SAMVEA | DESIGN STUDIO

Developer guide

Architecture, data formats and extension reference



Connected design.
Controlled project information.

Engineering models / Structured data / Document review

2026-10-24

SDS-DEV | Complete guide

DOCUMENTATION / WORKFLOWS AND REFERENCE

Contents

Complete developer guide | A4 portrait

Section	Topic	Page
Development	SDS Developer Guide	3

Page numbers include the cover and contents. Interactive examples are captured as they appear when the PDF is created.

SDS Developer Guide

This independent handbook describes the current SDS source layout, data contracts and extension points. It is intended for developers working on the native application, engineering data and integrated tools. Press **F2** or choose **Help > SDS Developer Guide** to reopen it. **F1** continues to open contextual user help in a separate tab.

The guide is one local HTML document. Its search, navigation and PDF export belong to the developer guide, independently of the user guide. Use the summary links below to jump to a section.

Summary

Section	Contents
1. Architecture and source map	Native shell, modules and ownership.
2. Build and run	VC2008 x64 builds, staging and test profiles.
3. TJS format and macros	Typed cells, stable sheet IDs and a complete example.
4. Combined tables and piping	Record identity, writeback and model refresh.
5. Vector and equipment extensions	Read-only drawing generators and equipment recipes.
6. DJS, DXF and PDF	Document formats, layout and publication paths.
7. Project control and services	Persistent IDs, libgit2 and service authorization.
8. Extending native HTML help	Module ABI, local HTML and the MuJS DOM profile.
9. Validation and troubleshooting	Focused tests, diagnostics and safe deployment.

1. Architecture and source map

The desktop host is a Windows MFC MDI application. The shell owns the ribbon, workspace tree, document tabs and selection routing. Modules own their document data and renderers. A visible row, window handle or model label is not a persistent identity.

Source area	Responsibility
PhoneServiceMdi.cpp / .rc / resource.h	Application lifecycle, command handlers, menus and resource IDs.
SdsRibbonBar.h	Ribbon groups and command reservations.
SdsTables.h / tables/	TJS model, grid adapter, property projections and piping editing.
libraries/equipment/	Versioned definitions, TJS bindings and geometry recipes.
documents/	DJS import, document layout, text editing and PDF publication.
project-control/	Project registry, native Git integration and controlled operations.

review-portal/ / webio/ / peer/	Review service, embedded HTTP integration and synchronization components.
documentation/	Independent user/developer help content and native HTML renderer.

Start a new command at the module that owns the data. Connect a unique command ID to its native handler, then to the menu or ribbon. Keep view refresh and selection notifications separate from file writes. A new extension is not available merely because its ribbon button exists.

Back to summary

2. Build and run

The qualified desktop target uses **Visual C++ 2008 SP1, x64, C++03 and the static runtime**. Match the architecture and runtime of linked libraries. Do not load an x64 module into the legacy Win32 executable.

x64/environment.ps1 defines the compiler, SDK, library roots and temporary directories for this workstation. **x64/build-shell-candidate.ps1** compiles the main shell; **x64/link-shell.ps1** links it against the existing qualified modules. These scripts depend on the configured toolchain and prebuilt libraries: they are not a clean-checkout bootstrap.

From the source root, compile to an isolated directory with adequate space:

```
powershell -NoProfile -ExecutionPolicy Bypass -File
  x64/build-shell-candidate.ps1
  -OutputDirectory T:\SDS-Developer-Test -CompileOnly
```

The line breaks above are for readability; enter the command on one line. Link to a candidate directory with **-ObjectDirectory**, **-LinkDirectory**, **-NoMap** and **-SkipPublish** on x64/link-shell.ps1. Do not overwrite an executable that is running.

Help content is editable without recompilation. Renderer changes require the documentation DLL build:

```
powershell -NoProfile -ExecutionPolicy Bypass -File
  documentation/build.ps1 -OutputDirectory T:\SDS-Help-Test
```

Install with **documentation/install.ps1 -ApplicationDirectory <runtime> -BuildDirectory <help-build>**. It stages both guides, their common banner/splash assets and the DOM runtime. The script checks the deployed DLL hash.

Launch SDS with **--documentation** to start in user help, then press F2. Use a private **SDS_PROFILE_PATH** and **SDS_PEER_CONFIG** for automated GUI tests. Preserve drive-letter paths where the native configuration requires them. Keep test outputs and large compiler intermediates on a drive with sufficient free space.

Back to summary

3. TJS format and macros

A TJS file is UTF-8 JSON with **format: sds.tables** and **version: 1**. A worksheet has a stable **id**, a display **name** and a **cells** map. A cell is a string, finite number, Boolean or null; cell addresses use A1 notation. The current core supports 64 worksheets and addresses A1 through BZ256. Use supported **layout** extensions for styles, dimensions, property

schemas or drawing declarations.

Save the following as **developer-example.tjs** and open it in SDS:

```
{
  "format": "sds.tables",
  "version": 1,
  "sheets": [{
    "id": "demo", "name": "Developer example",
    "cells": {"A1": "Length", "B1": 6.5}
  }],
  "macros": [{
    "name": "Double length",
    "code": "var t=sds.tables.get('demo');t.set('B2',Number(t.get('B1'))*2);"
  }]
}
```

Running **Double length** writes the number 13 to B2. Macros do not execute when a file opens. An explicit run uses a fresh, bounded MuJS state and stages writes against a draft workbook; an exception discards the changes. **sds.tables.get(id).get(cell)** reads a cell and **.set(cell,value)** updates it. Model access exists only when the host explicitly supplies a model context.

Do not assume imported Excel formulas are executable: preserved formula text and reference metadata are distinct from supported macro execution. Use stable sheet IDs in code, not translated tab names. Double-click a cell or use **Shift+F2** to edit it; plain F2 opens this developer guide.

Core entry points: **SdsTables.h**, **tables/TableView.cpp** and **tables/Tests.cpp**. When extending the schema, update serialization and round-trip tests together. Unknown top-level members are not automatically retained by every save path.

Back to summary

4. Combined tables and piping

A combined workbook is a temporary projection of persistent TJS records. It must not become a second authoritative copy. **FolderDescription** validates compatible table shapes and declared **layout.propertyTable** fields. Each field declares its key, label, unit, valueCell, type and nullable flag. A property table also declares version, family and schemaVersion.

FolderWorkbook records source-file identity and hashes. Each projected cell has a **FolderOrigin** mapping to its source record, sheet and address. Changing sort order or selecting a different row must not change that mapping. Headings, source filenames and unbound cells are protected.

FolderWriteback validates typed values, obtains the existing edit locks, checks source hashes and updates the original JSON. It patches cell values while preserving source metadata. A stale source or changed membership must produce a conflict instead of overwriting another edit.

```
Piping/
  .sds-piping-database.json
  INPUT_BASIC_ELEMENT_DATA/
    10.tjs
```

```
    20.tjs
  other-table-directory/
    record.tjs
```

The database manifest has **format: sds.piping.database**, **version: 1**, **tables** and **basicBindings**. **PipingDatabaseWorkbook** creates one temporary worksheet per physical table, including empty tables. Validate uniformity within each table; different tables can have different columns.

Connectivity uses persistent keys defined by the manifest and record fields, not spreadsheet row numbers or process memory addresses. Basic element fields map through basicBindings; other columns use keys of the form **mdb.TABLE.COLUMN**. Numeric element filenames represent their FROM node. Changing that node requires the renumbering workflow, not an arbitrary filename edit.

Use the existing native piping edit and refresh paths to regenerate geometry and synchronize the grid, inspector and 3D selection. An interrupted multi-file edit can leave **.sds-piping-pending.json**; recover the operation before further writes. Source references: **tables/PipingDatabase.h**, **PipingDatabaseWorkbook.h**, **PipingEdit.h**, **FolderWorkbook.h** and **FolderWriteback.h**.

Back to summary

5. Vector and equipment extensions

A sheet can declare **layout.drawing2d** with format **sds.drawing2d**, version 1, a generator macro name and a mapping from sheet IDs to drawing layer names. **tables/Drawing2d.h** executes the generator in a separate bounded interpreter with read-only cell access. This is not the ordinary cell-writing macro host.

Inside that drawing generator, the supported helpers include line, circle, rect, text and dimension:

```
var length=Number(sds.tables.get('demo').get('B1'));
sds.draw.rect('body','motor',0,0,80,30);
sds.draw.dimension('length','motor',0,-15,80,-15,
                   'Length = '+length+' m');
```

Declare the **motor** layer before using this fragment. Entity IDs must be unique and coordinates finite. A validated scene drives the small vector preview and its DXF export. Keep illustrative parameter diagrams separate from engineering-scale drawing and printed datasheet layout.

Equipment packages declare identity, capabilities, typed parameters and TJS bindings in **definition.json**; **parameters.tjs** stores the recipe. Equipment geometry macros run with the equipment host's **p** and **cad** APIs. They cannot be pasted unchanged into an ordinary TJS macro.

Preserve the pinned definition/version of an existing instance. Library discovery must not execute geometry scripts. Review changes to parameter defaults, bindings and recipes before updating instances. Examples live in **libraries/equipment/**; their current adapters and supported capabilities are defined in the source manifests.

Back to summary

6. DJS, DXF and PDF

DJS is UTF-8 JSON with **format: sds.document** and version 1. It stores styles, embedded assets and ordered sections containing paragraphs, tables and page breaks. Page dimensions and margins are in points (72 per inch). Stable section, block and text identities support editing and version comparison.

documents/DJS-FORMAT.txt defines the contract. **djs_import.py** normalizes DOCX or extracted JSON; **document.js** performs SDS layout; **DjsTextStore.cs** validates text patches. The DJS host uses WebView2. Replacing HTML help with wxHTML does not replace this separate document renderer or promise exact Microsoft Word pagination.

For DXF, preserve model-space, paper-space and layout ownership. Do not reuse the PCB scene layout for general CAD drawings. Publication dispatch must use the active document's source association: a PDF preview must not accidentally regenerate a background TJS or DXF.

Publication path	Implementation reference
TJS datasheet	tables/DatasheetPdf.h and the workbook's supported print/layout data.
DXF / DJS	SdsDocumentPdf.h / .inl dispatch and each format's renderer.
Native HTML help	documentation/DocumentationPdf.h, wxHtmlDCRenderer and Microsoft Print to PDF.

Help PDF preview/export uses the same corporate splash-screen cover and SAMVEA banner as the documentation set. The developer PDF has its own title and contents. The PDF renderer captures text and supported images from the current help DOM; interactive links become a static snapshot. Windows Microsoft Print to PDF must be available for this path.

Back to summary

7. Project control and services

The project registry stores **.sds/project.json**, **.sds/assets/<assetId>.json** and **.sds/documents/<documentId>.json**. Keep permanent UUID identity separate from a relative filename, display number, metadataRevision and review-service revision.

Native project control uses integrated **libgit2**, not git.exe. Existing workflows cover checkpoints, branches, merge previews and recovery. Semantic TJS/DJS merge uses supported stable identities; no general automatic CAD geometry or binary merge is implied. Use the existing service layer instead of writing a Git index or registry record directly.

Validate paths, supported schema versions and references before mutation. Preserve expected-hash checks, staged writes, journals and conflict handling. Never bypass a pending move or merge journal. **project-control/README.txt** and its adjacent schemas document the implemented batches and their limits.

Webio, review and peer integration are separate components with their own authentication and authorization boundaries. A disabled menu is not an access check: operations that modify files, publish documents or administer users must enforce the current account's rights in the owning service. Do not copy user credentials, private keys or private account data into portable project metadata or help examples.

Back to summary

8. Extending native HTML help

SdsDocumentationView.h is the MFC adapter. It dynamically loads **SdsDocumentation.dll**, validates C ABI version 1 and keeps one MDI frame per guide. The ABI exchanges HWNDs, wide strings and opaque instance handles; do not pass MFC/wx C++ objects or ownership across it.

Documentation.cpp embeds wxHtmlWindow and its Windows GDI renderer. **DomRuntime.h** hosts the qualified MuJS runtime; **runtime/dom.js** implements the bounded SDS DOM profile. The DOM supplies serialized HTML to wxHTML. This is a supported subset, not a modern browser or a claim of full DOM conformance.

The shell forwards message preprocessing and idle work to the module. Create, resize, navigate, pump and destroy on the SDS UI thread. Keep initialization out of DllMain. Closing one guide must not destroy the other guide's DOM or HWNDs.

```
documentation/
  SdsDocumentation.dll
  dom.js
  content/                  user guide
  developer/
    guide.txt               developer identity
    dom.js
    content/
      contents.tsv
      index.html            this handbook
      samvea.png
      splash.png
```

Keep developer content separate from the user contents.tsv. Use UTF-8 headings, paragraphs, lists, tables, local images and relative anchors. Retain the common SAMVEA banner. F5 reloads the current HTML; reopening the guide rebuilds its contents/search index. **stage-content.ps1** copies the shared logo, approved splash and DOM runtime into both packages without maintaining separate source images.

Supported optional scripts can use documented text-node updates and click handlers. External scripts, fetch/network APIs, modern CSS frameworks and arbitrary browser APIs are not available. Use trusted local help; interpreter quotas do not make arbitrary third-party pages an audited sandbox.

Back to summary

9. Validation and troubleshooting

Change	Verification
TJS parsing or macro behavior	tables/test-core.ps1; verify types, rollback and save/reopen.
Combined records or piping	FolderWorkbookTests.cpp, FolderWritebackTests.cpp and piping tests; verify persistent identity and conflicts.
DJS conversion or editing	documents/test_djs_import.py and text/PDF tests.
Help renderer or packaging	documentation/tests/renderer_smoke.py and independent-guide checks.

PDF or menu/shortcut integration	Native SDS GUI tests plus PDF text, page geometry and visual inspection.

Run tests on disposable records and private profiles. Check both a successful change and a relevant failure path. For a save change, verify persistence after reopen; for a rendering change, inspect the result, not only the return code. For this guide, verify F2 from the shell and help view, Help-menu activation, F1 returning to the user guide and independent close/reopen.

If the help module fails to load, check its expected path, architecture and ABI version. If images are missing, check local asset names and packaging. If PDF export fails, inspect the status message, Windows print service and writable cache. If a record changed externally, refresh or resolve its conflict before retrying the edit.

Keep a known working executable and module before deployment. Retain build logs and test evidence with the candidate. Deploy only the matching application/module/assets after the application has closed, then confirm the loaded build in the real SDS UI.

Back to summary