Applications Documentation

SaltOS 4.0 r2018

April 2025

Contents

1 Certs

1.1 Certificate Management Functions

```
apps/certs/php/certs.php
```

This file contains the main functions for managing certificates, including listing, inserting, checking, viewing, and deleting certificates from the NSS database.

1.1.1 List Certificates

```
function __certs_list($search, $offset, $limit)
```

This function retrieves a list of certificates from the NSS database, applies search filters, and paginates the results based on offset and limit parameters.

- @search => Search term or query to filter certificates.
- @offset => Offset for pagination.
- @limit => Maximum number of certificates to retrieve.

Return the list of certificates with their ID and name.

1.1.2 Insert Certificates

```
function __certs_insert($json)
```

This function inserts certificates into the NSS database. It validates the uploaded files, processes them, and handles errors if the import is unsuccessful.

- @json => JSON object containing the certificate files and their details.

Return the status and message of the operation.

1.1.3 Convert Certificate Hash to Nickname

```
function __certs_hash2nick($hash)
```

This function maps a certificate hash to its corresponding nickname in the NSS database.

- @hash => MD5 hash of the certificate nickname.

Return the nickname of the certificate, or an empty string if not found.

1.1.4 Check Certificate Existence

```
function __certs_check($hash)
```

This function checks if a certificate with the given hash exists in the NSS database.

- @hash => MD5 hash of the certificate nickname.

Return true if the certificate exists, false otherwise.

1.1.5 View Certificate Information

```
function __certs_view($hash)
```

This function retrieves detailed information about a certificate using its hash.

- @hash => MD5 hash of the certificate nickname.

Return the status, nickname, and detailed information of the certificate.

1.1.6 Delete Certificate

```
function __certs_delete($hash)
```

This function removes a certificate from the NSS database using its hash.

- @hash => MD5 hash of the certificate nickname.

Return the status of the delete operation.

1.2 Directory helper

```
apps/certs/php/nssdb.php
```

This function returns the nssdb directory used by all functions

1.2.1 Passthru helper

```
function __nssdb_passthru_helper($cmd)
```

This function uses the ob_passthru and improve the output

- @cmd => command line to be executed

1.2.2 Init nssdb repo

```
function __nssdb_init()
```

This function tries to initialize the nssdb component with an empty repo

1.2.3 Create certificate

```
function __nssdb_create($outfile, $outpass, $subject = '', $name = '')
```

This function create a dummy self signed certificate intended to be used in test cases

$outfile => the p12 file $outpass => the password used by the p12 file

1.2.4 Add certificate

```
function __nssdb_add($file, $pass)
```

This function adds the p12 file that must contains a valid certificate to the nssdb repo using the pass if needed

- @file => the p12 file
- @pass => the p12 password

1.2.5 List certificates

```
function __nssdb_list()
```

This function returns the list of valid nicks that you can use with pdfsig

1.2.6 Info of a certificate

```
function __nssdb_info($nick, $shortnames = false)
```

This function returns the info of the nick certificate of the nssdb repo

- @nick => the desired nick used to retrieve info

1.2.7 Pdf signature

```
function __nssdb_pdfsig($nick, $input, $output)
```

This function uses the pdfsig to add the signature to the pdf using the nick certificate of the nssdb repo

- @nick => the desired nick used to sign de pdf
- @input => the desired input file that you want to sign
- @output => the signed pdf file

1.2.8 Remove certificate

```
function __nssdb_remove($nick)
```

This function remove the certificate from the nssdb repo indentified by nick

- @nick => the desired nick that you want to remove

1.2.9 Reset nssdb repo

```
function __nssdb_reset()
```

This function removes the nssdb repo, is the inverted of the init function

1.2.10 Update pdf

```
function __nssdb_update($nick, $input)
```

This function creates a new pdf document using input as source and adding to each page a new box with the important information about the certificate used in the signature process

- @nick => the desired nick used to sign de pdf
- @input => the desired input pdf file where you want to add the cert info

2 Common

2.1 Profile application

```
apps/common/js/profile.js
```

This application implements the typical features associated with user profiles, such as managing themes, language settings, and authentication updates.

2.1.1 Main object

```
saltos.profile = {};
```

Contains all the logic and code for the SaltOS framework related to the profile application.

2.1.2 Initialization of profile settings

```
saltos.profile.init = arg
```

This method initializes the profile settings by setting the current Bootstrap theme, custom CSS theme, and language preferences in the respective input fields.

2.1.3 Update authentication settings

```
saltos.profile.authupdate = ()
```

This method restores the previous state of the application if necessary, validates required fields, and then updates the authentication credentials using the provided old password, new password, and its confirmation.

2.2 Data Actions Merger

```
apps/common/php/actions.php
```

This file contains functionality for merging action controls with data rows in tables or lists, including permission checks

2.2.1 Merge data with actions

```
function __merge_data_actions($data, $actions)
```

Combines dataset rows with action controls while verifying permissions. Processes each action definition, checks user permissions, and attaches permitted actions to each data row.

- @data => Dataset containing rows to display in table/list
- @actions => Action definitions to merge with each data row

Returns the modified dataset with actions attached to each row

Throws error if actions parameter is not an array

2.3 Apps helper module

```
apps/common/php/default.php
```

This file contains functions intended to be used as hepers of other functions, allowing to convert between formats like yaml to xml using a small specification

2.3.1 Make app file

```
function make_app_file($data)
```

This function returns the xml of an app using the follow specs passed in @data:

- @require => this field defines the required php that contains the make_app_file function like this file
- @_template => the xml app file used as template, this file contains the default specification of the app used in the quick apps defined in the yaml
- @screen => the screen used by the template
- @list => the list of fields used in the list screen, this must be a list of arrays with 3 elements: id, type and label, types can be as follow: -text => default field with text -select => this field resolves the text from a table and field, see the @select spec -boolean => this field uses an icon with V o X in green or red to see the value of the field (generaly 1 or 0) -hastext => this field is similar to boolean but the true or false is detedmined using the contents of the text, no text is false and some text is true
- @form => the list of fields used in the form screen, this must be a list of arrays with 3 elements: id, type and label, types can be as follow:
  - text, date, time, datetime, checkbox, switch => default widget
  - textarea, ckeditor, codemirror => widget with 10em of height
  - select => widget that uses the select spec to resolve the contents
- @select => this spec is intended to defined the selects used in the list and forms, requires an array of 3 elements: id, table and field, if the last field is not specified, saltos tries to resolve it using the manifest information (internally use the table2field feature)
- @attr => this part contain an array with extra features added to the forms widgets, the spec requires to use a named array that defines the id of the field and each array entry must to be another pair of key and val with the name of the property and the value of it

2.4 File management functions

```
apps/common/php/files.php
```

These functions handle file operations such as listing files, retrieving file paths, checking for file existence, and viewing file contents. They are primarily designed to interact with the 'data/logs' directory.

2.4.1 List files in the logs directory

```
function __files_list($search, $offset, $limit)
```

This function retrieves a list of files from the 'data/logs' directory, applies search filters, and paginates the results based on the specified offset and limit. The returned data includes metadata such as file size and type.

- @search => Search term or query to filter files.
- @offset => Offset for pagination.
- @limit => Maximum number of files to retrieve.

Returns a list of files with metadata including ID, name, size, and type.

2.4.2 Get the full path of a file

```
function __files_getfile($file)
```

This function searches for a file by its name in the 'data/logs' directory and returns its full path if found. Otherwise, it returns an empty string.

- @file => Name of the file to search for.

Returns the full path of the file or an empty string if not found.

2.4.3 Check if a file exists

```
function __files_check($file)
```

This function checks whether a file exists in the 'data/logs' directory by searching for its name.

- @file => Name of the file to check.

Returns 'true' if the file exists, otherwise 'false'.

2.4.4 View the contents of a file

```
function __files_view($file)
```

This function retrieves the contents of a file from the 'data/logs' directory. It supports both regular files and gzip-compressed files. The returned data includes the file name, size, type, and content.

- @file => Name of the file to view.

Returns metadata and content of the file. If the file is not found, it returns an error response.

2.5 Matrix functions

```
apps/common/php/matrix.php
```

This file contains all the functions required by the Excel widget for handling version and log data in a matrix format.

2.5.1 Create matrix for version data

```
function make_matrix_version($app, $id)
```

This function generates a matrix structure for displaying version-related data in the Excel widget, including headers, data formatting, and cell attributes.

2.5.2 Create matrix for log data

```
function make_matrix_log($app, $id)
```

This function processes log data and generates headers and formatting information for integration into the Excel widget.

3 Crm

3.1 Quotes application

```
apps/crm/js/quotes.js
```

This application provides core functionalities for managing quotes, including creating, editing, viewing, and exporting quotes in PDF format.

3.1.1 Main object

```
saltos.quotes = {};
```

This object contains all SaltOS code related to quote operations.

3.1.2 Initialize the quotes application

```
saltos.quotes.init = arg
```

This method handles initialization tasks based on the type of operation (‘view‘, ‘create‘, or ‘edit‘). It updates UI elements and attaches appropriate event listeners.

- @arg => Specifies the type of operation to initialize.

3.1.3 Calculate the dynamic width for each column in the "lines" table.

```
saltos.quotes.colWidths_lines = index
```

This function returns the width for each column. For the first column (index 0), it dynamically calculates the remaining space based on the container width minus a fixed width (500px) reserved for other columns. For all other columns, it returns 100px.

- @index => Column index

Returns the width in pixels

3.1.4 Define cell configuration for each column in the "lines" table.

```
saltos.quotes.cells_lines = (row, column, prop)
```

This function provides per-column behavior:

- Column 4: renders a dropdown with available tax values (from #alltaxes).
- Column 5: sets the cell as read-only (used for calculated values).
- Other columns: no special config.
- @row => Row index
- @column => Column index
- @prop => Column property name

Returns the configuration object

3.1.5 Calculate the dynamic width for each column in the "taxes" table.

```
saltos.quotes.colWidths_taxes = index
```

The first column (index 0) fills the available space minus 200px reserved for other columns. All other columns get a fixed width of 100px.

- @index => Column index

Returns the width in pixels

3.1.6 Define cell configuration for each column in the "taxes" table.

```
saltos.quotes.cells_taxes = (row, column, prop)
```

All cells in this table are read-only.

- @row => Row index
- @column => Column index
- @prop => Column property name

Returns the configuration object

3.1.7 Calculate equal column widths for the "totals" table.

```
saltos.quotes.colWidths_totals = index
```

The table is divided into 3 equal columns, based on the container width.

- @index => Column index

Returns the width in pixels

3.1.8 Define cell configuration for the "totals" table.

```
saltos.quotes.cells_totals = (row, column, prop)
```

All cells in this table are read-only.

- @row => Row index
- @column => Column index
- @prop => Column property name

Returns the configuration object

3.1.9 Handle and recalculate quote line totals and tax breakdowns after user edits.

```
saltos.quotes.afterChange_lines = (changes, source)
```

This function is triggered after the user modifies any cell in the "lines" matrix (quote lines). It performs the following operations:

- Ignores changes triggered by internal actions (non-'edit' sources).
- Validates and normalizes quantity, price, and discount values:
  - Ensures 'line[1]', 'line[2]', 'line[3]' (qty, price, discount) are numeric.
  - Defaults discount to 0 if empty but qty and price are valid.
- Calculates the line total as: 'qty × price × (1 - discount%)', rounded to 2 decimals.
- Updates the lines grid to reflect computed values.
- Recomputes the taxes matrix:
  - Aggregates taxable bases per tax value ('line[4]').
  - Computes tax amount for each group: 'base × (tax% / 100)'.
- Rounds all tax results to 2 decimals and updates the taxes grid.
- Computes the totals matrix:
  - 'subtotal' = sum of all taxable bases
  - 'tax' = sum of all tax amounts
  - 'total' = subtotal + tax
  - Results are rounded and shown in the totals grid.
- @changes => Array of cell changes from the spreadsheet component

- @source => Source of the change event (only ''edit'' triggers processing)

3.2 Reverses the process of 'make_matrix_data()' for the 'quotes' app.

```
apps/crm/php/quotes.php
```

This function takes a previously generated matrix-style array of quote data (grouped as lines, taxes, and totals), compares it with the current values in the database, and builds a minimal diff-like array that includes only the fields that have changed, been removed, or added.

3.2.1 unmake_matrix_data

```
function unmake_matrix_data($json, $quote_id)
```

Reverse the matrix-encoded quote data to a diff-style data array.

- @json => The matrix-encoded quote data (from make_matrix_data)
- @quote_id => The quote ID used to retrieve the original data

Return a diff array representing only the differences between the provided data and the current state of the database.

3.2.2 Generates or completes quote and final quote metadata.

```
function set_quote($json, $quote_id)
```

This function ensures that the given quote JSON structure has appropriate values for:

- 'code', 'date' and 'valid_until' (if not provided or if it's a new quote)

Rules:

- If no 'quote_id' is given or 'code' is not defined, a new code number is generated.
- @json => Quote data to process and complete.
- @quote_id => ID of the quote, used to retrieve current status from DB.

Return the updated JSON with appropriate codes and dates filled in.

4 Dashboard

4.1 Dashboard application

```
apps/dashboard/js/dashboard.js
```

This module implements the typical features associated with a dashboard, allowing dynamic interaction with various elements and functionalities.

4.1.1 Main object

```
saltos.dashboard = {};
```

Contains all the logic and code for the SaltOS framework related to the dashboard.

4.1.2 Initialization of dashboard

```
saltos.dashboard.init = arg
```

This method sets up listeners, configures the catalog layout, and initializes the dashboard widgets based on user-defined or default configurations.

4.2 Dashboard widgets application

```
apps/dashboard/js/dashboard_widgets.js
```

This module implements the typical features associated with a widget dashboard, allowing interaction and customization of elements in a dynamic environment.

4.2.1 Main object

```
saltos.dashboard_widgets = {};
```

Contains all the logic and code for the SaltOS framework related to the dashboard widgets.

4.2.2 Initialization of dashboard widgets

```
saltos.dashboard_widgets.init = arg
```

This method sets up the interactive elements of the dashboard, including assigning style classes and configuring drag-and-drop functionality between different containers.

4.3 Dashboard and navbar generation logic.

```
apps/dashboard/php/dashboard.php
```

This file contains helper functions used to build the SaltOS dashboard and navbar dynamically based on the applications configured in the system and the permissions of the current user.

Functions included:

- __dashboard_helper(): Generates the full dashboard widget layout (alerts, buttons, separators)
- __dashboard_config(): Applies user-specific configuration to customize the dashboard layout
- __navbar_helper(): Builds the application menu structure for the top navigation bar

These functions are used internally by the SaltOS UI rendering engine to generate the initial layout and navigation menus seen by each user on login.

4.3.1 Build the default dashboard layout with groups and application buttons.

```
function __dashboard_helper()
```

This function retrieves all active applications from ‘tbl_apps‘, groups them by ‘group‘, filters them by user permissions (menu access), and then formats them into a widget-based structure suitable for rendering on the dashboard, including alerts, buttons, and separators.

Each group of apps is preceded by an alert (group title and description), followed by buttons for each app, and a horizontal rule (‘<hr>‘) at the end.

- @return array An array of widgets (alerts, buttons, hr) to render the dashboard

4.3.2 Build a personalized dashboard layout based on user config.

```
function __dashboard_config()
```

This function loads the full list of dashboard items using ‘__dashboard_helper()‘ and then tries to fetch the widget configuration saved by the current user under the path ‘app/dashboard/widgets/default‘.

If configuration is found, it reconstructs the dashboard by preserving only the items selected by the user, in the specified order. Otherwise, it returns the default layout.

- @return array The personalized or default dashboard widget list

4.3.3 Build the application menu for the navbar.

```
function __navbar_helper()
```

This function generates the main application menu for the navbar by:

- Fetching all active apps grouped by ‘group‘
- Filtering them by user permissions
- Mapping the group metadata from ‘tbl_apps_groups‘
- Creating a linear structure of ‘<item>‘ elements including:
  - Group labels (disabled items)
  - App entries with onclick handlers
  - A divider after each group

The result is an array of menu items that represents the full application navigation tree grouped visually in the UI.

- @return array An array of navbar ‘<item>‘ definitions

5 Emails

5.1 Email application

```
apps/emails/js/driver.js
```

This application implements the typical features associated with email functionality, including templates and initialization processes for handling email-related tasks.

5.1.1 Driver emails object

```
saltos.driver.__types.emails = {};
```

This object stores the functions and properties used by the email driver, providing the necessary methods for managing email operations.

5.1.2 Create email template

```
saltos.driver.__types.emails.template = arg
```

This function generates a template for the email driver, configuring specific attributes and layout modifications. It sets the 'type' attribute to 'emails' and adjusts the layout of the corresponding element for proper display.

5.1.3 Initialization, open, and close handlers for emails

```
saltos.driver.__types.emails.init = saltos.driver.__types.type5.init; //
    Inherits initialization from type5
```

These methods inherit their implementations from the 'type5' driver, allowing reuse of core functionality for initializing, opening, and closing email resources.

5.2 Email application

```
apps/emails/js/emails.js
```

This application implements typical features associated with email functionality, such as initialization, server actions, email deletion, sending, setting states, handling signatures, generating and downloading PDFs, viewing email sources, replying and forwarding.

5.2.1 Main object

```
saltos.emails = {};
```

This object contains all SaltOS code related to email operations.

5.2.2 Initialize the email application

```
saltos.emails.init = arg
```

This method handles initialization tasks for different views (‘create‘, ‘view‘, ‘close‘, ‘list‘) in the email application. It manages UI updates, visibility behaviors, and email state changes.

- @param {string} arg Specifies the type of view or action to initialize.

5.2.3 Perform server-related actions for emails

```
saltos.emails.server = ()
```

This method triggers server-side actions for emails and displays the responses in toasts.

5.2.4 Delete selected emails

```
saltos.emails.delete1 = ()
```

This method deletes multiple emails selected by the user, after showing a confirmation modal.

5.2.5 Delete a single email

```
saltos.emails.delete2 = ()
```

This method deletes the currently viewed email after showing a confirmation modal.

5.2.6 Send an email

```
saltos.emails.send = ()
```

This method validates the required fields and sends the email data to the server. It handles responses and clears unsaved data on success.

5.2.7 Set email state

```
saltos.emails.setter = what
```

This function updates the state of a specific email identified by its ID. The updated state is sent to the server, and the UI is refreshed afterward.

- @param {string} what Specifies the state to set for the email.

5.2.8 Add signature to email

```
saltos.emails.signature = ()
```

This function applies the email signature based on the selected account and its related data. The signature is retrieved from the server and updated in the email form.

5.2.9 View email source

```
saltos.emails.source = ()
```

This function opens the raw source of the selected email for viewing in the driver.

5.2.10 Reply to an email

```
saltos.emails.reply = ()
```

This function opens the reply form for the currently selected email.

5.2.11 Reply to all recipients of an email

```
saltos.emails.replyall = ()
```

This function opens the reply-all form for the currently selected email, including all recipients.

5.2.12 Forward an email

```
saltos.emails.forward = ()
```

This function opens the forward form for the currently selected email.

5.3 Email Query Builder

```
apps/emails/php/filter.php
```

This function builds an SQL WHERE query for filtering emails based on various criteria such as account, fields, search terms, dates, and specific email states.

5.3.1 Generate WHERE query for emails

```
function make_where_query_emails($json)
```

This function takes a JSON input and generates a dynamic SQL WHERE clause to filter emails. The query is constructed based on several optional parameters such as account ID, search fields, dates, and flags indicating specific states (e.g., new emails, waiting emails, spam).

- @json => Input parameters for generating the query.

Returns the dynamically constructed SQL WHERE clause.

5.4 Get email library

```
apps/emails/php/getmail.php
```

This library provides the necesary functions to download, parse and manage emails.

5.4.1 Requires section

```
require_once 'apps/emails/lib/mimeparser/mime_parser.php';
```

This requires loads the external libraries needed to run this library.

5.4.2 Defines section

```
// phpcs:disable Generic.Files.LineLength
```

This defines allow to define some useful standards to do html pages and more.

5.4.3 Remove all body

```
function __getmail_removebody($array)
```

This function removes the body entry in the array, it is only for debug purposes

- @aarray => The array that you want to process

5.4.4 Process message

```
function __getmail_processmessage($disp, $type)
```

This function returns a boolean that identify if the disposition and the type allow the node to be processed.

- @disp => disposition, can be inline or attachment
- @type => type, can be message, html or plain

5.4.5 Process plain html

```
function __getmail_processplainhtml($disp, $type)
```

This function returns a boolean that identify if the disposition and the type allow the node to be processed.

- @disp => disposition, can be inline or attachment
- @type => type, can be message, html or plain

5.4.6 Process file

```
function __getmail_processfile($disp, $type)
```

This function returns a boolean that identify if the disposition and the type allow the node to be processed.

- @disp => disposition, can be inline or attachment
- @type => type, can be message, html or plain

5.4.7 Check permissions

```
function __getmail_checkperm($id)
```

This function allow to check if the current user has permissions to view the message identified by the id argument

- @id => id of the email

5.4.8 Get GZ File

```
function __getmail_gzfile($id)
```

This function returns the file that contains the email

- @id => id of the email

5.4.9 Get source

```
function __getmail_getsource($id)
```

This function returns the original RFC822 message as string

- @id => id of the email

Notes:

This function returns the email source without the base64 attachments

5.4.10 Mime decode protected

```
function __getmail_mime_decode_protected($input)
```

This function decodes the input string that contains the RFC822 message using the mime_parser_class to do it, and returns the decoded array.

- @input => the RFC822 string that contains the message

5.4.11 Get mime

```
function __getmail_getmime($id)
```

This function returns the decoded array of the email identified by the id argument, to do this more optimal, this function uses an internal cache file to improve the performance for repeated executions.

- @id => id of the email

5.4.12 Get Node

```
function __getmail_getnode($path, $array)
```

This function returns the node using a xpath notation

- @path => xpath that identify the desired path that must to be returned
- @array => the decoded message in an array format

5.4.13 Get type

```
function __getmail_gettype($array)
```

This function tries to unify the differents content-type to standarize it into the follow formats: html, plain, messsage, alternative, multipart or other.

- @array => the decoded message in an array format

5.4.14 Get disposition

```
function __getmail_getdisposition($array)
```

This function tries to unify the differents content-dispoaition to standarize it into the follow formats: attachment, inline or other.

- @array => the decoded message in an array format

5.4.15 Get files

```
function __getmail_getfiles($array, $level = 0)
```

This function returns an array with the attachment files of the message

- @array => the decoded message in an array format
- @level => this parameter is internally used to detect recursion

5.4.16 Get info

```
function __getmail_getinfo($array)
```

Returns all information of the decoded message in a structured format

- @array => the decoded message in an array format

5.4.17 Fix string

```
function __getmail_fixstring($arg)
```

This function is a helper used by all functions that pcoesses the headers of the decoded message.

- @arg => the string that must to be checked and fixed if needed

5.4.18 Get text body

```
function __getmail_gettextbody($array, $level = 0)
```

This function returns all text body concatenated as an unique string

- @array => the decoded message in an array format
- @level => this parameter is internally used to detect recursion

5.4.19 Get full body

```
function __getmail_getfullbody($array)
```

This function returns all body and attachments information as an array

- @array => the decoded message in an array format

5.4.20 Get cid

```
function __getmail_getcid($array, $hash)
```

This function returns the requested attachment indentified by the hash argument

- @array => the decoded message in an array format
- @hash => the hash of the content requested

5.4.21 Insert

```
function __getmail_insert(
```

This function do the insert in the app_emails table, and

- @file => the gzfile that contains the message in RFC822 format
- @messageid => the id of the message (account_id/uidl)
- @state_new => the 0/1 that sets the state new flag
- @state_reply => the 0/1 that sets the state reply flag
- @state_forward => the 0/1 that sets the state forward flag
- @state_wait => the 0/1 that sets the state wait flag
- @id_correo => the id of the related email (used to create relations between emails)
- @is_outbox => the 0/1 that sets the is outbox flag
- @state_sent => the 0/1 that sets the state sent flag
- @state_error => the string that contains the error (if exists an error)

5.4.22 Update

```
function __getmail_update($field, $value, $id)
```

This function updates the field with the value of the app_emails for the register identified by the id argument.

- @field => field that you want to update
- @value => value that you want to set
- @id => id of the register to do the update

5.4.23 Add bcc

```
function __getmail_add_bcc($id, $bcc)
```

This function adds the bbc to the database, this is because the messages does not contains the bcc field (is hidden in theory), and only is available if the current execution is the sender of the message.

- @id => id of the email
- @bcc => an array with the addresses of the emails

5.4.24 Gzfile size

```
function gzfilesize($filename)
```

This function is copied from http://php.net/manual/es/function.gzread.php#110078 and allow to know the file size of the file after a gzip descompression.

- @filename => the gzip filename that you want to know the size

5.4.25 Get email body

```
function getmail_body($id, $images = false)
```

This function returns the string that contains the body of the email intended to be rendered in an iframe, for example

- @id => id of the email

5.4.26 Generate formatted email header information

```
function __getmail_head_helper($decoded, $email_id)
```

This function extracts and formats header details from the decoded email structure, including sender, recipients, date, subject, and attachments. It handles edge cases such as missing data, and ensures that all information is presented in a structured and readable format. HTML encoding is applied to ensure safe display within a web interface.

- @decoded => Decoded structure of the email, containing metadata and header information.
- @email_id => ID of the email to retrieve additional account-specific details.

Returns a formatted HTML string with the email header information.

5.4.27 Extract and format the body of an email

```
function __getmail_body_helper($decoded, $images = false)
```

This function processes the decoded content of an email, extracting the plain text and HTML bodies while applying necessary transformations for safe and structured display. It handles inline images, styles, and embedded attachments, ensuring compatibility and security.

- @decoded => Decoded structure of the email, typically containing nodes with metadata and body content.

- @images => Specifies whether inline images should be embedded directly into the content.

Returns the formatted email body, including plain text, HTML, and inline attachments.

5.4.28 Get email source

```
function getmail_source($id)
```

This function returns the string that contains the source of the email intended to be rendered in an iframe, for example

- @id => id of the email

5.4.29 Get email files

```
function getmail_files($id)
```

This function returns an arryy that contains the files of the email intended to be rendered in an table, for example

- @id => id of the email

5.4.30 Get cid

```
function getmail_cid($id, $cid)
```

This function returns the requested attachment indentified by the cid argument

- @id => id of the email
- @cid => the cid of the content requested

5.4.31 Is outbox

```
function getmail_field($field, $id)
```

Returns the field of the email identified by the id argument

- @field => field requested
- @id => id of the email

5.4.32 Server

```
function getmail_server()
```

This function implements the old getmail action of the old saltos.

5.4.33 Delete

```
function getmail_delete($ids)
```

This function implements the old delete action of the old saltos.

- @ids => array with the emails id

5.4.34 Get viewpdf

```
function getmail_viewpdf($id, $cid)
```

This function returns the requested attachment indentified by the cid argument in a pdf format for the viewpdf widget

- @id => id of the email
- @cid => the cid of the content requested

5.4.35 Get download

```
function getmail_download($id, $cid)
```

This function returns the requested attachment indentified by the cid argument in an array format for the download feature

- @id => id of the email
- @cid => the cid of the content requested

5.4.36 Update email states

```
function getmail_setter($ids, $what)
```

This function modifies the state of emails (e.g., 'new', 'wait', or 'spam') based on the provided IDs and action. It validates user permissions for the selected emails before updating their states. The function calculates the number of records to be modified and updates the database accordingly.

- @ids => Comma-separated list of email IDs to be updated.
- @what => Specifies the state change in the format 'key=value' (e.g., 'new=0').

Returns a response message indicating the number of emails modified.

5.4.37 Generate PDF from emails

```
function getmail_pdf($ids)
```

This function generates PDF files for the provided email IDs using either the 'wkhtmltopdf' command or a fallback library. If multiple emails are selected, their PDFs are merged into a single file. It validates user permissions for accessing the emails and manages caching for performance optimization.

- @ids => List or comma-separated string of email IDs to generate PDFs for.

Returns metadata of the generated PDF file, including its name, type, and base64-encoded data.

5.4.38 Generate iframe content with secure policies

```
function __iframe_srcdoc_helper($html)
```

This function generates the ‘srcdoc‘ content for an iframe element, embedding the provided HTML string. It ensures the content is styled using predefined fonts and enforces a strict Content Security Policy (CSP) to limit the sources for styles, fonts, and images. This helps maintain security and compatibility within the iframe.

- @html => The HTML content to be embedded in the iframe.

Return a complete HTML document string suitable for ‘srcdoc‘ use in an iframe.

5.5 Html helper module

```
apps/emails/php/html.php
```

This file contain useful html helper functions

5.5.1 Defines section

```
define('__GIF_IMAGE__', '
   ywAAAAACgABAAACA4SPBQA7');
```

This defines allow to define some useful needed resources by this file.

5.5.2 Remove Script Tag

```
function remove_script_tag($html)
```

This function tries to remove all <script> tags of the string

- @temp => the string that you want to process

5.5.3 Remove Style Tag

```
function remove_style_tag($html)
```

This function tries to remove all <style> tags of the string

- @temp => the string that you want to process

5.5.4 Remove Comment Tag

```
function remove_comment_tag($html)
```

This function tries to remove all <!– –> tags of the string

- @temp => the string that you want to process

5.5.5 Remove Meta Tag

```
function remove_meta_tag($html)
```

This function tries to remove all <meta > tags of the string

- @temp => the string that you want to process

5.5.6 Remove Link Tag

```
function remove_link_tag($html)
```

This function tries to remove all <link > tags of the string

- @temp => the string that you want to process

5.5.7 Inline Img Tag

```
function inline_img_tag($html)
```

This function tries to convert all imgs to an inline imgs

- @temp => the string that you want to process

5.5.8 Inline Img Style

```
function inline_img_style($html)
```

This function tries to convert all imgs to an inline imgs

- @temp => the string that you want to process

5.5.9 Inline Img Background

```
function inline_img_background($html)
```

This function tries to convert all imgs to an inline imgs

- @temp => the string that you want to process

5.5.10 Inline Image Helper

```
function __inline_img_helper($src)
```

This function fetches and processes image URLs, ensuring compatibility and resizing where necessary. It also manages caching and handles errors gracefully.

- @src => Image source URL or path.

Returns the inline image data or a placeholder GIF in case of errors.

5.5.11 Extract Inline Images from Tags

```
function extract_img_tag($html)
```

This function extracts inline images from HTML ‘<img>‘ tags and replaces their ‘src‘ attributes with ‘cid‘ references for email compatibility.

- @html => HTML content to parse.

Returns the modified HTML content and an array of image files.

5.5.12 Extract Inline Images from Styles

```
function extract_img_style($html)
```

This function extracts images embedded in CSS ‘style‘ attributes and replaces them with ‘cid‘ references for email compatibility.

- @html => HTML content to parse.

Returns the modified HTML content and an array of image files.

5.5.13 Extract Inline Images from Backgrounds

```
function extract_img_background($html)
```

This function extracts images embedded in HTML ‘background‘ attributes and replaces them with ‘cid‘ references for email compatibility.

- @html => HTML content to parse.

Returns the modified HTML content and an array of image files.

5.5.14 Fix Image Tags

```
function fix_img_tag($html)
```

This function processes ‘<img>‘ tags in the given HTML to replace non-inline image ‘src‘ attributes with a placeholder image (‘GIF_IMAGE‘), ensuring compatibility and avoiding external dependencies.

- @html => The HTML content to process.

Returns the updated HTML content with fixed image tags.

5.5.15 Fix Image Styles

```
function fix_img_style($html)
```

This function processes CSS ‘style‘ attributes in HTML elements to replace non-inline image URLs within ‘url()‘ properties with a placeholder image (‘GIF_IMAGE‘).

- @html => The HTML content to process.

Returns the updated HTML content with fixed image styles.

5.5.16 Fix Image Backgrounds

```
function fix_img_background($html)
```

This function processes 'background' attributes in HTML elements to replace non-inline image URLs with a placeholder image ('GIF_IMAGE').

- @html => The HTML content to process.

Returns the updated HTML content with fixed image backgrounds.

5.5.17 Fix Base64-Encoded Files

```
function fix_file_b64($html)
```

This function replaces inline base64-encoded file data in HTML with the corresponding cached image content, ensuring that embedded images are properly handled.

- @html => The HTML content to process.

Return the updated HTML content with fixed base64-encoded files.

Notes:

We are using '{48}' to match base64 data of 48 bytes, which decodes into a 36-byte string that matches the MD5 hash style used in cached files (32 bytes plus 4-byte '.b64' extension).

5.6 Make indexing action

```
apps/emails/php/indexing.php
```

This fie contains useful functions related to the indexing action that internally uses the mroonga engine to search in the fulltext string generated by this action

5.7 Send email library

```
apps/emails/php/sendmail.php
```

This library provides the necesary functions to send emails.

5.7.1 Used libraries

```
use PHPMailer\PHPMailer\PHPMailer;
```

This use loads the external libraries needed to run this library.

5.7.2 Sendmail

```
function sendmail($account_id, $to, $subject, $body, $files = '', $async =
    true)
```

This function send an email in synchronous and/or asynchronous mode

$account_id => the account id used to detect the source of the email $to => can be an string with the destination email or an array with the follow prefixes => to:, cc:, bcc:, crt:, priority:, sensitivity:, replyto $subject => the subject string $body => the body string $files => an array with files

Notes:

This function must return the last_id if success, and a string if error

5.7.3 Debugoutput helper

```
function __sendmail_debugoutput_helper($str, $level)
```

This function tries to echoed all debug information with the SMTP Error: prefix, with this feature we can capture smtp errors with more details like the error stored in $mail->ErrorInfo that always contains SMTP connect() failed, the signature of this function will accomplish the neested callback arguments

- @str => the debug trace string that can contain the errors
- @level => unused in this function

5.7.4 Parser

```
function __sendmail_parser($oldaddr)
```

This function gets an address and tries to detect the name part and the addr part of the argument. It's returns an array with two elements, the first is for the addr and the second is for the name.

- @oldaddr => the string that must to be processed

5.7.5 Message Id

```
function __sendmail_messageid($account_id, $from)
```

This function returns the message id for a new email, to do it, tries to detect the outbox directory, compute an aproximation to the newest value and checks that is unique in the system to prevent concurrence.

- @account_id => the account id used to send the new email
- @from => the from used to compute the crc32

5.7.6 Eml saver

```
function __sendmail_emlsaver($message, $messageid)
```

This function is intended to save the RFC822 message into the eml gzfile

- @message => the contents in RFC822 format of the message
- @messageid => the message id computed previously

5.7.7 Obj saver

```
function __sendmail_objsaver($mail, $messageid)
```

This function is intended to save the PHPMailer object into the obj file

- @mail => the PHPMailer object of the asynchronous transaction
- @messageid => the message id computed previously

5.7.8 Prepare email for sending

```
function sendmail_prepare($action, $email_id)
```

This function constructs the required parameters for sending an email, including sender details, recipients, subject, body, and attachments. The behavior varies based on the specified action ('reply', 'replyall', or 'forward'). It retrieves necessary data from the database and processes the email's metadata to ensure compatibility with the given action.

- @action => Specifies the action to perform ('reply', 'replyall', or 'forward').
- @email_id => ID of the email being replied to, forwarded, or used for metadata.

Return the prepared email parameters, including sender, recipients, subject, body, state, and attachments.

5.7.9 Perform email sending action

```
function sendmail_action($json, $action, $email_id)
```

This function handles the process of sending an email, including assembling email data, recipients, and attachments. It also manages inline images, updates email states, and cleans up temporary files after execution. The function supports multiple actions such as 'reply', 'replyall', and 'forward'.

- @json => JSON object containing email data (e.g., recipients, subject, body, attachments).
- @action => Specifies the action to perform ('reply', 'replyall', or 'forward').
- @email_id => ID of the email being replied to, forwarded, or used for metadata.

Returns the status and message of the email action.

5.7.10 Send queued emails from the server

```
function sendmail_server()
```

This function processes and sends emails queued in the outbox. It uses a semaphore mechanism to prevent simultaneous execution, handles SMTP settings dynamically, updates email states, and manages errors during the sending process. The function is intended for use in cron jobs.

- @return array Returns an array of messages detailing the sending result, including errors and successes.

5.7.11 Retrieve email attachments

```
function sendmail_files($action, $email_id)
```

This function handles the retrieval of email attachments based on the specified action, such as 'forward'. It validates permissions, decodes the message, and stores attachments locally if not already uploaded. Finally, it retrieves a list of attachments from the upload table.

- @action => Specifies the email action ('forward' in this case).
- @email_id => ID of the email to retrieve attachments from.

Returns a list of attachments with metadata including ID, app, name, size, type, and hash.

5.7.12 Update email signature, CC, and encryption state

```
function sendmail_signature($json)
```

This function updates the email body with a new signature, adjusts the CC list, and replaces the encryption state ('state_crt') based on the selected account.

- @json => JSON object containing email data and account information.

Return the updated email body, CC list, and encryption state.

6 Sales

6.1 Invoices application

```
apps/sales/js/invoices.js
```

This application provides core functionalities for managing invoices, including creating, editing, viewing, and exporting invoices in PDF format.

6.1.1 Main object

```
saltos.invoices = {};
```

This object contains all SaltOS code related to invoice operations.

6.1.2 Initialize the invoices application

```
saltos.invoices.init = arg
```

This method handles initialization tasks based on the type of operation ('view', 'create', or 'edit'). It updates UI elements and attaches appropriate event listeners.

- @arg => Specifies the type of operation to initialize.

6.1.3 Calculate the dynamic width for each column in the "lines" table.

```
saltos.invoices.colWidths_lines = index
```

This function returns the width for each column. For the first column (index 0), it dynamically calculates the remaining space based on the container width minus a fixed width (500px) reserved for other columns. For all other columns, it returns 100px.

- @index => Column index

Returns the width in pixels

6.1.4 Define cell configuration for each column in the "lines" table.

```
saltos.invoices.cells_lines = (row, column, prop)
```

This function provides per-column behavior:

- Column 4: renders a dropdown with available tax values (from #alltaxes).
- Column 5: sets the cell as read-only (used for calculated values).
- Other columns: no special config.
- @row => Row index
- @column => Column index
- @prop => Column property name

Returns the configuration object

6.1.5 Calculate the dynamic width for each column in the "taxes" table.

```
saltos.invoices.colWidths_taxes = index
```

The first column (index 0) fills the available space minus 200px reserved for other columns. All other columns get a fixed width of 100px.

- @index => Column index

Returns the width in pixels

6.1.6 Define cell configuration for each column in the "taxes" table.

```
saltos.invoices.cells_taxes = (row, column, prop)
```

All cells in this table are read-only.

- @row => Row index
- @column => Column index
- @prop => Column property name

Returns the configuration object

6.1.7 Calculate equal column widths for the "totals" table.

```
saltos.invoices.colWidths_totals = index
```

The table is divided into 3 equal columns, based on the container width.

- @index => Column index

Returns the width in pixels

6.1.8 Define cell configuration for the "totals" table.

```
saltos.invoices.cells_totals = (row, column, prop)
```

All cells in this table are read-only.

- @row => Row index
- @column => Column index
- @prop => Column property name

Returns the configuration object

6.1.9 Handle and recalculate invoice line totals and tax breakdowns after user edits.

```
saltos.invoices.afterChange_lines = (changes, source)
```

This function is triggered after the user modifies any cell in the "lines" matrix (invoice lines). It performs the following operations:

- Ignores changes triggered by internal actions (non-'edit' sources).
- Validates and normalizes quantity, price, and discount values:
  - Ensures 'line[1]', 'line[2]', 'line[3]' (qty, price, discount) are numeric.
  - Defaults discount to 0 if empty but qty and price are valid.
- Calculates the line total as: 'qty × price × (1 - discount%)', rounded to 2 decimals.
- Updates the lines grid to reflect computed values.
- Recomputes the taxes matrix:
  - Aggregates taxable bases per tax value ('line[4]').
  - Computes tax amount for each group: 'base × (tax% / 100)'.
- Rounds all tax results to 2 decimals and updates the taxes grid.
- Computes the totals matrix:
  - 'subtotal' = sum of all taxable bases
  - 'tax' = sum of all tax amounts
  - 'total' = subtotal + tax
  - Results are rounded and shown in the totals grid.
- @changes => Array of cell changes from the spreadsheet component
- @source => Source of the change event (only ''edit'' triggers processing)

6.2 ChartJS widgets for invoices

```
apps/sales/php/chartjs.php
```

This file contains helper functions that generate Chart.js-compatible datasets for visualizing invoice-related data within SaltOS4.

Each function returns an associative array with 'labels' and 'datasets', directly usable by '<chartjs>' widgets in the XML layout system.

The metrics provided include:

- Daily total invoicing
- Daily average invoice value
- Top 5 customers by total invoiced
- Count of paid vs unpaid invoices
- Average number of days to get paid

All functions rely on data from the 'app_invoices' table and assume only closed invoices ('is_closed = 1') are relevant.

6.2.1 compute_invoice_total_by_day

```
function compute_invoice_total_by_day()
```

Computes the total invoiced amount per day, for closed invoices. Useful for displaying the evolution of total income over time.

Returns a Chart.js-compatible array with labels (dates) and a single dataset (total per day).

6.2.2 compute_invoice_avg_by_day

```
function compute_invoice_avg_by_day()
```

Computes the average invoice amount per day, considering only closed invoices. This helps analyze how invoice sizes vary over time.

Returns a Chart.js-compatible array with labels (dates) and a single dataset (average amount).

6.2.3 compute_top5_customers_by_total

```
function compute_top5_customers_by_total()
```

Retrieves the top 5 customers based on total invoiced amount for closed invoices. Cuts customer names using 'intelligence_cut()' for chart readability.

Returns a Chart.js-compatible array: labels (customer names) and one dataset (total invoiced).

6.2.4 compute_invoice_paid_vs_pending

```
function compute_invoice_paid_vs_pending()
```

Counts how many closed invoices have been paid or are still unpaid. Produces a pie chart data structure separating "Paid" and "Unpaid".

Returns a Chart.js-compatible array: labels (Paid/Unpaid) and one dataset (count of invoices).

6.2.5 compute_invoice_avg_days_to_pay

```
function compute_invoice_avg_days_to_pay()
```

Calculates the average number of days it takes to get paid, based on the difference between invoice and payment dates, grouped by payment day.

Returns a Chart.js-compatible array: labels (payment dates) and one dataset (average days to pay).

6.3 Reverses the process of 'make_matrix_data()' for the 'invoices' app.

```
apps/sales/php/invoices.php
```

This function takes a previously generated matrix-style array of invoice data (grouped as lines, taxes, and totals), compares it with the current values in the database, and builds a minimal diff-like array that includes only the fields that have changed, been removed, or added.

6.3.1 unmake_matrix_data

```
function unmake_matrix_data($json, $invoice_id)
```

Reverse the matrix-encoded invoice data to a diff-style data array.

- @json => The matrix-encoded invoice data (from make_matrix_data)
- @invoice_id => The invoice ID used to retrieve the original data

Return a diff array representing only the differences between the provided data and the current state of the database.

6.3.2 Generates or completes proforma and final invoice metadata.

```
function set_proforma_invoice($json, $invoice_id)
```

This function ensures that the given invoice JSON structure has appropriate values for:

- 'proforma_code' and 'proforma_date' (if not provided or if it's a new invoice)
- 'invoice_code' and 'invoice_date' (if the invoice is being closed)
- 'is_closed' (based on database if not explicitly set)
- 'paid_date' (if the invoice is marked as paid)
- 'due_date' (if the invoice is marked as closed)

Rules:

- If no 'invoice_id' is given or 'proforma_code' is not defined, a new proforma number is generated.
- If the invoice is being closed ('is_closed' is true), a new invoice number is generated.
- If 'is_closed' is not defined in the JSON, it's looked up in the database for the given invoice ID.
- If 'is_paid' is true and no 'paid_date' is set, the current date is assigned.
- If the invoice is closed and no 'due_date' is set, the current date is assigned.

- @json => Invoice data to process and complete.
- @invoice_id => ID of the invoice, used to retrieve current status from DB.

Return the updated JSON with appropriate codes and dates filled in.

7 Tester

7.1 Tester application

```
apps/tester/js/tester.js
```

This application implements the typical features associated with a tester, showcasing functionalities such as modals, toasts, and offcanvas elements.

7.1.1 Tester object

```
saltos.tester = {};
```

This object stores all the functions used by the Tester application, providing the necessary methods for interaction and testing features.

7.1.2 Campo 8: Alert functionality

```
saltos.tester.campo8 = ()
```

This function triggers a simple alert box when called, showcasing a button-click behavior.

7.1.3 Campo 9: Modal functionality

```
saltos.tester.campo9 = ()
```

This function displays a Bootstrap modal with a title, body, and footer. The modal includes:

- Text placeholders for content.
- A dropdown menu for additional options.
- Footer buttons to accept or cancel actions, with corresponding console logs.

7.1.4 Campo 10: Offcanvas functionality

```
saltos.tester.campo10 = ()
```

This function displays a Bootstrap offcanvas element with a title and body content. The body includes:

- Text placeholders for content.
- A dropdown menu for additional options.

7.1.5 Campo 11: Toast functionality

```
saltos.tester.campo11 = ()
```

This function displays a Bootstrap toast notification with customizable attributes, including:

- A title and subtitle for context.
- Body content with dynamic timestamp information.

7.1.6 Campo 22: Open statistics page

```
saltos.tester.campo23 = ()
```

This function opens the SaltOS statistics page in a new browser tab or window.

8 Users

8.1 Login application

```
apps/users/js/login.js
```

This application implements the tipical features associated to login

8.1.1 Main object

```
saltos.login = {};
```

This object contains all SaltOS code

8.1.2 Authenticate login function

```
saltos.login.authenticate = async ()
```

This function tries to authenticate the user using the user and pass fields of the form, to do it uses the authenticate function that send data to the authtoken action

8.2 Days functions

```
apps/users/php/days.php
```

This file contain all functions needed by the days feature

8.2.1 Days to bin

```
function days2bin($days)
```

This function tries to convert the days format used by the multiselect to the string expected by the database formed by ones and zeroes to represent if a day is operative for the user or not, for example, the selection 64,32,16,8,4 is returned like from monday to friday (1111100)

- @days => the string containing the days in power of two separated by comma

8.2.2 Bin to days

```
function bin2days($days)
```

This function tries to do the reverse action that the previous function, is able to get an string like 1111100 and returns the list of all bits in decimal like 64,32,16,8,4.

- @days => the string containing the days in binary format

8.2.3 Fix for days

```
function fix4days($data)
```

This function is intended to be used as wrapper in the result of the query that contains an element called days, in the database the days is stored using the binary notation like 1111100, and for the user interface, is needed to translate this string into a decimal string like 64,32,16,8,4.

- @data => the data obtained from an execute_query, for example, they must contain an entry called days.

8.3 Groups functions

```
apps/users/php/groups.php
```

This file contain all functions needed by the groups app

8.3.1 Insert group action

```
function insert_group($data)
```

This action allow to insert registers in the database associated to the groups app and only requires the data.

- @data => the array with all data used to create the new group

8.3.2 Update group action

```
function update_group($group_id, $data)
```

This action allow to update registers in the database associated to the groups app and requires the group_id and data.

- @group_id => the group_id desired to be updated
- @data => the array with all data used to update the group

8.3.3 Delete group action

```
function delete_group($group_id)
```

This action allow to delete registers in the database associated to the groups app and only requires the group_id.

- @group_id => the group_id desired to be deleted

8.4 Matrix functions

```
apps/users/php/matrix.php
```

This file contains all the functions required by the Excel widget for handling permissions and applications in a matrix structure.

8.4.1 Create matrix data

```
function make_matrix_data($perms, $apps, $main, $user)
```

This function generates a matrix of permissions and applications, associating each application with its respective permissions and their assigned values ("Allow", "Deny", or "").

8.4.2 Create matrix cells

```
function make_matrix_cell($perms, $apps, $main, $user)
```

This function generates matrix cells with additional attributes such as column and row indices, and dropdown options for editing cell values. Read-only attributes are applied to specific cells.

8.4.3 Unmake matrix data

```
function unmake_matrix_data($perms, $apps, $main, $json)
```

This function reverses the matrix data into a more manageable format by applying permission and application mappings while validating data consistency.

8.4.4 Generate matrix permissions

```
function make_matrix_perms($table, $field, $id)
```

This function generates the complete matrix data and metadata for use in the Excel widget, including column headers, row headers, data values, and cell details.

8.5 Users functions

```
apps/users/php/users.php
```

This file contain all functions needed by the users app

8.5.1 Insert user action

```
function insert_user($data)
```

This action allow to insert registers in the database associated to the users app and only requires the data.

- @data => the array with all data used to create the new user

8.5.2 Update user action

```
function update_user($user_id, $data)
```

This action allow to update registers in the database associated to the users app and requires the user_id and data.

- @user_id => the user_id desired to be updated
- @data => the array with all data used to update the user

8.5.3 Delete user action

```
function delete_user($user_id)
```

This action allow to delete registers in the database associated to the users app and only requires the user_id.

- @group_id => the user_id desired to be deleted