, <int>) -> <string>
```

Examples:

```
'hello hello'.replace('he', 'we')     // returns 'wello wello'
'hello hello'.replace('he', 'we', -1) // returns 'wello wello'
'hello hello'.replace('he', 'we', 1)  // returns 'wello hello'
'hello hello'.replace('he', 'we', 0)  // returns 'hello hello'
```

Split

Returns a list of strings split from the input by the given separator. The function accepts an optional argument specifying a limit on the number of substrings produced by the split.

When the split limit is 0, the result is an empty list. When the limit is 1, the result is the target string to split. When the limit is a negative number, the function behaves the same as split all.

```
<string>.split(<string>) -> <list<string>>
<string>.split(<string>, <int>) -> <list<string>>
```

Examples:

```
'hello hello hello'.split(' ')     // returns ['hello', 'hello', 'hello']
'hello hello hello'.split(' ', 0)  // returns []
'hello hello hello'.split(' ', 1)  // returns ['hello hello hello']
'hello hello hello'.split(' ', 2)  // returns ['hello', 'hello hello']
'hello hello hello'.split(' ', -1) // returns ['hello', 'hello', 'hello']
```

Substring

Returns the substring given a numeric range corresponding to character positions. Optionally may omit the trailing range for a substring from a given character position until the end of a string.

Character offsets are 0-based with an inclusive start range and exclusive end range. It is an error to specify an end range that is lower than the start range, or for either the start or end index to be negative or exceed the string length.

```
<string>.substring(<int>) -> <string>
<string>.substring(<int>, <int>) -> <string>
```

Examples:

```
'tacocat'.substring(4)    // returns 'cat'
'tacocat'.substring(0, 4) // returns 'taco'
'tacocat'.substring(-1)   // error
'tacocat'.substring(2, 1) // error
```

Trim

Returns a new string which removes the leading and trailing whitespace in the target string. The trim function uses the Unicode definition of whitespace which does not include the zero-width spaces. See: https://en.wikipedia.org/wiki/Whitespace_character#Unicode

```
<string>.trim() -> <string>
```

Examples:

```
'  \ttrim\n    '.trim() // returns 'trim'
```

UpperAscii

Returns a new string where all ASCII characters are upper-cased.

This function does not perform Unicode case-mapping for characters outside the ASCII range.

```
<string>.upperAscii() -> <string>
```

Examples:

```
 'TacoCat'.upperAscii()      // returns 'TACOCAT'
 'TacoCÆt Xii'.upperAscii()  // returns 'TACOCÆT XII'
```