Devel Documentation

SaltOS 4.0 r1930

April 2025

Contents

1 Introduction

SaltOS4 is a modular, extensible framework for building Rich Internet Applications (RIAs). It is designed for rapid development of dynamic, offline-capable applications using a clean separation between backend and frontend.

It is composed of:

- A PHP backend (‘api/‘)
- A JavaScript frontend (‘web/‘)
- REST/JSON API with support for CLI execution
- Offline operation through a Service Worker proxy
- Declarative and dinamic app definitions via XML and/or YAML
- Shared templates and logic to reduce boilerplate
- Clean separation of frontend and backend
- Rapid development of apps with shared templates and logic

2 Backend Structure & API (api/)

The backend is under the ‘api/‘ folder and contains four folders (autoload, database, lib and actions) to organize the code depending their usage and functionality.

2.1 Autoloaded Modules (‘php/autoload/‘)

Core functions automatically loaded on each request:

- ‘autoload/apps.php‘: * Apps helper module
- ‘autoload/array.php‘: * Array helper module
- ‘autoload/compat.php‘: * Compatibility helper module
- ‘autoload/config.php‘: * Config helper module
- ‘autoload/database.php‘: * Database helper module
- ‘autoload/datetime.php‘: * Datetime helper module
- ‘autoload/error.php‘: * Error helper module
- ‘autoload/exec.php‘: * Execution helper module
- ‘autoload/file.php‘: * File utils helper module
- ‘autoload/getdata.php‘: * Get data helper module
- ‘autoload/gettext.php‘: * Gettext helper module
- ‘autoload/iniset.php‘: * Iniset helper module
- ‘autoload/json.php‘: * Json helper module
- ‘autoload/log.php‘: * Log helper module

- ‘autoload/memory.php‘: * Memory helper module
- ‘autoload/mime.php‘: * Mime helper module
- ‘autoload/output.php‘: * Output helper module
- ‘autoload/pcov.php‘: * PCOV helper module
- ‘autoload/perms.php‘: * Permissions helper module
- ‘autoload/random.php‘: * Random helper module
- ‘autoload/semaphores.php‘: * Semaphore helper module
- ‘autoload/server.php‘: * Server helper module
- ‘autoload/sql.php‘: * SQL utils helper module
- ‘autoload/strings.php‘: * String utils helper module
- ‘autoload/system.php‘: * System helper module
- ‘autoload/tokens.php‘: * Tokens helper module
- ‘autoload/user.php‘: * User helper module
- ‘autoload/version.php‘: * Version helper module
- ‘autoload/xml2array.php‘: * XML to Array helper module
- ‘autoload/yaml.php‘: * Yaml helper module
- ‘autoload/zindex.php‘: * Main execution module

2.2 Database Drivers (‘php/database/‘)

Supports:

- ‘database/libsqlite.php‘: * SQLite3 functions library
- ‘database/mysqli.php‘: * MySQL improved driver
- ‘database/pdo_mssql.php‘: * PDO MsSQL driver
- ‘database/pdo_mysql.php‘: * PDO MySQL driver
- ‘database/pdo_sqlite.php‘: * PDO SQLite driver
- ‘database/sqlite3.php‘: * SQLite3 driver

2.3 Libraries (‘php/lib/‘)

Not autoloaded; provide extra functionality:

- ‘lib/actions.php‘: * Actions module
- ‘lib/array2xml.php‘: * Array to XML helper module
- ‘lib/ascii.php‘: * Make Table ASCII
- ‘lib/auth.php‘: * Login functions
- ‘lib/barcode.php‘: * Barcode helper module

- ‘lib/browser.php‘: * Browser helper module
- ‘lib/captcha.php‘: * Captcha helper module
- ‘lib/color.php‘: * Color helper module
- ‘lib/control.php‘: * Control helper module
- ‘lib/cron.php‘: * Cron utils helper module
- ‘lib/dbschema.php‘: * Database schema helper module
- ‘lib/depend.php‘: * Dependencies feature
- ‘lib/export.php‘: * Export helper module
- ‘lib/files.php‘: * Files module
- ‘lib/gc.php‘: * Garbage collector helper module
- ‘lib/gdlib.php‘: * GD utils helper module
- ‘lib/geoip.php‘: * GeoIP helper module
- ‘lib/help.php‘: * Help feature
- ‘lib/import.php‘: * Import file helper module
- ‘lib/indexing.php‘: * Make index helper module
- ‘lib/log.php‘: * Log helper module
- ‘lib/math.php‘: * Math utils helper module
- ‘lib/notes.php‘: * Notes module
- ‘lib/password.php‘: * Password helper module
- ‘lib/pdf.php‘: * PDF helper module
- ‘lib/push.php‘: * Push utils helper module
- ‘lib/qrcode.php‘: * QRCode helper module
- ‘lib/score.php‘: * Score image helper module
- ‘lib/security.php‘: * Security helper module
- ‘lib/setup.php‘: * Setup helper module
- ‘lib/trash.php‘: * Send file to trash
- ‘lib/unoconv.php‘: * Unoconv library
- ‘lib/upload.php‘: * Add upload file
- ‘lib/version.php‘: * Version helper module

2.4 Action Modules (‘php/action/‘)

Handle concrete system actions (CLI-aware where needed):

- ‘action/add.php‘: * Add log action
- ‘action/app.php‘: * Application action
- ‘action/auth.php‘: * Authentication helper module

- ‘action/cron.php‘: * Garbage Collector action
- ‘action/gc.php‘: * Garbage Collector action
- ‘action/image.php‘: * BarCode action
- ‘action/indexing.php‘: * Make indexing action
- ‘action/integrity.php‘: * Make indexing action
- ‘action/ping.php‘: * Ping action
- ‘action/push.php‘: * Garbage Collector action
- ‘action/setup.php‘: * DB Schema action
- ‘action/upload.php‘: * Add files action

2.5 Other API Components

- ‘index.php‘: entry point that loads autoload and then ‘zindex.php‘
- ‘img/‘: SaltOS logos and related branding
- ‘locale/‘: multilingual resources (‘.yaml‘, ‘.odt‘, ‘.pdf‘)
- ‘xml/‘: base configuration
  - ‘config.xml‘: contains the global system configuration, including general options, paths, preferences, and base parameters that affect all of SaltOS4.
  - ‘cron.xml‘: defines the scheduled tasks that the system must run automatically, such as data cleanup, email sending, or synchronizations.
  - ‘locale.xml‘: specifies the available languages in the system and their codes, allowing the interface localization to be managed.
  - ‘bs_theme.xml‘: defines the visual themes compatible with Bootstrap that the system can use to customize the interface appearance.
  - ‘css_theme.xml‘: contains additional visual style definitions, such as colors, fonts, and CSS rules to adapt the system’s look and feel.
  - ‘dbschema.xml‘: describes the general database schema — tables, columns, indexes, and relationships required for the overall functioning of SaltOS4.
  - ‘dbstatic.xml‘: includes static data that must be inserted into certain tables during system initialization, such as user types, default values, or base configurations.

2.6 API Access

SaltOS4 supports access via HTTP or CLI.

2.6.1 Web server access (Apache, Nginx, Cherokee)

The main idea of sending information to SaltOS is to use the latest technologies, to do it, we are using restful request

- GET with REST:

    - An example request: ‘https://host/?/app/invoices/view/2‘
    - ‘@rest/0‘ = app
    - ‘@rest/1‘ = invoices
    - ‘@rest/2‘ = view
    - ‘@rest/3‘ = 2
- POST with JSON:
    - An example request: ‘https://host/?/app/invoices/insert‘
    - Use ‘Content-Type: application/json‘
    - Body parsed into ‘@json/...‘

2.6.2 Command-line access (CLI)

- ‘php api/index.php app/customers/view/100‘
- ‘user=admin php api/index.php app/customers/view/100‘
- ‘cat data.json | user=admin php app/customers/insert‘

3 Frontend Structure (web/)

Frontend is a JavaScript SPA in the ‘web/‘ folder.

- ‘index.htm‘: loads Bootstrap, SaltOS scripts
- ‘web/js/‘: client-side modules
- ‘web/lib/‘: JS libraries

JavaScript Modules:

- ‘app.js‘: * Application helper module
- ‘auth.js‘: * Authentication helper module
- ‘backup.js‘: * Backup & Autosave helper module
- ‘bootstrap.js‘: * Bootstrap helper module
- ‘common.js‘: * Common helper module
- ‘core.js‘: * Core helper module
- ‘driver.js‘: * Driver module
- ‘filter.js‘: * Filter module
- ‘form.js‘: * Form helper module
- ‘gettext.js‘: * Gettext helper module
- ‘hash.js‘: * Hash helper module
- ‘object.js‘: * Object helper module
- ‘proxy.js‘: * Proxy module

- ‘push.js‘: * Push & favicon helper module
- ‘storage.js‘: * Token helper module
- ‘token.js‘: * Token helper module
- ‘window.js‘: * Window helper module

3.1 Frontend Deployment

To use SaltOS4 from the browser:

- Publish ‘web/‘
- Create symbolic links inside ‘web/‘:
  - ‘apps/‘ → application resources
  - ‘api/‘ → backend
- Inside ‘api/‘, symbolic links to:
  - ‘data/‘ → file storage
  - ‘apps/‘ → app definitions

4 Application System

Applications in SaltOS4 are modular, defined in folders under ‘apps/‘.

Each folder may contain:

- ‘xml/manifest.xml‘: declares the apps
- ‘xml/*.xml‘ or ‘xml/*.yaml‘: application definitions
- ‘php/‘, ‘js/‘, ‘locale/‘: optional logic and translations
- ‘dbschema.xml‘, ‘dbstatic.xml‘: database structure and static content

4.1 YAML-based Apps

SaltOS4 supports YAML for fast app declarations using the common template logic.

4.1.1 tokenslog.yaml

```
app: tokenslog
require: apps/common/php/default.php
template: apps/common/xml/default.xml
indent: true
screen: type2
list:
    # [id, type, label]
    - [user_id, select, User]
    - [created_at, text, Created]
    - [token, text, Token]
```

```
    - [active, boolean, Active]
form:
    # [id, type, label]
    - [active, switch, Active]
    - [user_id, select, User]
    - [created_at, datetime, Created]
    - [updated_at, datetime, Updated]
    - [remote_addr, text, Remote Addres]
    - [user_agent, text, User Agent]
    - [token, text, Token]
    - [expires_at, datetime, Expires]
select:
    # [id, table, optional field]
    - [user_id, tbl_users]
```

4.1.2 configlog.yaml

```
app: configlog
require: apps/common/php/default.php
template: apps/common/xml/default.xml
indent: true
screen: type2
list:
    # [id, type, label]
    - [user_id, select, User]
    - [key, text, Key]
form:
    # [id, type, label]
    - [user_id, select, User]
    - [key, text, Key]
    - [val, codemirror, Value]
select:
    # [id, table, optional field]
    - [user_id, tbl_users]
attr:
    # field:
    #   attr: value
    key:
        required: true
        autofocus: true
    val:
        required: true
        mode: json
        indent: true
```

4.1.3 types.yaml

```
app: types
require: apps/common/php/default.php
template: apps/common/xml/default.xml
indent: true
screen: type5
list:
```

```
    # [id, type, label]
    - [name, text, Name]
    - [description, text, Description]
    - [active, boolean, Active]
form:
    # [id, type, label]
    - [active, switch, Active]
    - [name, text, Name]
    - [description, textarea, Description]
attr:
    # field:
    #   attr: value
    name:
        required: true
        autofocus: true
    description:
        required: true
```

These examples demonstrate the simplicity of defining apps in YAML.

4.2 XML-based Complex Apps

SaltOS4 also allows full custom apps defined in XML, often used for more complex business logic and interface customization.

4.2.1 customers.xml

This file contains the follow important nodes:

- '<main>': Defines a call to app/customers/main that returns the screen type, literals, and the navbar specification
- '<list default="true">': Defines a call to app/customers/list that returns a cache load from app/customers/list/cache
- '<list id="cache">': Defines a call to app/customers/list/cache that returns the interface of a list screen
- '<list id="data">': Defines a call to app/customers/list/data that returns the data of a list
- '<_form>': This defines a form, there is no way to access it externally, it is for internal use
- '<_data require="php/lib/log.php" eval="true">': This defines a data block of a detail view, not accessible externally, intended for internal use
- '<create>': Defines a call to app/customers/create that returns a cache load from app/customers/create/cache
- '<create id="cache">': Defines a call to app/customers/create/cache that returns the interface of a creation screen
- '<create id="insert">': Defines a call to app/customers/insert that allows inserting a record and returns the status
- '<view>': Defines a call to app/customers/view that returns a cache load from app/customers/view/-cache

- ‘<view id="cache">‘: Defines a call to app/customers/view/cache that returns the interface of a creation screen
- ‘<edit>‘: Defines a call to app/customers/edit that returns a cache load from app/customers/edit/-cache
- ‘<edit id="cache">‘: Defines a call to app/customers/edit/cache that returns the interface of a creation screen
- ‘<edit id="update">‘: Defines a call to app/customers/update that allows updating a record and returns the status
- ‘<delete>‘: Defines a call to app/customers/delete that allows deleting a record and returns the status
- ‘<action id="setup">‘: Defines a call to app/customers/setup that allows to execute the setup for this application

5 Offline Support (Service Worker)

Handled in ‘proxy.js‘ + ‘core.js‘:

- Automatically registers in the browser
- Acts as a proxy for fetch requests
- Returns cached responses if offline
- Queues write requests and syncs when online
- Works transparently with core AJAX logic

6 Makefile Overview

This ‘Makefile‘ automates building, testing, documentation, and setup tasks in SaltOS4.

6.1 Build Targets

- ‘make web‘: Builds and minifies production assets:
  - Combines and compresses CSS/JS
  - Uses ‘fixpath.php‘, ‘md5sum.php‘, ‘uglifyjs‘, ‘minify‘, ‘sha384.php‘
  - Handles app-specific JS from ‘apps/*/js/*.js‘
  - Generates the ‘proxy.js‘ script with source maps
- ‘make devel‘: Prepares a development environment with unminified assets using ‘debug.php‘.
- ‘make clean‘: Deletes generated files:
  - Minified JS, CSS, maps, HTML, ‘proxy.js‘, and per-app compiled assets

6.2 Documentation

- ‘make docs‘: Generates ‘.t2t‘, ‘.pdf‘, and ‘.html‘ documentation using scripts:

- Backend: ‘docs/api.t2t‘
- Frontend: ‘docs/web.t2t‘
- Applications: ‘docs/apps.t2t‘
- PHP tests: ‘docs/utest.t2t‘
- JS tests: ‘docs/ujest.t2t‘
- Developer manual: ‘docs/devel.t2t‘

6.3 Testing

- ‘make test‘: Runs:
  - ‘phpcs‘, ‘php -l‘, ‘phpstan‘ on PHP files
  - ‘jscs‘, ‘node -c‘ on JS files
  - Accepts variable ‘file=...‘, ‘file=all‘, or none (uses SVN diff)
- ‘make utest‘: Runs PHPUnit with optional filtering by file
- ‘make ujest‘: Runs JS tests with Jest:
  - Cleans diff snapshots and temp coverage
  - Supports full or filtered test runs
  - Generates coverage report

6.4 Environment Checks

- ‘make check‘: Verifies required folders and system commands:
  - Symbolic links: ‘api/data‘, ‘web/apps‘, etc.
  - Commands: ‘php‘, ‘node‘, ‘jest‘, ‘phpunit‘, ‘uglifyjs‘, ‘txt2tags‘, etc.

6.5 Dependency Management

- ‘make libs‘: Checks required PHP libraries via ‘checklibs.php‘

6.6 Code Statistics

- ‘make cloc‘: Uses ‘cloc‘ to count lines of code excluding minified and ignored files

6.7 System Setup

- ‘make setup‘: Initializes SaltOS4 system
- ‘make setupmysql‘: Sets up all applications using MariaDB
- ‘make setupsqlite‘: Sets up all applications using SQLite
- ‘make setupinstall‘: Combines cleaning + full MySQL + SQLite setup

- ‘make setupclean‘: Cleans data directories and resets the database

6.8 System Actions

- ‘make gc‘: Launches garbage collection
- ‘make indexing‘: Performs indexing operations
- ‘make integrity‘: Runs integrity checks
- ‘make cron‘: Executes scheduled tasks (cron)

7 Script Directory Overview

This section describes the purpose of each script and configuration file in the ‘scripts/‘ directory.

All descriptions below are based on the actual content of each file.

- ‘checklibs.php‘: Validates the current versions of required libraries by parsing ‘checklibs.txt‘, performing curl requests, and comparing base64-encoded version strings. Updates the file if needed.
- ‘checklibs.txt‘: Contains a list of required libraries, with their URLs and expected version tags encoded in base64.
- ‘debug.php‘: Generates a debug-friendly version of the frontend using ‘debug.php‘.
- ‘fixpath.php‘: Adjusts file paths in generated HTML/JS/CSS to match the deployment structure.
- ‘jest.config.js‘: Configuration file for running JavaScript unit tests with Jest.
- ‘jest_coverage.php‘: Processes Jest coverage reports and outputs them in a readable format.
- ‘jest_tester.php‘: Parses the layout structure from ‘apps/tester/xml/tester.xml‘, checks for duplicate widget tags, and exports the data as ‘/tmp/tester.json‘.
- ‘jscs.json‘: Defines JavaScript coding standards used by the JSCS code style checker.
- ‘make_bootswatch.php‘: Cleans up Bootswatch CSS files by removing any external ‘@import‘ statements.
- ‘make_instance.sh‘: Creates a new runnable SaltOS4 instance by linking ‘.htaccess‘, preparing folders like ‘data/‘ and ‘tmp/‘, and applying permissions.
- ‘makehtml.php‘: Converts a ‘.t2t‘ documentation file into an HTML file using ‘txt2tags‘.
- ‘maket2t.php‘: Scans a source directory and extracts ‘/** ... */‘ comments from each file to generate a ‘.t2t‘ documentation file.
- ‘makepdf.php‘: Converts a ‘.t2t‘ documentation file into a ‘.pdf‘ using LaTeX.
- ‘md5sum.php‘: Generates an MD5 checksum of one or more files.
- ‘migrate_v3_to_v4.php‘: Migrates a SaltOS system from version 3 to version 4, adapting data and file structure.
- ‘phpcs.xml‘: Configuration file for PHP_CodeSniffer to enforce PHP coding standards.
- ‘phpstan.neon‘: Configuration file for PHPStan, specifying analysis rules and paths for static code analysis.

- ‘phpunit.xml‘: Configuration file for PHPUnit, specifying test directories, filters, and bootstrap files.
- ‘sha384.php‘: Calculates SHA-384 hashes for files to use in Subresource Integrity (SRI) attributes in HTML.
- ‘updatet2t.php‘: Updates the second and third lines of a ‘.t2t‘ file (used to update the version and date of ‘devel.t2t‘).