Introduction

The **WebAssembly System Interface (WASI)** is a group of standards-track API specifications for software compiled to the **W3C WebAssembly (Wasm) standard**. WASI is designed to provide a secure standard interface for applications that can be compiled to Wasm from any language, and that may run anywhere—from browsers to clouds to embedded devices.

By standardizing APIs for WebAssembly, WASI provides a way to compose software written in different languages—without costly and clunky interface systems like HTTP-based microservices. We believe that every project with a plugin model should be using WASI, and that WASI is ideally suited for projects with SDKs for multiple languages, e.g. client libraries.

To date, WASI has seen two milestone releases known as **0.1** and **0.2**. (Sometimes you will see these referred to as Preview 1 and Preview 2, or P1 and P2). The concepts and vocabulary of Wasm and WASI can sometimes be opaque to newcomers, so WASI.dev serves as an introduction to WASI for users of all backgrounds. It's very much a work-in-progress, and we welcome contributions on the GitHub repo.

Who are we?

WASI is an open standard under active development by the **WASI Subgroup** in the **W3C WebAssembly Community Group**. Discussions happen in GitHub issues, pull requests, and bi-weekly Zoom meetings.

Who are you?

WASI and Wasm are tools for any type of software developer: whether you're writing web apps, plugins, serverless functions, User-Defined Functions (UDFs) in a database, embedded controller components, sidecar networking filters, or something completely different. This site is intended to make WASI understandable regardless of your background, use-case, or familiarity with the WebAssembly ecosystem.

How to get started

There are many different runtimes that support WASI including Wasmtime, WAMR, WasmEdge, wazero, Wasmer, wasmi, wasm3, and jco. Many of these runtimes have different areas of focus (i.e., IoT, embedded devices, and edge for WAMR, server-side and non-web embeddings with components for Wasmtime, and running in JS environments and browsers for Jco). The introductory documentation for each is a great place to start.

WASI can be implemented by both core Wasm modules and applications built according to the **Component Model**, a specification for Wasm applications that are interoperable and composable. You can learn more about components in the Bytecode Alliance's **WebAssembly Component Model** documentation.

Continue reading to learn more about WASI interfaces, including available APIs and how they are defined.