Namespace Extism.Sdk

Classes

ByteArrayWasmSource
: Wasm Source represented by raw bytes.

CompiledPlugin
: A pre-compiled plugin ready to be instantiated.

CurrentPlugin
: Represents the current plugin. Can only be used within HostFunctions.

ExtismException
: Represents errors that occur during calling Extism functions.

HostFunction
: A function provided by the host that plugins can call.

Manifest
: The manifest is a description of your plugin and some of the runtime constraints to apply to it. You can think of it as a blueprint to build your plugin.

MemoryOptions
: Configures memory for the Wasm runtime. Memory is described in units of pages (64KB) and represent contiguous chunks of addressable memory.

PathWasmSource
: Wasm Source represented by a file referenced by a path.

Plugin
: Represents a WASM Extism plugin.

PluginIntializationOptions
: Options for initializing a plugin.

UrlWasmSource
: Wasm Source represented by a file referenced by a path.

WasmSource
: A named Wasm source.

Enums

HttpMethod

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource.

LogLevel
: Extism Log Levels

Delegates

ExtismFunction
: A host function signature.

LoggingSink
: Custom logging callback.

Class ByteArrayWasmSource

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Wasm Source represented by raw bytes.

```
public class ByteArrayWasmSource : WasmSource
```

Inheritance

object ← WasmSource ← ByteArrayWasmSource

Inherited Members

WasmSource.Name , WasmSource.Hash , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ByteArrayWasmSource(byte[], string?, string?)

Constructor

```
public ByteArrayWasmSource(byte[] data, string? name, string? hash = null)
```

Parameters

`data` byte[]

the byte array representing the Wasm code

`name` string

`hash` string

Properties

Data

The byte array representing the Wasm code

```
[JsonPropertyName("data")]
public byte[] Data { get; }
```

Property Value

byte[]

Class CompiledPlugin

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

A pre-compiled plugin ready to be instantiated.

```
public class CompiledPlugin : IDisposable
```

Inheritance

object ← CompiledPlugin

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

CompiledPlugin(Manifest, HostFunction[], bool)

Compile a plugin from a Manifest.

```
public CompiledPlugin(Manifest manifest, HostFunction[] functions, bool withWasi)
```

Parameters

`manifest` Manifest

`functions` HostFunction[]

`withWasi` bool

Methods

CheckNotDisposed()

Throw an appropriate exception if the plugin has been disposed.

```
protected void CheckNotDisposed()
```

Exceptions

ObjectDisposedException

Dispose()

Frees all resources held by this Plugin.

```
public void Dispose()
```

Dispose(bool)

Frees all resources held by this Plugin.

```
protected virtual void Dispose(bool disposing)
```

Parameters

`disposing` bool

~CompiledPlugin()

Destructs the current Plugin and frees all resources used by it.

```
protected ~CompiledPlugin()
```

Instantiate()

Instantiate a plugin from this compiled plugin.

```
public Plugin Instantiate()
```

Returns

Plugin

Class CurrentPlugin

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Represents the current plugin. Can only be used within HostFunctions.

```
public class CurrentPlugin
```

Inheritance

object ← CurrentPlugin

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

UserData

Returns the user data object that was passed in when a HostFunction was registered.

```
[Obsolete("Use GetUserData<T> instead.")]
public nint UserData { get; }
```

Property Value

nint

Methods

AllocateBlock(long)

Allocate a memory block in the currently running plugin.

```
public long AllocateBlock(long length)
```

Parameters

`length` long

Returns

long

BlockLength(long)

Get the length of an allocated block. NOTE: this should only be called from host functions.

```
public long BlockLength(long offset)
```

Parameters

`offset` long

Returns

long

FreeBlock(long)

Frees a block of memory belonging to the current plugin.

```
public void FreeBlock(long offset)
```

Parameters

`offset` long

GetCallHostContext<T>()

Get the current plugin call's associated host context data. Returns null if call was made without host context.

```
public T? GetCallHostContext<T>()
```

Returns

T

Type Parameters

T

GetMemory()

Returns a offset to the memory of the currently running plugin. NOTE: this should only be called from host functions.

```
public long GetMemory()
```

Returns

long

GetUserData<T>()

Returns the user data object that was passed in when a HostFunction was registered.

```
public T? GetUserData<T>()
```

Returns

T

Type Parameters

T

ReadBytes(long)

Returns a span of bytes for a given block.

```
public Span<byte> ReadBytes(long offset)
```

Parameters

`offset` long

Returns

Span<byte>

ReadString(long)

Reads a string from a memory block using UTF8.

```
public string ReadString(long offset)
```

Parameters

`offset` long

Returns

string

ReadString(long, Encoding)

Reads a string form a memory block.

```
public string ReadString(long offset, Encoding encoding)
```

Parameters

`offset` long

`encoding` Encoding

Returns

string

WriteBytes(long, Span<byte>)

Writes a byte array into a block of memory.

```
public void WriteBytes(long offset, Span<byte> bytes)
```

Parameters

`offset` long

`bytes` Span<byte>

WriteBytes(Span<byte>)

Writes a byte array into a newly allocated block of memory.

```
public long WriteBytes(Span<byte> bytes)
```

Parameters

`bytes` Span<byte>

Returns

long

Returns the offset of the allocated block

WriteString(string)

Writes a string into the current plugin memory using UTF-8 encoding and returns the offset of the block.

```
public long WriteString(string value)
```

Parameters

`value` string

Returns

long

WriteString(string, Encoding)

Writes a string into the current plugin memory and returns the offset of the block.

```
public long WriteString(string value, Encoding encoding)
```

Parameters

`value` string

`encoding` Encoding

Returns

long

Class ExtismException

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Represents errors that occur during calling Extism functions.

```
public class ExtismException : Exception, ISerializable
```

Inheritance

object ← Exception ← ExtismException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

ExtismException()

Initializes a new instance of the ExtismException class.

```
public ExtismException()
```

ExtismException(string)

Initializes a new instance of the ExtismException class with a specified error message.

```
public ExtismException(string message)
```

Parameters

`message` string

The message that describes the error .

ExtismException(string, Exception)

Initializes a new instance of the ExtismException class with a specified error message and a reference to the inner exception that is the cause of this exception.

```
public ExtismException(string message, Exception innerException)
```

Parameters

`message` string

The message that describes the error.

`innerException` Exception

The exception that is the cause of the current exception, or a null reference (Nothing in Visual Basic) if no inner exception is specified.

Delegate ExtismFunction

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

A host function signature.

```
public delegate void ExtismFunction(CurrentPlugin plugin, Span<ExtismVal> inputs,
Span<ExtismVal> outputs)
```

Parameters

`plugin` CurrentPlugin

Plugin Index

`inputs` Span<ExtismVal>

Input parameters

`outputs` Span<ExtismVal>

Output parameters, the host function can change this.

Class HostFunction

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

A function provided by the host that plugins can call.

```
public class HostFunction : IDisposable
```

Inheritance

object ← HostFunction

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

HostFunction(string, Span<ExtismValType>, Span<ExtismValType>, object?, ExtismFunction)

Registers a Host Function.

```
public HostFunction(string functionName, Span<ExtismValType> inputTypes,
Span<ExtismValType> outputTypes, object? userData, ExtismFunction hostFunction)
```

Parameters

`functionName` string

The literal name of the function, how it would be called from a Plugin.

`inputTypes` Span<ExtismValType>

The types of the input arguments/parameters the Plugin caller will provide.

`outputTypes` Span<ExtismValType>

The types of the output returned from the host function to the Plugin.

`userData` object

A state object that will be preserved and can be retrieved during function execution using GetUserData<T>(). This allows you to maintain context between function calls.

`hostFunction` ExtismFunction

Methods

CheckNotDisposed()

Throw an appropriate exception if the Host Function has been disposed.

```
protected void CheckNotDisposed()
```

Exceptions

ObjectDisposedException

Dispose()

Frees all resources held by this Host Function.

```
public void Dispose()
```

Dispose(bool)

Frees all resources held by this Host Function.

```
protected virtual void Dispose(bool disposing)
```

Parameters

`disposing` bool

~HostFunction()

Destructs the current Host Function and frees all resources used by it.

```
protected ~HostFunction()
```

FromMethod(string, object, Action<CurrentPlugin>)

Registers a HostFunction from a method that takes no parameters an returns no values.

```
public static HostFunction FromMethod(string functionName, object userData,
Action<CurrentPlugin> callback)
```

Parameters

`functionName` string

The literal name of the function, how it would be called from a Plugin.

`userData` object

A state object that will be preserved and can be retrieved during function execution using GetUserData<T>(). This allows you to maintain context between function calls.

`callback` Action<CurrentPlugin>

The host function implementation.

Returns

HostFunction

FromMethod<I1>(string, object, Action<CurrentPlugin, I1>)

Registers a HostFunction from a method that takes 1 parameter an returns no values. Supported parameter types: int, uint, long, ulong, float, double

```
public static HostFunction FromMethod<I1>(string functionName, object userData,
Action<CurrentPlugin, I1> callback) where I1 : struct
```

Parameters

`functionName` string

The literal name of the function, how it would be called from a Plugin.

`userData` object

A state object that will be preserved and can be retrieved during function execution using GetUserData<T>(). This allows you to maintain context between function calls.

`callback` Action<CurrentPlugin, I1>

The host function implementation.

Returns

HostFunction

Type Parameters

`I1`

Type of first parameter. Supported parameter types: int, uint, long, ulong, float, double

FromMethod<R>(string, object, Func<CurrentPlugin, R>)

Registers a HostFunction from a method that takes no parameters an returns a value. Supported return types: int, uint, long, ulong, float, double

```
public static HostFunction FromMethod<R>(string functionName, object userData,
Func<CurrentPlugin, R> callback) where R : struct
```

Parameters

`functionName` string

The literal name of the function, how it would be called from a Plugin.

`userData` object

A state object that will be preserved and can be retrieved during function execution using GetUserData<T>(). This allows you to maintain context between function calls.

`callback` Func<CurrentPlugin, R>

The host function implementation.

Returns

HostFunction

Type Parameters

`R`

Type of the first parameter. Supported parameter types: int, uint, long, ulong, float, double

FromMethod<I1, I2>(string, object, Action<CurrentPlugin, I1, I2>)

Registers a HostFunction from a method that takes 2 parameters an returns no values. Supported parameter types: int, uint, long, ulong, float, double

```
public static HostFunction FromMethod<I1, I2>(string functionName, object userData,
Action<CurrentPlugin, I1, I2> callback) where I1 : struct where I2 : struct
```

Parameters

`functionName` string

The literal name of the function, how it would be called from a Plugin.

`userData` object

A state object that will be preserved and can be retrieved during function execution using GetUserData<T>(). This allows you to maintain context between function calls.

`callback` Action<CurrentPlugin, I1, I2>

The host function implementation.

Returns

HostFunction

Type Parameters

`I1`

Type of the first parameter. Supported parameter types: int, uint, long, ulong, float, double

`I2`

Type of the second parameter. Supported parameter types: int, uint, long, ulong, float, double

FromMethod<I1, R>(string, object, Func<CurrentPlugin, I1, R>)

Registers a HostFunction from a method that takes 1 parameter an returns a value. Supported return and parameter types: int, uint, long, ulong, float, double

```
public static HostFunction FromMethod<I1, R>(string functionName, object userData,
Func<CurrentPlugin, I1, R> callback) where I1 : struct where R : struct
```

Parameters

`functionName` string

The literal name of the function, how it would be called from a Plugin.

`userData` object

A state object that will be preserved and can be retrieved during function execution using GetUserData<T>(). This allows you to maintain context between function calls.

`callback` Func<CurrentPlugin, I1, R>

The host function implementation.

Returns

HostFunction

Type Parameters

`I1`

Type of the first parameter. Supported parameter types: int, uint, long, ulong, float, double

`R`

Type of the first parameter. Supported parameter types: int, uint, long, ulong, float, double

FromMethod<I1, I2, I3>(string, object, Action<CurrentPlugin, I1, I2, I3>)

Registers a HostFunction from a method that takes 3 parameters an returns no values. Supported parameter types: int, uint, long, ulong, float, double

```
public static HostFunction FromMethod<I1, I2, I3>(string functionName, object userData, Action<CurrentPlugin, I1, I2, I3> callback) where I1 : struct where I2 : struct where I3 : struct
```

Parameters

`functionName` string

The literal name of the function, how it would be called from a Plugin.

`userData` object

A state object that will be preserved and can be retrieved during function execution using GetUserData<T>(). This allows you to maintain context between function calls.

`callback` Action<CurrentPlugin, I1, I2, I3>

The host function implementation.

Returns

HostFunction

Type Parameters

`I1`

Type of the first parameter. Supported parameter types: int, uint, long, ulong, float, double

`I2`

Type of the second parameter. Supported parameter types: int, uint, long, ulong, float, double

`I3`

Type of the third parameter. Supported parameter types: int, uint, long, ulong, float, double

FromMethod<I1, I2, R>(string, object, Func<CurrentPlugin, I1, I2, R>)

Registers a HostFunction from a method that takes 2 parameter an returns a value. Supported return and parameter types: int, uint, long, ulong, float, double

```
public static HostFunction FromMethod<I1, I2, R>(string functionName, object
userData, Func<CurrentPlugin, I1, I2, R> callback) where I1 : struct where I2 :
struct where R : struct
```

Parameters

`functionName` string

The literal name of the function, how it would be called from a Plugin.

`userData` object

A state object that will be preserved and can be retrieved during function execution using GetUserData<T>(). This allows you to maintain context between function calls.

`callback` Func<CurrentPlugin, I1, I2, R>

The host function implementation.

Returns

HostFunction

Type Parameters

`I1`

Type of the first parameter. Supported parameter types: int, uint, long, ulong, float, double

`I2`

Type of the second parameter. Supported parameter types: int, uint, long, ulong, float, double

`R`

Type of the first parameter. Supported parameter types: int, uint, long, ulong, float, double

FromMethod<I1, I2, I3, R>(string, object, Func<CurrentPlugin, I1, I2, I3, R>)

Registers a HostFunction from a method that takes 3 parameter an returns a value. Supported return and parameter types: int, uint, long, ulong, float, double

```
public static HostFunction FromMethod<I1, I2, I3, R>(string functionName, object userData, Func<CurrentPlugin, I1, I2, I3, R> callback) where I1 : struct where I2 : struct where I3 : struct where R : struct
```

Parameters

`functionName` string

The literal name of the function, how it would be called from a Plugin.

`userData` object

A state object that will be preserved and can be retrieved during function execution using GetUserData<T>(). This allows you to maintain context between function calls.

`callback` Func<CurrentPlugin, I1, I2, I3, R>

The host function implementation.

Returns

HostFunction

Type Parameters

`I1`

Type of the first parameter. Supported parameter types: int, uint, long, ulong, float, double

`I2`

Type of the second parameter. Supported parameter types: int, uint, long, ulong, float, double

`I3`

Type of the third parameter. Supported parameter types: int, uint, long, ulong, float, double

`R`

Type of the first parameter. Supported parameter types: int, uint, long, ulong, float, double

SetNamespace(string)

Sets the function namespace. By default it's set to `env`.

```
public void SetNamespace(string ns)
```

Parameters

`ns` string

WithNamespace(string)

Sets the function namespace. By default it's set to `extism:host/user`.

```
public HostFunction WithNamespace(string ns)
```

Parameters

`ns` string

Returns

HostFunction

Enum HttpMethod

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource.

```
public enum HttpMethod
```

Fields

`CONNECT = 5`

The CONNECT method establishes a tunnel to the server identified by the target resource.

`DELETE = 4`

The DELETE method deletes the specified resource.

`GET = 0`

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

`HEAD = 1`

The HEAD method asks for a response identical to a GET request, but without the response body.

`OPTIONS = 6`

The OPTIONS method describes the communication options for the target resource.

`PATCH = 8`

The PATCH method applies partial modifications to a resource.

`POST = 2`

The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.

`PUT = 3`

The PUT method replaces all current representations of the target resource with the request payload.

`TRACE = 7`

The TRACE method performs a message loop-back test along the path to the target resource.

Enum LogLevel

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Extism Log Levels

```
public enum LogLevel
```

Fields

`Debug = 4`

Designates lower priority information.

`Error = 1`

Designates very serious errors.

`Info = 3`

Designates useful information.

`Trace = 5`

Designates very low priority, often extremely verbose, information.

`Warn = 2`

Designates hazardous situations.

Delegate LoggingSink

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Custom logging callback.

```
public delegate void LoggingSink(string line)
```

Parameters

`line` string

Class Manifest

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

The manifest is a description of your plugin and some of the runtime constraints to apply to it. You can think of it as a blueprint to build your plugin.

```
public class Manifest
```

Inheritance

object ← Manifest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Manifest()

Create an empty manifest.

```
public Manifest()
```

Manifest(params WasmSource[])

Create a manifest from one or more Wasm sources.

```
public Manifest(params WasmSource[] sources)
```

Parameters

`sources` WasmSource[]

Properties

AllowedHosts

List of host names the plugins can access. Example:

```
AllowedHosts = new List<string> {
    "www.example.com",
    "api.*.com",
    "example.*",
}
```

```
[JsonPropertyName("allowed_hosts")]
public IList<string> AllowedHosts { get; set; }
```

Property Value

IList<string>

AllowedPaths

List of directories that can be accessed by the plugins. Examples:

```
AllowedPaths = new Dictionary<string, string>
{
    { "/usr/plugins/1/data", "/data" }, // src, dest
    { "d:/plugins/1/data", "/data" }    // src, dest
};
```

```
[JsonPropertyName("allowed_paths")]
public IDictionary<string, string> AllowedPaths { get; set; }
```

Property Value

IDictionary<string, string>

Config

Configurations available to the plugins. Examples:

```
Config = new Dictionary<string, string>
{
    { "userId", "55" }, // key, value
    { "mySecret", "super-secret-key" } // key, value
};
```

```
[JsonPropertyName("config")]
public IDictionary<string, string> Config { get; set; }
```

Property Value

IDictionary<string, string>

MemoryOptions

Configures memory for the Wasm runtime. Memory is described in units of pages (64KB) and represent contiguous chunks of addressable memory.

```
[JsonPropertyName("memory")]
public MemoryOptions? MemoryOptions { get; set; }
```

Property Value

MemoryOptions

Sources

List of Wasm sources. See PathWasmSource and ByteArrayWasmSource.

```
[JsonPropertyName("wasm")]
public IList<WasmSource> Sources { get; set; }
```

Property Value

IList<WasmSource>

Timeout

Plugin call timeout.

```
[JsonPropertyName("timeout_ms")]
[JsonConverter(typeof(TimeSpanMillisecondsConverter))]
public TimeSpan? Timeout { get; set; }
```

Property Value

TimeSpan?

Class MemoryOptions

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Configures memory for the Wasm runtime. Memory is described in units of pages (64KB) and represent contiguous chunks of addressable memory.

```
public class MemoryOptions
```

Inheritance

object ← MemoryOptions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

MaxHttpResponseBytes

Max number of bytes allowed in an HTTP response when using extism_http_request.

```
[JsonPropertyName("max_http_response_bytes")]
public int MaxHttpResponseBytes { get; set; }
```

Property Value

int

MaxPages

Max number of pages. Each page is 64KB.

```
[JsonPropertyName("max_pages")]
```

```
public int MaxPages { get; set; }
```

Property Value

int

MaxVarBytes

Max number of bytes allowed in the Extism var store

```
[JsonPropertyName("max_var_bytes")]
public int MaxVarBytes { get; set; }
```

Property Value

int

Class PathWasmSource

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Wasm Source represented by a file referenced by a path.

```
public class PathWasmSource : WasmSource
```

Inheritance

object ← WasmSource ← PathWasmSource

Inherited Members

WasmSource.Name , WasmSource.Hash , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PathWasmSource(string, string?, string?)

Constructor

```
public PathWasmSource(string path, string? name = null, string? hash = null)
```

Parameters

`path` string

path to wasm plugin.

`name` string

`hash` string

Properties

Path

Path to wasm plugin.

```
[JsonPropertyName("path")]
public string Path { get; set; }
```

Property Value

string

Class Plugin

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Represents a WASM Extism plugin.

```
public class Plugin : IDisposable
```

Inheritance

object ← Plugin

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Plugin(Manifest, HostFunction[], PluginIntializationOptions)

Initialize a plugin from a Manifest.

```
public Plugin(Manifest manifest, HostFunction[] functions,
PluginIntializationOptions options)
```

Parameters

`manifest` Manifest

`functions` HostFunction[]

`options` PluginIntializationOptions

Plugin(Manifest, HostFunction[], bool)

Create a plugin from a Manifest.

```
public Plugin(Manifest manifest, HostFunction[] functions, bool withWasi)
```

Parameters

`manifest` Manifest

`functions` HostFunction[]

`withWasi` bool

Plugin(ReadOnlySpan<byte>, HostFunction[], bool)

Create and load a plugin from a byte array.

```
public Plugin(ReadOnlySpan<byte> wasm, HostFunction[] functions, bool withWasi)
```

Parameters

`wasm` ReadOnlySpan<byte>

A WASM module (wat or wasm) or a JSON encoded manifest.

`functions` HostFunction[]

List of host functions expected by the plugin.

`withWasi` bool

Enable/Disable WASI.

Properties

Id

Get the plugin's ID.

```
public Guid Id { get; }
```

Property Value

Guid

Methods

AllowHttpResponseHeaders()

Enable HTTP response headers in plugins using `extism:host/env::http_request`

```
public void AllowHttpResponseHeaders()
```

Call(string, ReadOnlySpan<byte>, CancellationToken?)

Calls a function in the current plugin and returns the output as a byte buffer.

```
public ReadOnlySpan<byte> Call(string functionName, ReadOnlySpan<byte> input,
CancellationToken? cancellationToken = null)
```

Parameters

`functionName` string

Name of the function in the plugin to invoke.

`input` ReadOnlySpan<byte>

A buffer to provide as input to the function.

`cancellationToken` CancellationToken?

CancellationToken used for cancelling the Extism call.

Returns

ReadOnlySpan<byte>

The output of the function call

Exceptions

ExtismException

Call(string, string, CancellationToken?)

Calls a function in the current plugin and returns the output as a UTF8 encoded string.

```
public string Call(string functionName, string input, CancellationToken?
cancellationToken = null)
```

Parameters

`functionName` string

Name of the function in the plugin to invoke.

`input` string

A string that will be UTF8 encoded and passed to the plugin.

`cancellationToken` CancellationToken?

CancellationToken used for cancelling the Extism call.

Returns

string

The output of the function as a UTF8 encoded string

CallWithHostContext<T>(string, ReadOnlySpan<byte>, T, CancellationToken?)

Calls a function in the current plugin and returns the output as a byte buffer.

```
public ReadOnlySpan<byte> CallWithHostContext<T>(string functionName,
ReadOnlySpan<byte> input, T hostContext, CancellationToken? cancellationToken
```

```
= null)
```

Parameters

`functionName` string

Name of the function in the plugin to invoke.

`input` ReadOnlySpan<byte>

A buffer to provide as input to the function.

`hostContext` T

An object that will be passed back to HostFunctions

`cancellationToken` CancellationToken?

CancellationToken used for cancelling the Extism call.

Returns

ReadOnlySpan<byte>

The output of the function call

Type Parameters

`T`

Exceptions

ExtismException

Call<TOutput>(string, string, JsonSerializerOptions?, CancellationToken?)

Calls a function on the plugin and deserializes the output as UTF8 encoded JSON.

```
public TOutput? Call<TOutput>(string functionName, string input,
JsonSerializerOptions? serializerOptions = null, CancellationToken?
```

```
cancellationToken = null)
```

Parameters

`functionName` string

Name of the function in the plugin to invoke.

`input` string

Function input.

`serializerOptions` JsonSerializerOptions

JSON serialization options used for serialization/derserialization.

`cancellationToken` CancellationToken?

CancellationToken used for cancelling the Extism call.

Returns

TOutput

Type Parameters

`TOutput`

Type of the output payload returned by the function.

Call<TOutput>(string, string, JsonTypeInfo<TOutput?>, CancellationToken?)

Calls a function on the plugin with a payload. The payload is serialized into JSON and encoded in UTF8.

```
public TOutput? Call<TOutput>(string functionName, string input,
JsonTypeInfo<TOutput?> outputJsonInfo, CancellationToken? cancellationToken = null)
```

Parameters

`functionName` string

Name of the function in the plugin to invoke.

`input` string

Function input.

`outputJsonInfo` JsonTypeInfo<TOutput>

Metadata about output type.

`cancellationToken` CancellationToken?

CancellationToken used for cancelling the Extism call.

Returns

TOutput

Type Parameters

`TOutput`

Type of the output payload returned by the function.

Call<TInput, TOutput>(string, TInput, JsonSerializerOptions?, CancellationToken?)

Calls a function on the plugin with a payload. The payload is serialized into JSON and encoded in UTF8.

```
public TOutput? Call<TInput, TOutput>(string functionName, TInput input,
JsonSerializerOptions? serializerOptions = null, CancellationToken?
cancellationToken = null)
```

Parameters

`functionName` string

Name of the function in the plugin to invoke.

`input` TInput

An object that will be serialized into JSON and passed into the function as a UTF8 encoded string.

`serializerOptions` JsonSerializerOptions

JSON serialization options used for serialization/derserialization

`cancellationToken` CancellationToken?

CancellationToken used for cancelling the Extism call.

Returns

TOutput

Type Parameters

`TInput`

Type of the input payload.

`TOutput`

Type of the output payload returned by the function.

Call<TInput, TOutput>(string, TInput, JsonTypeInfo<TInput>, JsonTypeInfo<TOutput?>, CancellationToken?)

Calls a function on the plugin with a payload. The payload is serialized into JSON and encoded in UTF8.

```
public TOutput? Call<TInput, TOutput>(string functionName, TInput input,
JsonTypeInfo<TInput> inputJsonInfo, JsonTypeInfo<TOutput?> outputJsonInfo,
CancellationToken? cancellationToken = null)
```

Parameters

`functionName` string

Name of the function in the plugin to invoke.

`input` TInput

An object that will be serialized into JSON and passed into the function as a UTF8 encoded string.

`inputJsonInfo` JsonTypeInfo<TInput>

Metadata about input type.

`outputJsonInfo` JsonTypeInfo<TOutput>

Metadata about output type.

`cancellationToken` CancellationToken?

CancellationToken used for cancelling the Extism call.

Returns

TOutput

Type Parameters

`TInput`

Type of the input payload.

`TOutput`

Type of the output payload returned by the function.

CheckNotDisposed()

Throw an appropriate exception if the plugin has been disposed.

```
protected void CheckNotDisposed()
```

Exceptions

ObjectDisposedException

ConfigureCustomLogging(LogLevel)

Enable a custom log handler, this will buffer logs until DrainCustomLogs(LoggingSink) is called.

```
public static void ConfigureCustomLogging(LogLevel level)
```

Parameters

`level` LogLevel

ConfigureFileLogging(string, LogLevel)

Set log file and level

```
public static void ConfigureFileLogging(string path, LogLevel level)
```

Parameters

`path` string

Log file path

`level` LogLevel

Minimum log level

Dispose()

Frees all resources held by this Plugin.

```
public void Dispose()
```

Dispose(bool)

Frees all resources held by this Plugin.

```
protected virtual void Dispose(bool disposing)
```

Parameters

`disposing` bool

DrainCustomLogs(LoggingSink)

Calls the provided callback function for each buffered log line. This only needed when ConfigureCustomLogging(LogLevel) is used.

```
public static void DrainCustomLogs(LoggingSink callback)
```

Parameters

`callback` LoggingSink

ExtismVersion()

Get Extism Runtime version.

```
public static string ExtismVersion()
```

Returns

string

~Plugin()

Destructs the current Plugin and frees all resources used by it.

```
protected ~Plugin()
```

FunctionExists(string)

Checks if a specific function exists in the current plugin.

```
public bool FunctionExists(string name)
```

Parameters

`name` string

Returns

bool

Reset()

Reset the Extism runtime, this will invalidate all allocated memory

```
public bool Reset()
```

Returns

bool

UpdateConfig(Dictionary<string, string>, JsonSerializerOptions)

Update plugin config values, this will merge with the existing values.

```
public bool UpdateConfig(Dictionary<string, string> value,
JsonSerializerOptions serializerOptions)
```

Parameters

`value` Dictionary<string, string>

`serializerOptions` JsonSerializerOptions

Returns

bool

UpdateConfig(ReadOnlySpan<byte>)

Update plugin config values, this will merge with the existing values.

```
public bool UpdateConfig(ReadOnlySpan<byte> json)
```

Parameters

`json` ReadOnlySpan<byte>

The configuration JSON encoded in UTF8.

Returns

bool

Class PluginIntializationOptions

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Options for initializing a plugin.

```
public class PluginIntializationOptions
```

Inheritance

object ← PluginIntializationOptions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

FuelLimit

Limits number of instructions that can be executed by the plugin.

```
public long? FuelLimit { get; set; }
```

Property Value

long?

WithWasi

Enable WASI support.

```
public bool WithWasi { get; set; }
```

Property Value

bool

Class UrlWasmSource

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

Wasm Source represented by a file referenced by a path.

```
public class UrlWasmSource : WasmSource
```

Inheritance

object ← WasmSource ← UrlWasmSource

Inherited Members

WasmSource.Name , WasmSource.Hash , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

UrlWasmSource(string, string?, string?)

Constructor

```
public UrlWasmSource(string url, string? name = null, string? hash = null)
```

Parameters

`url` string

uri to wasm plugin.

`name` string

`hash` string

UrlWasmSource(Uri, string?, string?)

Constructor

```
public UrlWasmSource(Uri url, string? name = null, string? hash = null)
```

Parameters

`url` Uri

uri to wasm plugin.

`name` string

`hash` string

Properties

Headers

HTTP headers

```
[JsonPropertyName("headers")]
public Dictionary<string, string> Headers { get; set; }
```

Property Value

Dictionary<string, string>

Method

HTTP Method

```
[JsonPropertyName("method")]
public HttpMethod? Method { get; set; }
```

Property Value

HttpMethod?

Url

Uri to wasm plugin.

```
[JsonPropertyName("url")]
public Uri Url { get; set; }
```

Property Value

Uri

Class WasmSource

Namespace: Extism.Sdk

Assembly: Extism.Sdk.dll

A named Wasm source.

```
public abstract class WasmSource
```

Inheritance

object ← WasmSource

Derived

ByteArrayWasmSource, PathWasmSource, UrlWasmSource

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Hash

Hash of the WASM source

```
[JsonPropertyName("hash")]
public string? Hash { get; set; }
```

Property Value

string

Name

Logical name of the Wasm source

```
[JsonPropertyName("name")]
public string? Name { get; set; }
```

Property Value

string

Namespace Extism.Sdk.Native

Structs

ExtismVal

`ExtismVal` holds the type and value of a function argument/return

ExtismValUnion

A union type for host function argument/return values.

Enums

ExtismValType

Represents Wasm data types that Extism can understand

Struct ExtismVal

Namespace: Extism.Sdk.Native

Assembly: Extism.Sdk.dll

`ExtismVal` holds the type and value of a function argument/return

```
public struct ExtismVal
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Fields

t

The type for the argument

```
public ExtismValType t
```

Field Value

ExtismValType

v

The value for the argument

```
public ExtismValUnion v
```

Field Value

ExtismValUnion

Enum ExtismValType

Namespace: Extism.Sdk.Native

Assembly: Extism.Sdk.dll

Represents Wasm data types that Extism can understand

```
public enum ExtismValType
```

Fields

`ExternRef = 6`

A reference to opaque data in the Wasm instance.

`F32 = 2`

Floating point 32 bit integer. Equivalent of float

`F64 = 3`

Floating point 64 bit integer. Equivalent of double

`FuncRef = 5`

A reference to opaque data in the Wasm instance.

`I32 = 0`

Signed 32 bit integer. Equivalent of int or uint

`I64 = 1`

Signed 64 bit integer. Equivalent of long or long

`PTR = 1`

A wrapper around I64 to specify arguments that are pointers to memory blocks

`V128 = 4`

A 128 bit number.

Struct ExtismValUnion

Namespace: Extism.Sdk.Native

Assembly: Extism.Sdk.dll

A union type for host function argument/return values.

```
public struct ExtismValUnion
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Fields

f32

Set this for 32 bit floats

```
public float f32
```

Field Value

float

f64

Set this for 64 bit floats

```
public double f64
```

Field Value

double

i32

Set this for 32 bit integers

```
public int i32
```

Field Value

int

i64

Set this for 64 bit integers

```
public long i64
```

Field Value

long

ptr

Set this for 64 bit integers

```
public long ptr
```

Field Value

long