CSPro Logic Functions & Action Invoker API Reference

Kotlin Multiplatform (KMP) Web Implementation

Document Version: 1.0
Last Updated: 2025
Platform: CSPro WASM/KMP Web Application

Table of Contents

---

1. Executive Summary

This document provides a comprehensive reference for all CSPro Logic Functions and Action Invoker API namespaces available in the Kotlin Multiplatform (KMP) web implementation. It includes:

- **45+ Action Invoker Functions** across 14 namespaces
- **450+ CSPro Logic Functions** for data entry, processing, and reporting
- **Web Porting Strategies** for functions incompatible with browser environments
- **Implementation Status** tracking for KMP port completion

Key Architecture Components

Component	Description	Location
`ActionInvoker.h`	C++ Runtime class with all action functions	`zAction/ActionInvoker.h`
`action-invoker.js`	Unified JavaScript API for web mode	`public/action-invoker.js`
`ActionInvoker.kt`	Kotlin ActionInvoker bridge	`src/wasmJsMain/kotlin/.../ActionInvoker.kt`
`WasmEngineInterface.kt`	WASM engine binding interface	`src/wasmJsMain/kotlin/.../WasmEngineInterface.kt`
`WASMBindings.cpp`	C++ to JavaScript bindings	`WASM/WASMBindings.cpp`

---

2. Action Invoker Namespaces

The Action Invoker provides a structured API for invoking CSPro functionality from JavaScript/Kotlin. Each namespace groups related functions.

2.1 Application Namespace

Purpose: Access application-level information and files.

Function	Sync	Async	Web Compatible	Description
`getFormFile`	✅	✅	⚠️ Proxy	Get the form file content
`getQuestionnaireContent`	✅	✅	⚠️ Proxy	Get questionnaire content
`getQuestionText`	✅	✅	⚠️ Proxy	Get question text for a field

Web Porting Strategy:

- Use `_webProxyAction()` to route to WASM engine
- Form files cached in OPFS during application load

C++ Signature:

```
 Result Application_getFormFile(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Application_getQuestionnaireContent(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Application_getQuestionText(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.2 Clipboard Namespace

Purpose: Read and write system clipboard.

Function	Sync	Async	Web Compatible	Description
`getText`	✅	✅	✅ Native	Read text from clipboard
`putText`	✅	✅	✅ Native	Write text to clipboard

Web Porting Strategy:

- ✅ **Fully Compatible** - Uses `navigator.clipboard` API
- Async methods preferred for web compatibility

Web Implementation:

```
getText: (args) => {
    if (self._isWebMode) {
        try {
            return navigator.clipboard.readText();
        } catch (e) {
            console.warn('Clipboard read failed');
            return '';
        }
    }
    // Native implementation...
}
```

C++ Signature:

```
Result Clipboard_getText(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Clipboard_putText(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.3 Data Namespace

Purpose: Access case data and dictionary information.

Function	Sync	Async	Web Compatible	Description
`getCase`	✅	✅	⚠️ Proxy	Get case data as JSON

Web Porting Strategy:

- Routes to WASM engine via `_webProxyAction()`
- Case data managed in engine memory

C++ Signature:

```
Result Data_getCase(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.4 Dictionary Namespace

Purpose: Access dictionary metadata and structure.

Function	Sync	Async	Web Compatible	Description
`getDictionary`	✅	✅	⚠️ Proxy	Get dictionary definition as JSON

Web Porting Strategy:

- Routes to WASM engine for dictionary access
- Dictionary cached in application state

C++ Signature:

```
Result Dictionary_getDictionary(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.5 File Namespace

Purpose: File system operations.

Function	Sync	Async	Web Compatible	Description
`copy`	✅	✅	⚠️ Proxy	Copy file
`readBytes`	✅	✅	⚠️ Proxy	Read file as bytes
`readLines`	✅	✅	⚠️ Proxy	Read file as lines array
`readText`	✅	✅	⚠️ Proxy	Read file as text
`writeBytes`	✅	✅	⚠️ Proxy	Write bytes to file
`writeLines`	✅	✅	⚠️ Proxy	Write lines to file
`writeText`	✅	✅	⚠️ Proxy	Write text to file

Web Porting Strategy:

- **OPFS (Origin Private File System)** used for persistent storage
- Sync methods return cached results or empty values
- Async methods route to WASM engine with OPFS backend
- File downloads use `URL.createObjectURL()` and programmatic download links

KMP Implementation:

```
suspend fun readFile(path: String): String {
    val virtualPath = OPFSFileSystem.resolveVirtualPath(path)
    return OPFSFileSystem.readFile(virtualPath)
}
```

C++ Signature:

```
 Result File_copy(const JsonNode<wchar_t>& json_node, Caller& caller);
Result File_readBytes(const JsonNode<wchar_t>& json_node, Caller& caller);
Result File_readLines(const JsonNode<wchar_t>& json_node, Caller& caller);
Result File_readText(const JsonNode<wchar_t>& json_node, Caller& caller);
Result File_writeBytes(const JsonNode<wchar_t>& json_node, Caller& caller);
Result File_writeLines(const JsonNode<wchar_t>& json_node, Caller& caller);
Result File_writeText(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.6 Hash Namespace

Purpose: Cryptographic hash generation.

Function	Sync	Async	Web Compatible	Description
`createHash`	✅	✅	✅ Native	Create SHA-256 hash
`createMd5`	✅	✅	✅ Native	Create MD5 hash

Web Porting Strategy:

- ✅ **Fully Compatible** - Uses Web Crypto API
- `crypto.subtle.digest()` for SHA-256
- External library or WASM for MD5 (not in Web Crypto)

Web Implementation:

```
createHash: async (args) => {
    const text = args?.text || '';
    const encoder = new TextEncoder();
    const data = encoder.encode(text);
    const hashBuffer = await crypto.subtle.digest('SHA-256', data);
    const hashArray = Array.from(new Uint8Array(hashBuffer));
    return hashArray.map(b => b.toString(16).padStart(2, '0')).join('');
}
```

C++ Signature:

```
 Result Hash_createHash(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Hash_createMd5(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.7 Localhost Namespace

Purpose: Local server mapping for HTML dialogs.

Function	Sync	Async	Web Compatible	Description
`mapActionResult`	✅	✅	⚠️ Proxy	Map action result URL
`mapFile`	✅	✅	⚠️ Proxy	Map file to localhost URL
`mapSymbol`	✅	✅	⚠️ Proxy	Map symbol to localhost URL
`mapText`	✅	✅	⚠️ Proxy	Map text to localhost URL

Web Porting Strategy:

- Uses Blob URLs (`URL.createObjectURL()`) instead of localhost server
- File content loaded from OPFS and converted to Blob
- Temporary URLs cleaned up after dialog closes

C++ Signature:

```
 Result Localhost_mapActionResult(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Localhost_mapFile(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Localhost_mapSymbol(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Localhost_mapText(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.8 Logic Namespace

Purpose: Evaluate CSPro logic expressions and invoke functions.

Function	Sync	Async	Web Compatible	Description
`eval`	✅	✅	⚠️ Proxy	Evaluate logic expression
`getSymbol`	✅	✅	⚠️ Proxy	Get symbol by name
`getSymbolMetadata`	✅	✅	⚠️ Proxy	Get symbol metadata
`getSymbolValue`	✅	✅	⚠️ Proxy	Get symbol value
`invoke`	✅	✅	⚠️ Proxy	Invoke logic function
`updateSymbolValue`	✅	✅	⚠️ Proxy	Update symbol value

Web Porting Strategy:

- **Always routes to WASM engine** - logic evaluation requires compiled engine
- Results returned as JSON strings
- Symbol access provides field values, computed values, and metadata

KMP Implementation:

```
 suspend fun evaluateLogic(expression: String): String {
    return engine.processAction("Logic.eval", """{"expression": "$expression"}""")
}

suspend fun invokeLogicFunction(name: String, args: String): String {
    return engine.invokeLogicFunction(name, args)
}
```

C++ Signature:

```
 Result Logic_eval(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Logic_getSymbol(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Logic_getSymbolMetadata(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Logic_getSymbolValue(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Logic_invoke(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Logic_updateSymbolValue(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.9 Message Namespace

Purpose: Message formatting and localization.

Function	Sync	Async	Web Compatible	Description
`formatText`	✅	✅	✅ Simple	Format text with arguments
`getText`	✅	✅	✅ Cached	Get localized message text

Web Porting Strategy:

- ✅ **Partially Compatible** - Simple string formatting in web mode
- Message cache pre-populated during application load
- Complex formatting falls back to WASM engine

Web Implementation:

```
formatText: (args) => {
   if (self._isWebMode) {
       let text = args?.text || '';
       const formatArgs = args?.args || [];
       if (Array.isArray(formatArgs)) {
           formatArgs.forEach((arg, i) => {
               text = text.replace(`%${i+1}`, String(arg));
               text = text.replace('%s', String(arg));
               text = text.replace('%d', String(arg));
           });
       }
       return text;
   }
   // Native implementation...
}
```

C++ Signature:

```
Result Message_formatText(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Message_getText(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.10 Path Namespace

Purpose: File path operations and dialogs.

Function	Sync	Async	Web Compatible	Description
`createDirectory`	✅	✅	⚠️ Proxy	Create directory
`getDirectoryListing`	✅	✅	⚠️ Proxy	List directory contents
`getPathInfo`	✅	✅	✅ Simple	Get path information
`getSpecialPaths`	✅	✅	❌ Empty	Get special folder paths
`selectFile`	✅	✅	⚠️ Proxy	Select file dialog
`showFileDialog`	✅	✅	⚠️ Proxy	Show file dialog

Web Porting Strategy:

- `getPathInfo` : Simple parsing in JavaScript
- `getSpecialPaths` : Returns empty (no desktop/documents on web)
- File dialogs: Use `<input type="file">` element
- Directory operations: OPFS API

Web Implementation:

```
getPathInfo: (args) => {
   if (self._isWebMode) {
       const path = args?.path || '';
       const parts = path.split(/[/\\]/);
       const name = parts.pop() || '';
       const ext = name.includes('.') ? name.split('.').pop() : '';
       return { path, name, extension: ext, type: 'file', exists: false };
   }
   // Native implementation...
}
```

C++ Signature:

```
 Result Path_createDirectory(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Path_getDirectoryListing(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Path_getPathInfo(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Path_getSpecialPaths(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Path_selectFile(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Path_showFileDialog(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.11 Settings Namespace

Purpose: Persistent settings storage.

Function	Sync	Async	Web Compatible	Description
`getValue`	✅	✅	✅ Native	Get setting value
`putValue`	✅	✅	✅ Native	Save setting value

Web Porting Strategy:

- ✅ **Fully Compatible** - Uses `localStorage` API

- Settings prefixed with `cspro_setting_` to avoid conflicts

Web Implementation:

```
getValue: (args) => {
    if (self._isWebMode) {
        const key = args?.key || args;
        if (typeof key === 'string') {
            try {
                return localStorage.getItem(`cspro_setting_${key}`);
            } catch (e) { return null; }
        }
        return null;
    }
    // Native implementation...
}
```

C++ Signature:

```
Result Settings_getValue(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Settings_putValue(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.12 Sqlite Namespace

Purpose: SQLite database operations.

Function	Sync	Async	Web Compatible	Description
`close`	✅	✅	⚠️ Proxy	Close database connection
`exec`	✅	✅	⚠️ Proxy	Execute SQL statement
`open`	✅	✅	⚠️ Proxy	Open database connection
`rekey`	✅	✅	⚠️ Proxy	Change database encryption key

Web Porting Strategy:

- **sql.js (SQLite WASM)** used for web SQLite support
- Databases persisted to OPFS
- Sync API simulated with blocking WASM calls
- Async API preferred for web usage

KMP Implementation:

```
// SQLite managed by WASM engine with OPFS persistence
suspend fun sqliteExec(db: String, sql: String): String {
    return engine.processAction("Sqlite.exec", """{"db": "$db", "sql": "$sql"}""")
}
```

C++ Signature:

```
 Result Sqlite_close(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Sqlite_exec(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Sqlite_open(const JsonNode<wchar_t>& json_node, Caller& caller);
Result Sqlite_rekey(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.13 System Namespace

Purpose: System information and operations.

Function	Sync	Async	Web Compatible	Description
`getSharableUri`	✅	✅	❌ N/A	Get sharable file URI
`selectDocument`	✅	✅	⚠️ File Input	Select document
`getOS`	✅	-	✅ Native	Get operating system
`getDeviceId`	✅	-	✅ Native	Get device identifier
`exec`	✅	✅	❌ N/A	Execute system command

Web Porting Strategy:

- `getOS` : Parse `navigator.userAgent`
- `getDeviceId` : Generate and persist UUID in localStorage
- `selectDocument` : Use file input element
- `getSharableUri` : Not applicable (no native sharing)
- `exec` : Not supported (security sandbox)

Web Implementation:

```
 getOS: () => {
    if (self._isWebMode) {
        const ua = navigator.userAgent;
        if (ua.indexOf("Win") !== -1) return "Windows";
        if (ua.indexOf("Mac") !== -1) return "MacOS";
        if (ua.indexOf("Linux") !== -1) return "Linux";
        if (ua.indexOf("Android") !== -1) return "Android";
        if (ua.indexOf("like Mac") !== -1) return "iOS";
        return "Web";
    }
    return "Native";
},
getDeviceId: () => {
    if (self._isWebMode) {
        let id = localStorage.getItem('cspro_device_id');
        if (!id) {
            id = crypto.randomUUID();
            localStorage.setItem('cspro_device_id', id);
        }
        return id;
    }
    return "NativeDevice";
}
```

C++ Signature:

```
Result System_getSharableUri(const JsonNode<wchar_t>& json_node, Caller& caller);
Result System_selectDocument(const JsonNode<wchar_t>& json_node, Caller& caller);
```

2.14 UI Namespace

Purpose: User interface dialogs and display.

Function	Sync	Async	Web Compatible	Description
`alert`	✅	✅	✅ Native	Show alert message
`closeDialog`	✅	✅	✅ Native	Close current dialog
`enumerateWebViews`	✅	✅	❌ Empty	List web views
`getDisplayOptions`	✅	✅	✅ Cached	Get display options
`getInputData`	✅	✅	✅ Cached	Get dialog input data
`getMaxDisplayDimensions`	✅	✅	✅ Native	Get max display size
`postWebMessage`	✅	✅	✅ Native	Post message to parent
`setDisplayOptions`	✅	✅	✅ Native	Set display options
`showDialog`	✅	✅	⚠️ Proxy	Show HTML dialog
`view`	✅	✅	✅ Native	Open URL in new window

Web Porting Strategy:

- `alert` : Uses native `window.alert()`
- `closeDialog` : Posts message to parent frame
- `getMaxDisplayDimensions` : Uses `window.innerWidth/innerHeight`
- `showDialog` : Uses iframe-based dialog system
- `view` : Uses `window.open()`

Web Implementation:

```
 alert: (args) => {
    if (self._isWebMode) {
        const message = typeof args === 'string' ? args : (args?.message || '');
        const title = args?.title || '';
        if (title) {
            alert(`${title}\n\n${message}`);
        } else {
            alert(message);
        }
        return null;
    }
    // Native implementation...
},
closeDialog: (args) => {
    if (self._isWebMode) {
        window.parent.postMessage({
            type: 'cspro-dialog-close',
            result: args
        }, '*');
        return null;
    }
    // Native implementation...
},
getMaxDisplayDimensions: (args) => {
    if (self._isWebMode) {
        return { width: window.innerWidth || 800, height: window.innerHeight || 600 };
    }
    // Native implementation...
}
```

C++ Signature:

```
 Result UI_alert(const JsonNode<wchar_t>& json_node, Caller& caller);
Result UI_closeDialog(const JsonNode<wchar_t>& json_node, Caller& caller);
Result UI_enumerateWebViews(const JsonNode<wchar_t>& json_node, Caller& caller);
Result UI_getDisplayOptions(const JsonNode<wchar_t>& json_node, Caller& caller);
Result UI_getInputData(const JsonNode<wchar_t>& json_node, Caller& caller);
Result UI_getMaxDisplayDimensions(const JsonNode<wchar_t>& json_node, Caller& caller);
Result UI_postWebMessage(const JsonNode<wchar_t>& json_node, Caller& caller);
Result UI_setDisplayOptions(const JsonNode<wchar_t>& json_node, Caller& caller);
Result UI_showDialog(const JsonNode<wchar_t>& json_node, Caller& caller);
Result UI_view(const JsonNode<wchar_t>& json_node, Caller& caller);
```

---

3. CSPro Logic Functions

CSPro includes 450+ logic functions organized by category. These are executed by the WASM engine.

3.1 Core Commands

Code	Function	Description	Web Status
0	`CONST_CODE`	Constant value	✅
1	`SVAR_CODE`	String variable	✅
2	`MVAR_CODE`	Numeric variable	✅
3	`CPT_CODE / UF_CODE`	Computed/User function	✅
4-10	Math operators	ADD, SUB, MULT, DIV, MOD, MINUS, EXP	✅
11-13	Logic operators	OR, AND, NOT	✅
14-19	Comparison	EQ, NE, LE, LT, GE, GT	✅
20	`EQU_CODE`	Equation assignment	✅
23	`IF_CODE`	If statement	✅
24	`WHILE_CODE`	While loop	✅
37	`USERFUNCTIONCALL_CODE`	User function call	✅

3.2 Data Entry Commands

Code	Function	Description	Web Status
40	`SKIPTO_CODE`	Skip to field	✅
41	`ADVANCE_CODE`	Advance cursor	✅
42	`REENTER_CODE`	Reenter field	✅
43	`NOINPUT_CODE`	Skip input	✅
44	`ENDSECT_CODE`	End section	✅
45	`ENDLEVL_CODE`	End level	✅
46	`ENTER_CODE`	Enter field	✅
153	`MOVETO_CODE`	Move to field	✅
257	`ASK_CODE`	Ask for input	✅

3.3 Batch Commands

Code	Function	Description	Web Status
47	`SKIPCASE_CODE`	Skip current case	✅
49	`STOP_CODE`	Stop processing	✅
53	`BREAK_CODE`	Break from loop	✅
54	`EXPORT_CODE`	Export data	⚠️

3.4 Numeric Functions

Code	Function	Description	Web Status
58	`FNSQRT_CODE`	Square root	✅
59	`FNEXP_CODE`	Exponential	✅
60	`FNINT_CODE`	Integer part	✅
61	`FNLOG_CODE`	Natural logarithm	✅
62	`FNSEED_CODE`	Random seed	✅
63	`FNRANDOM_CODE`	Random number	✅
64	`FNNOCCURS_CODE`	Number of occurrences	✅
67	`FNCOUNT_CODE`	Count	✅
68	`FNSUM_CODE`	Sum	✅
69	`FNAVERAGE_CODE`	Average	✅
70	`FNMIN_CODE`	Minimum	✅
71	`FNMAX_CODE`	Maximum	✅
127	`FNMINVALUE_CODE`	Minimum value	✅
128	`FNMAXVALUE_CODE`	Maximum value	✅
202	`FNABS_CODE`	Absolute value	✅
203	`FNRANDOMIN_CODE`	Random in range	✅
233	`FNROUND_CODE`	Round	✅

3.5 Alpha/String Functions

Code	Function	Description	Web Status
74	`FNCONCAT_CODE`	Concatenate strings	✅
75	`FNTONUMB_CODE`	String to number	✅
76	`FNPOS_CODE`	Find position	✅
77	`FNCOMPARE_CODE`	Compare strings	✅
78	`FNLENGTH_CODE`	String length	✅
79	`FNSTRIP_CODE`	Strip whitespace	✅
80	`FNPOSCHAR_CODE`	Find character position	✅
81	`FNEDIT_CODE`	Edit/format string	✅
174	`FNTOLOWER_CODE`	To lowercase	✅
175	`FNTOUPPER_CODE`	To uppercase	✅
281	`FNREGEXMATCH_CODE`	Regex match	✅
350	`FNSTARTSWITH_CODE`	Starts with	✅
380	`FNREPLACE_CODE`	Replace string	✅
454	`FNCOMPARENOCASE_CODE`	Compare case-insensitive	✅

3.6 Date Functions

Code	Function	Description	Web Status
90	`FNSYSTIME_CODE`	System time	✅
91	`FNSYSDATE_CODE`	System date	✅
179	`FNDATEDIFF_CODE`	Date difference	✅
211	`FNDATEADD_CODE`	Add to date	✅
212	`FNDATEVALID_CODE`	Validate date	✅
252	`FNTIMESTAMP_CODE`	Unix timestamp	✅
272	`FNTIMESTRING_CODE`	Format time string	✅

3.7 File I/O Functions

Code	Function	Description	Web Status
162	`FNFILE_CREATE_CODE`	Create file	⚠️ OPFS
163	`FNFILE_EXIST_CODE`	Check file exists	⚠️ OPFS
164	`FNFILE_DELETE_CODE`	Delete file	⚠️ OPFS
165	`FNFILE_COPY_CODE`	Copy file	⚠️ OPFS
166	`FNFILE_RENAME_CODE`	Rename file	⚠️ OPFS
167	`FNFILE_SIZE_CODE`	Get file size	⚠️ OPFS
168	`FNFILE_CONCAT_CODE`	Concatenate files	⚠️ OPFS
169	`FNFILE_READ_CODE`	Read file	⚠️ OPFS
170	`FNFILE_WRITE_CODE`	Write file	⚠️ OPFS
206	`FNFILE_EMPTY_CODE`	Check if empty	⚠️ OPFS
224	`FNDIREXIST_CODE`	Directory exists	⚠️ OPFS
225	`FNDIRCREATE_CODE`	Create directory	⚠️ OPFS
228	`FNDIRLIST_CODE`	List directory	⚠️ OPFS
276	`FNDIRDELETE_CODE`	Delete directory	⚠️ OPFS
337	`FNFILETIME_CODE`	File timestamp	⚠️ OPFS

3.8 Case/Data Functions

Code	Function	Description	Web Status
95	`FNCLRCASE_CODE`	Clear case	✅
105	`FNLOADCASE_CODE`	Load case	✅
107	`FNRETRIEVE_CODE`	Retrieve data	✅
109	`FNWRITECASE_CODE`	Write case	✅
110	`FNDELCASE_CODE`	Delete case	✅
111	`FNFIND_CODE`	Find in data	✅
112	`FNKEY_CODE`	Get key	✅
113	`FNOPEN_CODE`	Open data file	✅
114	`FNCLOSE_CODE`	Close data file	✅
115	`FNLOCATE_CODE`	Locate record	✅
144	`FNINSERT_CODE`	Insert occurrence	✅
145	`FNDELETE_CODE`	Delete occurrence	✅
146	`FNSORT_CODE`	Sort occurrences	✅
184	`FNENDCASE_CODE`	End current case	✅
235	`FNSAVEPARTIAL_CODE`	Save partial case	✅
243-244	Case labels	Get/Set case label	✅
339	`FORCASE_CODE`	For each case	✅
340	`FNSELCASE_CODE`	Select case	✅
341	`FNCOUNTCASES_CODE`	Count cases	✅
342	`FNKEYLIST_CODE`	Get key list	✅
395	`FNCURRENTKEY_CODE`	Current key	✅

3.9 List Functions

Code	Function	Description	Web Status
227	`LISTVAR_CODE`	List variable	✅
306	`LISTFN_ADD_CODE`	Add to list	✅
307	`LISTFN_CLEAR_CODE`	Clear list	✅
308	`LISTFN_INSERT_CODE`	Insert in list	✅
309	`LISTFN_LENGTH_CODE`	List length	✅
310	`LISTFN_REMOVE_CODE`	Remove from list	✅
311	`LISTFN_SEEK_CODE`	Find in list	✅
312	`LISTFN_SHOW_CODE`	Show list	✅
362	`LISTFN_SORT_CODE`	Sort list	✅
363	`LISTFN_REMOVEDUPLICATES_CODE`	Remove duplicates	✅
364	`LISTFN_REMOVEIN_CODE`	Remove items in	✅

3.10 Map Functions

Code	Function	Description	Web Status
286	`MAPFN_SHOW_CODE`	Show map	⚠️ Limited
287	`MAPFN_HIDE_CODE`	Hide map	⚠️ Limited
288	`MAPFN_ADD_MARKER_CODE`	Add marker	⚠️ Limited
289-305	Marker functions	Various marker operations	⚠️ Limited
320-324	Map geometry	Markers, buttons, coordinates	⚠️ Limited
411-427	Geometry functions	Load, save, trace, area	⚠️ Limited
429	`MAPFN_SAVESNAPSHOT_CODE`	Save map snapshot	❌ N/A
450	`MAPFN_CLEAR_CODE`	Clear map	⚠️ Limited

Web Porting Strategy for Maps:

- Use Leaflet.js or Google Maps JavaScript API
- Geometry stored as GeoJSON
- Tracing requires touch/mouse event handling
- GPS requires `navigator.geolocation`

3.11 Audio/Media Functions

Code	Function	Description	Web Status
232	`FNGETIMAGE_CODE`	Get image	⚠️ File Input
352-360	Audio functions	Clear, concat, load, play, save, stop, record	⚠️ MediaRecorder
396-405	Image functions	Compute, capture, load, resample, save, view	⚠️ Canvas

Web Porting Strategy for Media:

- Audio: Use Web Audio API and MediaRecorder
- Images: Use Canvas API for manipulation
- Camera: Use `getUserMedia` API
- Signatures: Canvas-based capture

3.12 Sync Functions

Code	Function	Description	Web Status
236	`FNSYNC_CONNECT_CODE`	Connect to server	✅ Fetch
237	`FNSYNC_DISCONNECT_CODE`	Disconnect	✅
238	`FNSYNC_DATA_CODE`	Sync data	✅ Fetch
239	`FNSYNC_FILE_CODE`	Sync file	✅ Fetch
240	`FNSYNC_SERVER_CODE`	Server sync	✅ Fetch
336	`FNSYNC_APP_CODE`	Sync application	⚠️ Limited
345	`FNSYNC_MESSAGE_CODE`	Sync message	✅ Fetch
370	`FNSYNC_PARADATA_CODE`	Sync paradata	✅ Fetch
430	`FNSYNC_TIME_CODE`	Get server time	✅ Fetch

Web Porting Strategy:

- All sync operations use `fetch()` API
- CORS headers required on server
- Authentication via headers
- Chunked upload for large files

3.13 UI Functions

Code	Function	Description	Web Status
72	`FNDISPLAY_CODE`	Display message	✅
73	`FNERRMSG_CODE`	Error message	✅
94	`FNACCEPT_CODE`	Accept input	✅
173	`FNSHOWLIST_CODE`	Show list dialog	✅
185	`FNUSERBAR_CODE`	User toolbar	⚠️ Limited
231	`FNPROMPT_CODE`	Prompt dialog	✅
269	`FNSHOW_CODE`	Show dialog	✅
270	`FNSHOWARRAY_CODE`	Show array dialog	✅
326	`FNVIEW_CODE`	View URL/file	✅
431	`FNHTMLDIALOG_CODE`	HTML dialog	✅
455	`CASEFN_VIEW_CODE`	View case	✅

---

4. Web Porting Strategies

4.1 Compatibility Classification

Symbol	Meaning	Strategy
✅	Fully Compatible	Native JavaScript/Web API implementation
⚠️	Partially Compatible	Modified behavior or polyfill required
❌	Not Available	Feature not possible in browser sandbox

4.2 File System Strategy (OPFS)

The Origin Private File System (OPFS) provides persistent file storage in browsers:

```
 // KMP OPFS Implementation
object OPFSFileSystem {
    suspend fun readFile(path: String): String
    suspend fun writeFile(path: String, content: String)
    suspend fun deleteFile(path: String): Boolean
    suspend fun listDirectory(path: String): List<String>
    fun resolveVirtualPath(path: String): String
}
```

Path Mapping:

- Android: `/storage/emulated/0/CSEntry/` → OPFS: `/data/`
- Windows: `C:\CSEntry\` → OPFS: `/data/`

4.3 Dialog System Strategy

Dialogs are implemented using iframe-based overlays:

```
// KMP Dialog Implementation
class CSProDialogManager {
    fun showDialog(url: String, inputData: Any?, options: DialogOptions): DialogResult
    fun closeDialog(result: Any?)
    fun processActionRequest(request: ActionRequest): ActionResponse
}
```

Message Protocol:

```
// Parent → Dialog
{ type: 'cspro-dialog-init', inputData: {...}, accessToken: '...' }

// Dialog → Parent
{ type: 'cspro-dialog-close', result: {...} }
{ type: 'cspro-action-request', method: '...', args: {...} }

// Parent → Dialog (response)
{ type: 'cspro-action-response', requestId: 1, result: {...} }
```

4.4 Native API Alternatives

Native Feature	Web Alternative
File system	OPFS, IndexedDB
SQLite	sql.js (WASM)
GPS	`navigator.geolocation`
Camera	`getUserMedia`
Bluetooth	Web Bluetooth API
Clipboard	`navigator.clipboard`
System exec	Not available
Native dialogs	Custom HTML dialogs

4.5 Security Considerations

- **CORS**: Server must allow cross-origin requests
- **CSP**: Content Security Policy must allow WASM, iframes
- **HTTPS**: Required for clipboard, geolocation, camera
- **Storage**: OPFS requires secure context

---

5. KMP Implementation Status

5.1 Core Components

Component	Status	Notes
ActionInvoker.kt	✅	Fully ported
WasmEngineInterface.kt	✅	All methods implemented
CSProWasmModule.kt	✅	Embind declarations complete
EngineFunctionDispatcher.kt	✅	UI function routing
CSProDialogManager.kt	✅	Dialog management
OPFSFileSystem.kt	✅	File system bridge

5.2 Namespace Implementation

Namespace	JS API	KMP Bridge	WASM Engine
Application	✅	✅	✅
Clipboard	✅	✅	✅
Data	✅	✅	✅
Dictionary	✅	✅	✅
File	✅	✅	✅
Hash	✅	✅	✅
Localhost	✅	✅	✅
Logic	✅	✅	✅
Message	✅	✅	✅
Path	✅	✅	✅
Settings	✅	✅	✅
Sqlite	✅	✅	✅
System	✅	⚠️	✅
UI	✅	✅	✅

5.3 Function Categories

Category	Total	Web Compatible	Partial	Not Available
Core Commands	40	40	0	0
Data Entry	15	15	0	0
Batch	10	8	2	0
Numeric	30	30	0	0
String	25	25	0	0
Date	15	15	0	0
File I/O	20	0	20	0
Case/Data	35	35	0	0
List	15	15	0	0
Map	40	0	30	10
Audio/Media	20	0	15	5
Sync	12	10	2	0
UI	25	20	5	0
Total	**302**	**213 (71%)**	**74 (24%)**	**15 (5%)**

---

6. Architecture Reference

6.1 Call Flow Diagram


Browser Environment
HTML Dialog (iframe)
Main UI (index.html)
Kotlin/JS Application
postMessage
action-invoker.js
Applica-tion
Clip-board
Data
Dictio-nary
File
_webProxyAction() / Native API
WasmEngineInterface.kt
processAction(namespace: String, args: String): String
getSymbolValue(name: String): String
invokeLogicFunction(name: String, args: String): String
WASM Module
CSProWasmModule.kt
external fun ProcessAction(action: String, args: String): String
external fun InvokeLogicFunction(name: String): String
Embind
WASMBindings.cpp
EMSCRIPTEN_BINDINGS(cspro_engine) {
function("ProcessAction", &processAction);
function("InvokeLogicFunction", &invokeLogicFunction);
}
ActionInvoker.h (C++ Runtime)
Result Application_getFormFile(...);
Result Clipboard_getText(...);
Result Logic_eval(...);
Result UI_showDialog(...);

6.2 File Locations

```
 CSEntryWeb_KMP/
├── src/wasmJsMain/kotlin/gov/census/cspro/
│   ├── engine/
│   │   ├── ActionInvoker.kt              # ActionInvoker bridge
│   │   ├── WasmEngineInterface.kt        # WASM engine interface
│   │   ├── CSProEngineService.kt         # Engine service
│   │   ├── CSProWasmModule.kt            # Embind declarations
│   │   └── EngineFunctionDispatcher.kt   # UI function routing
│   ├── dialogs/
│   │   ├── CSProDialogManager.kt         # Dialog management
│   │   └── DialogBridge.kt               # Dialog communication
│   └── storage/
│       └── OPFSFileSystem.kt             # OPFS file system
├── public/
│   ├── action-invoker.js                 # JavaScript ActionInvoker API
│   └── kotlin/                           # Compiled Kotlin/JS
└── wasm-engine/src/
    └── zAction/
        └── ActionInvoker.h               # C++ ActionInvoker Runtime
```

Appendix A: Action Message Codes

Code	Action
26802	Application.getFormFile
46403	Application.getQuestionnaireContent
25339	Application.getQuestionText
30785	Clipboard.getText
14476	Clipboard.putText
49575	Data.getCase
41376	Dictionary.getDictionary
28073	File.copy
43116	File.readBytes
49883	File.readLines
30620	File.readText
63076	File.writeBytes
52284	File.writeLines
41003	File.writeText
46119	Hash.createHash
42203	Hash.createMd5
22437	Localhost.mapActionResult
31960	Message.formatText
449	Message.getText
20380	Path.createDirectory
36724	Path.getDirectoryListing
59076	Path.getPathInfo
41709	Path.getSpecialPaths
62012	Path.selectFile
35645	Path.showFileDialog
64779	Settings.getValue
17067	Settings.putValue
36421	Sqlite.close
31287	Sqlite.exec
55316	Sqlite.open

Code	Action
40839	Sqlite.rekey
49116	System.getSharableUri
14855	System.selectDocument
48073	UI.alert
60265	UI.closeDialog
50927	UI.enumerateWebViews
63831	UI.getDisplayOptions
57200	UI.getInputData
22001	UI.getMaxDisplayDimensions
39555	UI.postWebMessage
62732	UI.setDisplayOptions
41655	UI.showDialog
27633	UI.view

Appendix B: Logic Namespace Enum

```
namespace Logic {
   enum class FunctionNamespace : int {
       Barcode,
       Path,
       CS,
       CS_Application,
       CS_Clipboard,
       CS_Data,
       CS_Dictionary,
       CS_File,
       CS_Hash,
       CS_Localhost,
       CS_Logic,
       CS_Message,
       CS_Path,
       CS_Settings,
       CS_Sqlite,
       CS_System,
       CS_UI,
   };
}
```

Document generated for CSPro WASM/KMP Web Application