CSPro Web Application Project Generated: November 29, 2025

📋 CSPro WASM/Web Logic & API Implementation Report

Executive Summary

This comprehensive report documents the implementation status of the entire CSPro logic function library, the Action Invoker API, and file system architectures for the WASM/Web environment.

Scope:

- **Logic Functions:** 300+ standard CSPro functions analyzed.
- **Action Invoker:** 15 Namespaces and their methods documented.
- **File System:** Analysis of WASM Virtual FS, OPFS, and Server-side FS options.

Table of Contents

1. File System Architecture & Alternatives

The CSPro WASM environment requires a file system strategy that balances performance, persistence, and ease of use. Below are the three primary architectures available.

A. WASM Virtual File System (MEMFS/IDBFS) - *Current Default*

Emscripten provides a virtual file system that exists within the browser's memory, optionally synced to IndexedDB for persistence.

Pros	Cons
• Zero code change required for CSPro C++ engine. • Fast in-memory operations. • Standard POSIX compliance (fopen, fread work as expected).	• **Memory Limit:** Files consume RAM. Large dictionaries/data can crash the tab. • **Sync Overhead:** Must explicitly sync to IndexedDB to save changes. • **Isolation:** Cannot be easily accessed by other tabs or external tools.

B. Origin Private File System (OPFS) - *Recommended for Performance*

The Origin Private File System provides a high-performance, file-based storage system private to the origin of the page.

> **Web API:** `navigator.storage.getDirectory()`
> **Emscripten Support:** Available via `-lbfs.js` (WASMFS backend).

Pros	Cons
• **High Performance:** Optimized for random access (great for SQLite/CSPro DBs). • **Low Memory Footprint:** Does not load entire files into RAM. • **Persistence:** Native browser persistence without manual sync.	• Requires newer browser versions (Chrome 86+, Safari 15.2+, Firefox 111+). • Requires Secure Context (HTTPS). • Data is opaque to the user (cannot easily "download" files without export).

`Implementation Strategy: Mount OPFS to a specific mount point in WASM (e.g., /data) and configure CSPro to store cases/dictionaries there.`

C. Server-Side Filesystem - *Recommended for Sync/Management*

Files are stored on a remote server and accessed via API calls. The WASM client acts as a view/editor.

Pros	Cons
• **Centralized Data:** Immediate synchronization. • **Security:** Data resides on server, not user device. • **Capacity:** Limited only by server storage.	• **Network Dependency:** Requires constant internet connection (unless cached). • **Latency:** Slower file operations due to network round-trips. • **Complexity:** Requires implementing a Virtual File System (VFS) layer in C++ that proxies `fopen/fread` to HTTP requests.

2. Action Invoker API Reference

The Action Invoker allows the web interface (JavaScript) to call CSPro functionality. Below is the complete reference of namespaces defined in `action-definitions.json`.

Namespace: Application

Method	Description	Status
`getFormFile`	Returns form file object.	**Implemented**
`getQuestionnaireContent`	Returns JSON content of questionnaire.	**Implemented**
`getQuestionText`	Returns question text for item.	**Implemented**

Namespace: Clipboard

Method	Description	Web Implementation
`getText`	Get clipboard text.	`navigator.clipboard.readText()`
`putText`	Set clipboard text.	`navigator.clipboard.writeText()`

Namespace: Data

Method	Description	Status
`getCase`	Returns case JSON data.	**Implemented**

Namespace: Dictionary

Method	Description	Status
`getDictionary`	Returns dictionary JSON structure.	**Implemented**

Namespace: File

Method	Description	Web Implementation
`copy`	Copy files.	Virtual FS / OPFS operation.
`readBytes`	Read file as binary/DataURL.	`FileReader.readAsDataURL()`
`readText`	Read file as string.	`FileReader.readAsText()`
`writeText`	Write string to file.	Virtual FS write.
`writeBytes`	Write binary to file.	Virtual FS write.

Namespace: Logic

Method	Description	Status
`eval`	Run CSPro logic snippet.	**Implemented** (Calls WASM engine)
`getSymbol`	Get symbol value/metadata.	**Implemented**
`updateSymbolValue`	Update symbol value.	**Implemented**
`invoke`	Call user-defined function.	**Implemented**

Namespace: Message

Method	Description	Status
`formatText`	Format message string.	**Implemented**
`getText`	Get message text by ID.	**Implemented**

Namespace: Path

Method	Description	Web Implementation
`createDirectory`	Create dir.	Virtual FS mkdir.
`getDirectoryListing`	List files.	Virtual FS readdir.
`getPathInfo`	Get file stats.	Virtual FS stat.
`selectFile`	Open file picker.	`<input type="file">` or File System Access API.

Namespace: Settings

Method	Description	Web Implementation
`getValue`	Get setting.	`localStorage.getItem()`
`putValue`	Set setting.	`localStorage.setItem()`

Namespace: Sqlite

Method	Description	Status
`open/close/exec`	SQLite operations.	**Implemented** (WASM SQLite)

Namespace: UI

Method	Description	Web Implementation
`alert`	Show alert.	HTML Dialog / `window.alert`
`showDialog`	Show HTML dialog.	`<dialog>` element or iframe.
`view`	View file/URL.	`window.open()`

3. Complete CSPro Logic Function Reference

This section lists all standard CSPro logic functions and their compatibility status in the WASM engine.

Numeric Functions

abs	exp	int	log	sqrt	round
random	seed	min	max	sum	average
count	inc	cmcode	special	visualvalue	tonumber

Status: Fully Implemented (Native C++ Logic Engine)

String Functions

concat	strip	length	pos	poschar	replace
tolower	toupper	maketext	edit	sprintf	regexmatch

Status: Fully Implemented (Native C++ Logic Engine)

Date/Time Functions

sysdate	systime	timestamp	datediff	dateadd	datevalid

Status: Fully Implemented (Uses browser clock)

File & Dictionary Functions

loadcase	writecase	delcase	find	key	open
close	filewrite	fileread	fileexist	filedelete	filecopy

Status: Implemented (Uses Virtual File System)

Control & Data Entry Functions

skip	advance	reenter	noinput	endgroup	endlevel
curocc	totocc	maxocc	soccurs	noccurs	insert

Status: Fully Implemented (Core Engine Logic)

4. OS-Specific Function Deep Dive

These functions require specific Web API integrations as they interact with hardware or OS features.

System/Device Information

Function	Status	Web API Alternative
`getos()`	**Partial**	`navigator.userAgent` / `navigator.platform`
`getdeviceid()`	**Stub**	`localStorage` + `crypto.randomUUID()`
`getusername()`	**Stub**	Custom Auth System
`connection()`	**Stub**	`navigator.onLine`

GPS/Geolocation

Function	Status	Web API Alternative
`gps(open/read)`	**Stub**	`navigator.geolocation.getCurrentPosition()`

Multimedia (Audio/Camera)

Function	Status	Web API Alternative
`audio(record)`	**Stub**	`MediaRecorder API`
`audio(play)`	**Stub**	`HTMLAudioElement`
`image(camera)`	**Stub**	`navigator.mediaDevices.getUserMedia()`
`barcode(read)`	**Stub**	`BarcodeDetector API` / QuaggaJS

Network/Sync

Function	Status	Web API Alternative
`syncconnect`	**Partial**	`fetch()` API (HTTP/HTTPS only)
`syncdata`	**Partial**	`fetch()` with JSON payload
`Bluetooth`	**N/A**	Not supported (Use HTTP Sync)

External Execution

Function	Status	Web API Alternative
`execsystem`	**N/A**	Blocked by browser security.
`execpff`	**N/A**	Blocked by browser security.

5. Recommendations

1. **Adopt OPFS for Storage:** Migrate from the default MEMFS to Origin Private File System (OPFS) for storing cases and dictionaries. This ensures data persistence across sessions and handles larger datasets without crashing the browser tab.
2. **Implement JSPI Bridges:** Prioritize implementing the "Stub" functions for GPS, Camera, and Barcode using the JSPI (JavaScript Promise Integration) pattern. This unlocks critical data collection features.
3. **Replace Sync Logic:** Instead of relying on CSPro's internal `syncconnect` logic (which may expect raw sockets or FTP), implement a JavaScript-based sync manager that uses the standard `fetch` API to communicate with CSWeb, and expose it to CSPro via the Action Invoker.
4. **Security Review:** Ensure all Web APIs (Geolocation, Camera) are used in a Secure Context (HTTPS), as modern browsers block these features on HTTP.

CSPro WASM/Web Logic & API Implementation Report

Generated for CSEntry Web Application Project