# Home

# NetSuite Scripts Documentation

## Overview

This repository contains various NetSuite scripts to handle customer statements, dunning letters, and related tasks. The scripts are written using SuiteScript 2.1 and are organized into different categories like Client Scripts, Suitelets, and Map/Reduce Scripts.

## File Structure

- **common**
  - Contains common modules used across different scripts.
  - **Files:**
    - `SS_Constants.js`
    - `SS_CurrentRecord.js`
    - `SS_DataTable.js`
    - `SS_DataTableBackend.js`
    - `SS_DataTableSuitelet.js`
    - `SS_DataTableTask.js`
    - `SS_Dialog.js`
    - `SS_DunningLetters.js`
    - `SS_DunningStatements.js`
    - `SS_File.js`
    - `SS_Format.js`
    - `SS_Query.js`
    - `SS_Record.js`
    - `SS_Request.js`
    - `SS_Runtime.js`
    - `SS_Script.js`
    - `SS_Search.js`
    - `SS_String.js`
    - `SS_Task.js`
    - `SS_Transaction.js`
    - `SS_UI.js`
  - **DataTable_Template.html**: HTML template for DataTable

## Scripts

### Client Scripts

1. **SS_CS_DunningStatements.js**

- Manages dunning statements for customers.
- Dependencies: `SS_Constants` , `SS_CurrentRecord` , `SS_Dialog` , `SS_Format` , `SS_String`

2. **SS_CS_DunningLettersInv.js**

- Manages dunning letters for invoices.
- Dependencies: `SS_Constants` , `SS_CurrentRecord` , `SS_Dialog` , `SS_Format` , `SS_String`

3. **SS_CS_CustomerNotices.js**

- Manages customer notices.
- Dependencies: `SS_Constants` , `SS_CurrentRecord` , `SS_Dialog` , `SS_Format` , `SS_String`

4. **SS_CS_DunningLetters.js**

- Manages dunning letters for customers.
- Dependencies: `SS_Constants` , `SS_CurrentRecord` , `SS_Dialog` , `SS_Format` , `SS_String`

### Suitelets

1. **SS_SL_DataTables.js**

- Renders the DataTable for customer statements.
- Dependencies: `SS_DataTableSuitelet`

2. **SS_SL_DataTables_Backend.js**

- Processes backend operations for DataTable.
- Dependencies: `SS_DataTableBackend`

3. **R-IT_FRL_SL_DunningStatements.js**

- Handles dunning statements through a Suitelet.
- Dependencies: `SS_Constants` , `SS_DataTableBackend` , `SS_Search` , `SS_Record`

### Library Scripts

1. **SS_Lib_CustomerStatements.js**

- Manages customer statements, builds search filters, creates statement tasks.
- Dependencies: `SS_Constants` , `SS_File` , `SS_Record` , `SS_Script` , `SS_Search` , `SS_String` , `SS_UI` , `SS_Url` , `SS_Transaction`

### Map/Reduce Scripts

1. **SS_MR_DataTableTask.js**

- Handles data table tasks using Map/Reduce.
- Dependencies: `N/record` , `N/search` , `N/runtime` , `N/format` , `N/render` , `N/email` , `N/xml` , `N/file` , `SS_Constants`

### Script Files

- **SS_CS_DunningStatements.js**

- SS_CS_DunningLettersInv.js
- SS_CS_CustomerNotices.js
- SS_CS_DunningLetters.js

MODULE

# SS_CS_CustomerNotices

This script is a NetSuite Client Script designed to handle customer notices. The key functionalities include: Initialization (pageInit): Sets up the page, applies filters to retrieve data from the backend, and initializes the DataTable for displaying customer data. Applying Filters (applyFilters): Collects filter values from the current record, sends a request to fetch data based on these filters, and displays the data in a DataTable. Initializing DataTable (initDataTable): Configures and initializes the DataTable with the fetched data, including setting up columns, data rendering, and selection checkboxes. Saving Records (saveRecord): Validates the selected rows, confirms the action with the user, and sends a request to generate and send customer statements. Trigger Buttons (triggerbtn_II, triggerbtn_III, triggerbtn_IV, triggerbtn_V): Handle different levels of dunning letters by getting selected IDs and sending appropriate requests based on the chosen level. Helper Functions: Various utility functions to manage data selection, URL building, DOM manipulation, and sending backend requests.

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 1

### Requires

- module:SS_Constants
- module:SS_CurrentRecord
- module:SS_Dialog
- module:SS_Format
- module:SS_String



# Members

INNER CONSTANT

Type: `string`

### TABLE_ID

ID of the DataTable element

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 58

INNER

Type: `string`

### TITLE

Title of the Suitelet

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 66

# Methods

INNER

### applyFilters() → {void}

Collects filter values from the current record, sends a request to fetch data based on these filters, and displays the data in a DataTable.

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 74

Returns: Type: `void`

### Example

```
SS_CS_CustomerNotices.applyFilters();
```

INNER

### buildUrlWithParams(options) → {string}

Builds a URL with the given parameters

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | | |
| action | `string` | Action to be performed |
| type | `string` | Type of Suitelet |
| urlParams | `Array` | Array of URL parameters |

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 118

Returns: Type: `string`

### Example

```
buildUrlWithParams({ action: 'search', type: 'customer_notices', urlParams: ['param1=value1', 'param2=value2'] });
```

INNER

### getById(id) → {Element}

Get an element by its ID

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| id | string | ID of the element |

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 145

Returns: Type: Element

INNER

### getSelectedIds() → {Array|Array}

Get the selected row IDs from the DataTable

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 156

Returns: Type: Array

selectedIds Type: Array

INNER

### initDataTable() → {void}

Configures and initializes the DataTable with the fetched data

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 183

Returns: Type: void

**Example**

```
SS_CS_CustomerNotices.initDataTable();
```

INNER

### pageInit(context) → {void}

Sets up the page, applies filters to retrieve data from the backend, and initializes the DataTable for displaying customer data.

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| context | Object | Current context |

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 250

Returns: Type: void

**Example**

```
SS_CS_CustomerNotices.pageInit(context);
```

INNER

### query(selector) → {Element}

Get an element by its selector using document.querySelector

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| selector | string | Selector of the element |

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 281

Returns: Type: Element

INNER

### saveRecord(context) → {boolean}

Validates the selected rows, confirms the action with the user, and sends a request to generate and send customer statements.

**Parameters:**

| Name | Type | Description |
|---|---|---|
| context | Object | Current context |

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 292

Returns: Type: boolean

INNER

### sendDunningLetterRequest(list) → {void}

Send a request to generate and send dunning letters

**Parameters:**

| Name | Type | Description |
|---|---|---|
| list | Array | List of selected IDs |

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 337

Returns: Type: void

INNER

### sendRequest(options) → {void}

Send a request to the backend

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | Request options |

View Source DataTableSuitelets/SS_CS_CustomerNotices.js, line 376

Returns: Type: void

MODULE

# SS_CS_DunningLetters

This NetSuite Client Script manages the process of sending customer dunning letters based on user-selected filters and options. Here's a brief overview of the script:
Initialization (pageInit): Initializes the page, sets up the table layout, and applies filters to fetch initial data.
Applying Filters (applyFilters): Gathers filter values from the current record, sends a request to the backend, and displays the filtered data in a DataTable.
DataTable Initialization (initDataTable): Configures and initializes the DataTable with the retrieved data, including setting up columns, formatting, and checkboxes for row selection.
Saving Records (saveRecord): Validates selected rows, confirms the action with the user, and sends a request to generate and send dunning letters for the selected rows.
Trigger Buttons (triggerbtn_II, triggerbtn_III, triggerbtn_IV, triggerbtn_V): Handle sending dunning letters at different levels by retrieving selected IDs and sending appropriate requests based on the chosen level.
Helper Functions: Various utility functions for URL building (buildUrlWithParams), DOM element retrieval (getById), querying elements (query), sending requests (sendRequest), and managing selected IDs (getSelectedIds, getAllDataSelectedIdsByLevel).
Dialogs and Alerts: Utilizes dialogs to alert users and confirm actions before proceeding with sending dunning letters.

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 1

| Returns: | Public methods | Type: `Object` |
|---|---|---|
| | applyFilters | Type: `function` |
| | pageInit | Type: `function` |
| | saveRecord | Type: `function` |
| | triggerbtn_II | Type: `function` |
| | triggerbtn_III | Type: `function` |
| | triggerbtn_IV | Type: `function` |
| | triggerbtn_V | Type: `function` |

### Requires

- module:SS_Constants
- module:SS_CurrentRecord
- module:SS_Dialog
- module:SS_Format
- module:SS_String

## Members

INNER

Type: `string`

### TITLE

Title of the Suitelet

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 70

INNER

Type: `string`

### Table_ID

ID of the DataTable element

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 63

## Methods

INNER

### applyFilters() → {void}

Gathers filter values from the current record, sends a request to the backend, and displays the filtered data in a DataTable.

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 78

Returns: Type: `void`

### Example

```
SS_CS_DunningLetters.applyFilters();
```

INNER

### buildUrlWithParams(options) → {string}

Builds a URL with the specified parameters

### Parameters:

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| options | Object | Options for building the URL+ |
| action | string | Action to be performed |
| type | string | Type of Suitelet |
| urlParams | Array | Additional URL parameters |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 124

Returns: - The built URL Type: `string`

INNER

### getAllDataSelectedIdsByLevel(level) → {Array}

Retrieves all selected IDs by level

**Parameters:**

| Name | Type | Description |
|---|---|---|
| level | string | The level of the dunning letter |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 557

Returns: - An array of selected IDs Type: `Array`

**Example**

```
getAllDataSelectedIdsByLevel(level);
```

INNER

### getById(id) → {Element}

Retrieves an element by its ID

**Parameters:**

| Name | Type | Description |
|---|---|---|
| id | string | ID of the element to retrieve |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 149

Returns: - The element with the specified ID Type: `Element`

**Example**

```
getById('custpage_table');
```

INNER

### getSelectedIds() → {Array}

Retrieves the selected IDs from the DataTable

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 161

Returns: - An array of selected IDs Type: `Array`

**Example**

```
getSelectedIds();
```

INNER

### initDataTable() → {void}

Configures and initializes the DataTable with the retrieved data

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 188

Returns: Type: `void`

**Example**

```
initDataTable();
```

INNER

### pageInit(context) → {void}

Initializes the page, sets up the table layout, and applies filters to fetch initial data

**Parameters:**

| Name | Type | Description |
|---|---|---|
| context | Object | The SuiteScript context object |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 254

Returns: Type: void

**Example**

```
pageInit(context);
```

INNER

### query(selector) → {Element}

Queries an element by its selector

**Parameters:**

| Name | Type | Description |
|---|---|---|
| selector | string | The selector to query |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 284

Returns: - The element with the specified selector Type: Element

INNER

### saveRecord(context) → {boolean}

Validates selected rows, confirms the action with the user, and sends a request to generate and send dunning letters for the selected rows

**Parameters:**

| Name | Type | Description |
|---|---|---|
| context | Object | The SuiteScript context object |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 295

Returns: - Returns false to prevent the record from being saved Type: boolean

**Example**

```
saveRecord(context);
```

INNER

### sendDunningLetterRequest(list) → {void}

Sends a request to generate and send dunning letters for the selected rows

**Parameters:**

| Name | Type | Description |
|---|---|---|
| list | Array | An array of selected IDs |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 343

Returns: Type: void

INNER

### sendRequest(options) → {void}

Sends a request to the backend

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | Options for sending the request |

| | | |
|---|---|---|
| action | `string` | Action to be performed |
| requestParams | `Object` | Request parameters |
| success | `function` | Success callback function |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 383

Returns: Type: `void`

**Example**

```
sendRequest({ action: 'search', requestParams: { method: 'get' }, success: (data) => { console.log(data); } });
```

INNER

## triggerbtn_II(context) → {boolean}

Handle sending dunning letters at LEVEL 1

**Parameters:**

| Name | Type | Description |
|---|---|---|
| context | `Object` | The SuiteScript context object |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 419

Returns: - Returns false to prevent the record from being saved Type: `boolean`

**Example**

```
triggerbtn_II(context);
```

INNER

## triggerbtn_III(context) → {boolean}

Handle sending dunning letters at LEVEL 2

**Parameters:**

| Name | Type | Description |
|---|---|---|
| context | `Object` | The SuiteScript context object |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 458

Returns: - Returns false to prevent the record from being saved Type: `boolean`

**Example**

```
triggerbtn_III(context);
```

INNER

## triggerbtn_IV(context) → {boolean}

Handle sending dunning letters at LEVEL 3

**Parameters:**

| Name | Type | Description |
|---|---|---|
| context | `Object` | The SuiteScript context object |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 491

Returns: - Returns false to prevent the record from being saved Type: `boolean`

**Example**

```
triggerbtn_IV(context);
```

# INNER

## triggerbtn_V(context) → {boolean}

Handle sending dunning letters at LEVEL 4

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| context | Object | The SuiteScript context object |

View Source DataTableSuitelets/SS_CS_DunningLetters.js, line 524

Returns: - Returns false to prevent the record from being saved Type: boolean

### Example

```
triggerbtn_V(context);
```

MODULE

# SS_CS_DunningLettersInv

Client script for the Dunning Letters/Invoices page

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 1

### Requires

- module:SS_Constants
- module:SS_CurrentRecord
- module:SS_Dialog
- module:SS_Format
- module:SS_String

# Members

INNER CONSTANT

Type: `string`

### TABLE_ID

ID of the DataTable element

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 35

INNER

Type: `string`

### TITLE

Title of the Suitelet

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 43

# Methods

INNER

### applyFilters() → {void}

Collects filter values from the current record, sends a request to fetch data based on these filters, and displays the data in a DataTable.

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 52

Returns: Type: `void`

### Example

```
SS_CS_CustomerNotices.applyFilters();
```

INNER

### buildUrlWithParams(options) → {string}

Builds a URL with the given parameters

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | | |
| action | `string` | Action to be performed |
| type | `string` | Type of Suitelet |
| urlParams | `Array` | Array of URL parameters |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 97

Returns: Type: `string`

### Example

```
buildUrlWithParams({ action: 'search', type: 'customer_notices', urlParams: ['param1=value1', 'param2=value2'] });
```

INNER

### getAllDataSelectedIdsByLevel(level) → {Array}

Get all selected rows by level

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| `level` | | the level of the dunning letter |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 529

Returns: selectedIds Type: `Array`

INNER

### getById(id) → {Element}

Get an element by its ID

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| `id` | `string` | ID of the element |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 123

Returns: Type: `Element`

INNER

### getSelectedIds() → {Array|Array}

Get the selected row IDs from the DataTable

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 133

Returns: Type: `Array`

selectedIds Type: `Array`

INNER

### initDataTable() → {void}

Configures and initializes the DataTable with the fetched data

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 160

Returns: Type: `void`

**Example**

```
SS_CS_CustomerNotices.initDataTable();
```

INNER

### pageInit(context) → {void}

Sets up the page, applies filters to retrieve data from the backend, and initializes the DataTable for displaying customer data.

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| `context` | `Object` | Current context |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 231

Returns: Type: `void`

**Example**

```
SS_CS_CustomerNotices.pageInit(context);
```

INNER

### query(selector) → {Element}

Get an element by its selector using document.querySelector

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| selector | string | Selector of the element |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 262

Returns: Type: Element

INNER

### saveRecord(context) → {boolean}

Validates the selected rows, confirms the action with the user, and sends a request to generate and send customer statements.

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| context | Object | Current context |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 272

Returns: Type: boolean

INNER

### sendDunningLetterRequest(list) → {void}

Send a request to generate and send dunning letters

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| list | Array | List of selected IDs |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 323

Returns: Type: void

INNER

### sendRequest(options) → {void}

Send a request to the backend

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| options | Object | Request options |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 362

Returns: Type: void

INNER

### triggerbtn_II(context) → {boolean}

Trigger the action for the LEVEL 1 button

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| context | | |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 392

Returns: Type: boolean

**Example**

```
SS_CS_DunningLettersInv.triggerbtn_II(context);
```

INNER

### triggerbtn_III(context) → {boolean}

Trigger the action for the LEVEL 2 button

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| context | | |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 433

Returns: Type: `boolean`

INNER

### triggerbtn_IV(context) → {boolean}

Trigger the action for the LEVEL 3 button

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| context | | |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 464

Returns: Type: `boolean`

INNER

### triggerbtn_V(context) → {boolean}

Trigger the action for the LEVEL 4 button

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| context | | |

View Source DataTableSuitelets/SS_CS_DunningLettersInv.js, line 497

Returns: Type: `boolean`

MODULE

# SS_CS_DunningStatements

Client script for sending customer statements

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 1

### Requires

- module:SS_Constants
- module:SS_CurrentRecord
- module:SS_Dialog
- module:SS_Format
- module:SS_String

## Members

INNER CONSTANT

Type: `string`

### TABLE_ID

ID of the DataTable element

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 32

INNER

Type: `string`

### TITLE

Title of the Suitelet

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 40

## Methods

INNER

### applyFilters() → {void}

Collects filter values from the current record, sends a request to fetch data based on these filters, and displays the data in a DataTable.

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 49

Returns: Type: `void`

### Example

```
SS_CS_CustomerNotices.applyFilters();
```

INNER

### buildUrlWithParams(options) → {string}

Builds a URL with the given parameters

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | | |
| action | `string` | Action to be performed |
| type | `string` | Type of Suitelet |
| urlParams | `Array` | Array of URL parameters |

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 94

Returns: Type: `string`

### Example

```
buildUrlWithParams({ action: 'search', type: 'customer_notices', urlParams: ['param1=value1', 'param2=value2'] });
```

INNER

### getById(id) → {Element}

Get an element by its ID

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| id | string | ID of the element |

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 121

Returns: Type: Element

INNER

### getSelectedIds() → {Array|Array}

Get the selected row IDs from the DataTable

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 131

Returns: Type: Array

selectedIds Type: Array

# INNER

### initDataTable() → {void}

Configures and initializes the DataTable with the fetched data

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 158

Returns: Type: void

**Example**

```
SS_CS_CustomerNotices.initDataTable();
```

INNER

### pageInit(context) → {void}

Sets up the page, applies filters to retrieve data from the backend, and initializes the DataTable for displaying customer data.

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| context | Object | Current context |

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 225

Returns: Type: void

**Example**

```
SS_CS_CustomerNotices.pageInit(context);
```

INNER

### query(selector) → {Element}

Get an element by its selector using document.querySelector

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| selector | string | Selector of the element |

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 255

Returns: Type: Element

# INNER

### saveRecord(context) → {boolean}

Validates the selected rows, confirms the action with the user, and sends a request to generate and send customer statements.

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| context | Object | Current context |

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 265

Returns: Type: boolean

INNER

### sendDunningLetterRequest(list) → {void}

Send a request to generate and send dunning letters

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| list | Array | List of selected IDs |

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 306

Returns: Type: void

INNER

### sendRequest(options) → {void}

Send a request to the backend

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| options | Object | Request options |

View Source DataTableSuitelets/SS_CS_DunningStatements.js, line 345

Returns: Type: void

MODULE

# SS_Constants

The module wraps all constants used within the dunning logic related to transactions, custom records, forms, and other configurations.

View Source common/SS_Constants.js, line 1

Returns:

| | |
|---|---|
| CustomLists | Type: `object` |
| CustomRecords | Type: `object` |
| Entity | Type: `object` |
| Forms | Type: `object` |
| NS5_Categories | Type: `object` |
| RequestParameters | Type: `object` |
| ScriptParameters | Type: `object` |
| Scripts | Type: `object` |
| Templates - Datatable Templates for the Suitelet {DataTable_Template.html} | Type: `object` |

### Requires

- module:SS_Transaction



# Members

INNER

Type: `object`

## CUSTOM_LISTS

Constant that organizes and stores various custom list values and their associated id used in a Dunning project.

### Properties:

| Name | Type | Description |
|---|---|---|
| BankingMethod | `object` | Banking Method Custom List |
| DataTableSuiteletTypes | `object` | DataTable Suitelet Types |
| DataTableTaskStatus | `object` | DataTable Task Status |
| DunningLevels | `object` | Dunning Levels |
| TaxClassification | `string` | Tax Classification Custom List |
| TypeOfAccount | `string` | Type of Account Custom List |

View Source common/SS_Constants.js, line 77

### Example

```
const CUSTOM_LISTS = {
            BankingMethod: 'customlist_vr_bankdetailmethod',
            DataTableSuiteletTypes: {
                DUNNING_STATEMENTS: '1',
                DUNNING_LETTERS: '101',
            },
            DataTableTaskStatus: {
                PENDING: '1',
                IN_PROGRESS: '2',
                COMPLETED: '3',
                ...
            },
            DunningLevels: {
                Id: 'customlist_ss_dunning_levels',
                Values: [
                    { id: '1', from: 0, to: 0 },
                    { id: '2', from: 1, to: 30 },
                    ...
                ]
            },
            TaxClassification: 'customlist_vr_taxclassificationlist',
            TypeOfAccount: 'customlist_vr_typeofaccount'
        };
```

INNER

Type: `object`

## CUSTOM_RECORDS

Constant that organizes and stores various custom record values and their associated id used in a Dunning project.

### Properties:

| Name | Type | Description |
| --- | --- | --- |
| DataTableTask | `object` | DataTable Task Custom Record |
| VendorRequest | `object` | Vendor Request Custom Record |

View Source common/SS_Constants.js, line 142

### Example

```
const CUSTOM_RECORDS = {
          DataTableTask: {
          Id: 'customrecord_ss_dt_task',
          Fields: {
          DATA                        : 'custrecord_ss_dt_task_data',
          ERRORS                      : 'custrecord_ss_dt_task_errors',
          EXPECTED                    : 'custrecord_ss_dt_task_expected',
          GENERATED                   : 'custrecord_ss_dt_task_generated',
          STATUS                      : 'custrecord_ss_dt_task_status',
          ...
          VendorRequest: {
          Id: 'customrecord_vendor_request',
          Fields: {
          ACCOUNT_HOLDER_FULL_NAME    : 'custrecord_vr_fullnameaccountholder',
          ACCOUNT_NUMBER              : 'custrecord_vr_accountnumber',
          ACCOUNT_NUMBER_CBU          : 'custrecord_vr_cbuaccountnumber',
          ...
```

INNER

Type: `object`

## ENTITY

Declare Entity Object and assign Custom Fields to it

### Properties:

| Name | Type | Description |
| --- | --- | --- |
| CustomFields | `object` | Custom Fields for the Entity |

View Source common/SS_Constants.js, line 34

### Example

```
let ENTITY = {};
ENTITY.CustomFields = {
    CAS: 'custentity_cas'
}
```

INNER

Type: `object`

## FORMS

Constant that organizes and stores various form values and their associated id used in a Dunning project.

### Properties:

| Name | Type | Description |
| --- | --- | --- |
| DUNNING_STATEMENTS | `object` | Dunning Statements Form |
| DUNNING_LETTERS | `object` | Dunning Letters Form |

View Source common/SS_Constants.js, line 301

### Example

```
const FORMS = {
   DUNNING_STATEMENTS: {
          Title: 'Send Customer Statements',
          Buttons: [
          {
          id: 'custpage_btn_apply_filters',
          label: 'Apply Filters',
          functionName: 'applyFilters'
          },
          {
          label: 'Send Statements',
          submit: true
          }
          ],
          ClientScriptFile: 'SS_CS_DunningStatements.js',
          FieldGroups: [
          {
          id: 'custpage_filters',
          ...
          },
  DUNNING_LETTERS: {
                Title: 'Dunning Letters',
                Buttons: [
                    {
                        id: 'custpage_btn_apply_filters',
                        label: 'Apply Filters',
                        functionName: 'applyFilters'
                    },
                    {
                        id: 'custpage_btn_level1',
                        label: 'Send Level 1',
                        functionName: 'triggerbtn_II'
                    },
```

```
{...
```

INNER

Type: `object`

## NS5_CATEGORIES

Constant that organizes and stores various NS5 categories.

### Properties:

| Name | Type | Description |
| --- | --- | --- |
| ON_BOARD | `string` | On Board |
| OFF_SITE | `string` | Off Site |
| SUPPLIER | `string` | Supplier |

View Source common/SS_Constants.js, line 618

### Example

```
const NS5_CATEGORIES = {
          ON_BOARD:                    '1',
          OFF_SITE:                    '2',
          SUPPLIER:                    '3'
          }
```

INNER

Type: `object`

## REQUEST_PARAMETERS

Constant that organizes and stores various request parameters used in a Dunning project.

### Properties:

| Name | Type | Description |
| --- | --- | --- |
| DATATABLE_BACKEND | `Array.<string>` | DataTable Backend Request Parameters |
| DUNNING_LETTERS | `Array.<string>` | Dunning Letters Request Parameters |
| DUNNING_STATEMENTS | `Array.<string>` | Dunning Statements Request Parameters |

View Source common/SS_Constants.js, line 641

### Example

```
const REQUEST_PARAMETERS = {
   DATATABLE_BACKEND: [
   'action',
   'sltype'
   ],
   DUNNING_STATEMENTS: [
   'dosearch',
   'subsidiary',
   ...
```

INNER

Type: `object`

## SCRIPTS

Constant that organizes and stores various script type and their associated id used in a Dunning project.

### Properties:

| Name | Type | Description |
| --- | --- | --- |
| DataTableSuitelet | `object` | DataTable Suitelet Scripts |
| DataTableTaskMapReduce | `object` | DataTable Task Map Reduce Scripts |
| DunningStatementsMapReduce | `object` | Dunning Statements Map Reduce Scripts |
| DunningLettersMapReduce | `object` | Dunning Letters Map Reduce Scripts |

View Source common/SS_Constants.js, line 685

### Example

```
const SCRIPTS = {
  DataTableSuitelet: {
   BACKEND: {
   scriptId: 'customscript_ss_sl_dt_backend',
   },
   DUNNING_STATEMENTS: {
   deploymentId: 'customdeploy_ss_sl_dt_backend_dunstm'
   },
   DUNNING_LETTERS: {
   ...
```

INNER

Type: `object`

## SCRIPT_PARAMETERS

Constant that organizes and stores various script record parameters used in a Dunning project.

### Properties:

| Name | Type | Description |
|---|---|---|
| DunningStatementMapReduce | `object` | Dunning Statement Map Reduce Script Parameters |
| HtmlTemplateFile | `string` | HTML Template File |
| DataTableSuiteletBackend | `object` | DataTable Suitelet Backend Script Parameters |
| DataTableTaskMapReduce | `object` | DataTable Task Map Reduce Script Parameters |
| CustomerStatementTaskSearch | `string` | Customer Statement Task Search |
| FileUploadFolder | `string` | File Upload Folder |
| DataTableSuitelet | `string` | DataTable Suitelet Script Parameters |

View Source common/SS_Constants.js, line 728

### Example

```
const SCRIPT_PARAMETERS = {
  DunningStatements: {
  StatementSearchId: 'custscript_ss_cs_search',
  TaskSearchId: 'custscript_ss_cs_tasks_search'
  },
  CustomerStatementTaskSearch: '',
  FileUploadFolder: 'custscript_ss_sl_ven_onboard_form_folder',
  HtmlTemplateFile: 'custscript_ss_sl_ven_onboard_form_html',
  DataTableTaskRecord: 'custscript_ss_mr_dt_task_record',
  DataTableSuitelet: {
  Type: 'custscript_ss_sl_dt_type'
  ...
```

INNER

Type: `object`

## TRANSACTION

Declare Transaction Object and assign custom modules {./common/SS_Transaction},Also define prorety CustomTypes , CustomFields, and CustomSublistFields

### Properties:

| Name | Type | Description |
|---|---|---|
| CustomTypes | `object` | Custom Types for the Transaction |
| CustomFields | `object` | Custom Fields for the Transaction |
| CustomSublistFields | `object` | Custom Sublist Fields for the Transaction |

View Source common/SS_Constants.js, line 52

### Example

```
let TRANSACTION = SS_Transaction;
        TRANSACTION.CustomTypes = {};
        TRANSACTION.CustomFields = {
            LAST_COMMUNICATION_DATE:            'custbody_ss_last_statement_comm_date'
        };
        TRANSACTION.CustomSublistFields = {};
}
```

MODULE

# SS_CurrentRecord

Wrapper module for N/currentRecord

See:

- SS_CurrentRecord

View Source common/SS_CurrentRecord.js, line 1

Returns: get function from SS_CurrentRecord module Type: `Object`

### Requires

- module:NS_CurrentRecord
- module:SS_Record

# Methods

INNER

### `get() → {SS_Record}`

Retrieves the current record

View Source common/SS_CurrentRecord.js, line 30

Throws: MISSING_REQD_ARGUMENT if options is undefined Type: `error.SuiteScriptError`

MODULE

# SS_DataTable

Wrapper module for jQuery DataTables

View Source common/SS_DataTable.js, line 1

## Classes

DataTable

# Members

INNER

Type: `string`

### MODULE

Module name for logging

View Source common/SS_DataTable.js, line 18

## Example

```
MODULE = `outside|DataTable`;
log.debug({ title: MODULE, details: 'message' });
```

# Methods

MODULE

# SS_DataTableBackend

SS_DataTableBackend module for handling DataTable Suitelet backend processing

View Source common/SS_DataTableBackend.js, line 1

### Requires

- module:SS_Constants
- module:SS_File
- module:SS_Script
- module:SS_Search
- module:SS_String
- module:SS_UI
- module:SS_Url
- module:SS_DunningStatements
- module:SS_DunningLetters



# Members

INNER CONSTANT

Type: `Object`

## HANDLERS

constant object that contains the handlers for the DataTable Suitelet send actions from SS_DunningStatements and SS_DunningLetters module

View Source common/SS_DataTableBackend.js, line 70

### Example

```
const HANDLERS = {
   'dunning_statement': SS_DunningStatements.send,
   'dunning_letter': SS_DunningLetters.send
   }
```

INNER CONSTANT

Type: `string`

## MODULE

constant Module name for logging

View Source common/SS_DataTableBackend.js, line 48

### Example

```
MODULE = `outside|DataTableBackend`;
log.debug({ title: MODULE, details: 'message' });
```

INNER CONSTANT

Type: `Object`

## TYPES

constant declaration that take DataTable suitelet types create on the custom record for processing from SS_Constants module

See:

- https://5678674-sb2.app.netsuite.com/app/common/search/ubersearchresults.nl?quicksearch=T&searchtype=Uber&frame=be&Uber_NAMEtype=KEYWORDSTARTSWITH&Uber_NAME=SS%20%7C%20Data%20Table%20Suitelet%20Types%09

View Source common/SS_DataTableBackend.js, line 59

### Example

```
const TYPES = SS_Constants.CustomLists.DataTableSuiteletTypes;
```

# Methods

INNER

## buildSearchFilters(options) → {Array}

function that build the search filters for the DataTable Suitelet

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | `Object` | The options object for the DataTable Suitelet |
| parameters | `Object` | The parameters for the DataTable Suitelet |

View Source common/SS_DataTableBackend.js, line 89

Returns: filters - The filters for the DataTable Suitelet Type: `Array`

**Example**

```
const buildSearchFilters = (options) => {
            const TITLE = `${MODULE}.BuildSearchFilters`;
            let { parameters } = options;
            let filters = [];

            switch (parameters.type) {
                case TYPES.DUNNING_STATEMENTS: {
                    filters = SS_DunningStatements.buildSearchFilters(parameters);
                    break;
                }
            }

            return filters;
        };
```

INNER

### getInProgressTasks(options) → {Array}

function that get the in progress tasks for the DataTable Suitelet

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | `Object` | The options object for the DataTable Suitelet |
| searchId | `string` | The searchId for the DataTable Suitelet |
| type | `string` | The type for the DataTable Suitelet |

View Source common/SS_DataTableBackend.js, line 126

Returns: inProgress - The inProgress for the DataTable Suitelet Type: `Array`

INNER

### getTableData(options) → {Object}

function that get the table data for the DataTable Suitelet

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | `Object` | The options object for the DataTable Suitelet |
| query | `Object` | The query for the DataTable Suitelet |
| searchId | `string` | The searchId for the DataTable Suitelet |

View Source common/SS_DataTableBackend.js, line 169

Returns: data - The data for the DataTable Suitelet Type: `Object`

INNER

### process(options) → {Object}

function that process the DataTable backend Suitelet for the DataTable Suitelet

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | `Object` | The options object for the DataTable Suitelet |
| request | `Object` | The request for the DataTable Suitelet |
| response | `Object` | The response for the DataTable Suitelet |

View Source common/SS_DataTableBackend.js, line 222

Returns: data - The data for the DataTable Suitelet Type: `Object`

INNER

### readQueryParameters(options) → {Object}

function that read the query parameters for the DataTable Suitelet

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| options | Object | The options object for the DataTable Suitelet |
| request | Object | The request for the DataTable Suitelet |
| response | Object | The response for the DataTable Suitelet |

View Source common/SS_DataTableBackend.js, line 277

Returns: returnValues - The returnValues for the DataTable Suitelet Type: Object

INNER

### readScriptParameters(options) → {Object}

function that read the script parameters for the DataTable Suitelet

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| options | Object | The options object for the DataTable Suitelet |
| response | Object | The response for the DataTable Suitelet |

View Source common/SS_DataTableBackend.js, line 324

Returns: scriptParams - The scriptParams for the DataTable Suitelet Type: Object

INNER

### search(options) → {Object}

function that search for the DataTable Suitelet

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| options | Object | The options object for the DataTable Suitelet |
| body | Object | The body for the DataTable Suitelet |
| query | Object | The query for the DataTable Suitelet |
| response | Object | The response for the DataTable Suitelet |
| script | Object | The script for the DataTable Suitelet |

View Source common/SS_DataTableBackend.js, line 374

Returns: data - The data for the DataTable Suitelet Type: Object

INNER

### sendError(options) → {Object}

function that send error for the DataTable Suitelet

**Parameters:**

| Name | Type | Description |
| --- | --- | --- |
| options | Object | The options object for the DataTable Suitelet |
| error | Object | The error for the DataTable Suitelet |
| response | Object | The response for the DataTable Suitelet |
| title | Object | The title for the DataTable Suitelet |

View Source common/SS_DataTableBackend.js, line 411

Returns: data - The data for the DataTable Suitelet Type: Object

MODULE

# SS_DataTableSuitelet

DataTable Suitelet module for rendering a DataTable Suitelet

View Source common/SS_DataTableSuitelet.js, line 1

### Requires

- module:NS_Url
- module:SS_Constants
- module:SS_File
- module:SS_Script
- module:SS_UI
- module:SS_Url
- module:SS_DunningStatements



# Members

INNER

Type: `string`

### MODULE

Module name for logging

View Source common/SS_DataTableSuitelet.js, line 39

INNER

Type: `Object`

### TYPES

Object containing the Suitelet types

View Source common/SS_DataTableSuitelet.js, line 47

# Methods

INNER

### getDefaultUrlValues(options) → {Object}

Retrieves the default URL values for the Suitelet

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | `Object` | The options object for the Suitelet |
| type | `int` | The type of Suitelet |

View Source common/SS_DataTableSuitelet.js, line 83

Returns: The default URL values for the Suitelet — Type: `Object`

**Example**

```
getDefaultUrlValues({ type: 101 });
```

INNER

### getRenderedForm(options) → {nlobjForm}

Retrieves the rendered form for the Suitelet

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | `Object` | The options object for the Suitelet |
| renderOptions | `Object` | The render options for the Suitelet |
| type | `string` | The type of Suitelet |

View Source common/SS_DataTableSuitelet.js, line 125

Returns: The rendered form for the Suitelet — Type: `nlobjForm`

INNER

### readScriptParameters() → {Object}

Reads the script parameters for the Suitelet

View Source common/SS_DataTableSuitelet.js, line 160

Throws: MISSING_REQD_ARGUMENT if a required parameter is missing — Type: `error.SuiteScriptError`

Returns: The script parameters for the Suitelet — Type: `Object`

INNER

### render(options) → {nlobjForm}

Renders the Suitelet

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | `Object` | The options object for the Suitelet |
| request | `Object` | The request object for the Suitelet |
| response | `Object` | The response object for the Suitelet |

View Source common/SS_DataTableSuitelet.js, line 182

Returns: The rendered form for the Suitelet — Type: `nlobjForm`

**Example**

```
render({
   request: request,
   response: response
   });
```

INNER

### writeDataTable(options)

Writes the data table to the Suitelet form

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | `Object` | The options object for the data table |
| form | `nlobjForm` | The form object for the Suitelet |
| template | `string` | The template for the data table |

View Source common/SS_DataTableSuitelet.js, line 246

MODULE

# SS_DataTableTask

DataTable Task module for creating a DataTable Task record

View Source common/SS_DataTableTask.js, line 1

### Requires

- module:SS_Constants
- module:SS_Record
- module:SS_Task
- module:SS_Url

## Members

INNER

Type: `Object`

### FIELDS

View Source common/SS_DataTableTask.js, line 46

#### Example

```
{{filter: boolean, container: string, id: string, label: string, source: string, type: string}|{filter: boolean, container: string, id: string, label: string, source: string, typ
```

INNER CONSTANT

Type: `string`

### MODULE

Module name for logging

View Source common/SS_DataTableTask.js, line 29

INNER

Type: `Object`

### TASK

View Source common/SS_DataTableTask.js, line 38

#### Example

```
{{Fields: {STATUS: string, ERRORS: string, DATA: string, EXPECTED: string, GENERATED: string, TYPE: string}, Id: string}}
```

## Methods

INNER

### create(options) → {Object}

Create a DataTable Task record

#### Parameters:

| Name | Type | Description |
|---|---|---|
| options | `Object` | |
| data | `Object` | |
| type | `string` | |

View Source common/SS_DataTableTask.js, line 53

Returns: Type: `Object`

#### Example

```
{status: boolean, data: string, url: string}
```

MODULE

# SS_Dialog

Wrapper module for N/ui/dialog

View Source common/SS_Dialog.js, line 1

| Returns: | | |
|---|---|---|
| | NS_Dialog | Type: `Object` |
| | NS_Dialog.alert | Type: `function` |
| | NS_Dialog.confirm | Type: `function` |
| | NS_Dialog.create | Type: `function` |

## Requires

- module:N/ui/dialog

MODULE

# SS_DunningLetters

Dunning Letters module for rendering and sending dunning letters

View Source common/SS_DunningLetters.js, line 1

### Requires

- module:SS_Constants
- module:SS_DataTableTask
- module:SS_Search
- module:SS_Task
- module:SS_UI



# Members

INNER CONSTANT

### FORM

Type: `Object`

Form configuration

View Source common/SS_DunningLetters.js, line 47

INNER CONSTANT

### MODULE

Type: `string`

Module name for logging

View Source common/SS_DunningLetters.js, line 38

INNER CONSTANT

### OPERATOR

Type: `Object`

Search operators

View Source common/SS_DunningLetters.js, line 70

INNER CONSTANT

### REQUEST_FIELDS

Type: `Object`

Request fields

View Source common/SS_DunningLetters.js, line 62

INNER CONSTANT

### REQUEST_PARAMS

Type: `Object`

Request parameters

View Source common/SS_DunningLetters.js, line 55

INNER CONSTANT

### REQUEST_TYPE

Type: `Object`

Request types

View Source common/SS_DunningLetters.js, line 86

INNER CONSTANT

### TRANSACTION

Type: `Object`

Transaction fields

View Source common/SS_DunningLetters.js, line 78

# Methods

INNER

### buildSearchFilters(options) → {Array}

Build search filters

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | | |
| balance | string | |
| cas | string | |
| condition | string | |
| customer | string | |
| lastcomm | string | |
| levels | string | |
| openbalanceusd | string | |
| subsidiary | string | |

View Source common/SS_DunningLetters.js, line 94

Returns: Type: Array

INNER

### render(options) → {Object}

Render the dunning letter form

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | |
| request | Object | |
| response | Object | |
| defaultValues | Object | |

View Source common/SS_DunningLetters.js, line 247

Returns: Type: Object

INNER

### send(options) → {void}

Send dunning letters

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | |
| body | string | |
| query | Object | |
| response | Object | |
| script | Object | |

View Source common/SS_DunningLetters.js, line 193

Returns: Type: void

MODULE

# SS_DunningStatements

Dunning Statements module for creating and sending dunning statements

View Source common/SS_DunningStatements.js, line 1

### Requires

- module:SS_Constants
- module:SS_DataTableTask
- module:SS_Search
- module:SS_Task
- module:SS_UI

# Members

INNER

### FORM

Type: `Object`

Form configuration object

View Source common/SS_DunningStatements.js, line 44

INNER

### MODULE

Type: `string`

Module name for logging

View Source common/SS_DunningStatements.js, line 37

INNER

### OPERATOR

Type: `Object`

Search operator object

View Source common/SS_DunningStatements.js, line 65

INNER

### REQUEST_FIELDS

Type: `Object`

Request fields object

View Source common/SS_DunningStatements.js, line 58

INNER

### REQUEST_PARAMS

Type: `Object`

Request parameters object

View Source common/SS_DunningStatements.js, line 51

INNER

### REQUEST_TYPE

Type: `Object`

Request type object

View Source common/SS_DunningStatements.js, line 79

INNER

### TRANSACTION

Type: `Object`

Transaction object

View Source common/SS_DunningStatements.js, line 72

# Methods

INNER

### buildSearchFilters(options) → {Array}

Build search filters for dunning statements

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | |
| balance | string | |
| cas | string | |
| condition | string | |
| customer | string | |
| lastcomm | string | |
| levels | string | |
| openbalanceusd | string | |
| subsidiary | string | |

View Source common/SS_DunningStatements.js, line 86

Returns: Type: Array

INNER

### render(options) → {Object}

Render the dunning statements form

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | |
| request | Object | |
| response | Object | |
| defaultValues | Object | |

View Source common/SS_DunningStatements.js, line 233

Returns: Type: Object

INNER

### send(options) → {void}

Send dunning statements

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | |
| body | Object | |
| query | Object | |
| response | Object | |
| script | Object | |

View Source common/SS_DunningStatements.js, line 184

Returns: Type: void

MODULE

# SS_File

Wrapper module for N/file

View Source common/SS_File.js, line 1

Returns:

| | | |
|---|---|---|
| SS_File | Type: `Object` |
| SS_File.create | Type: `function` |
| SS_File.get | Type: `function` |
| SS_File.load | Type: `function` |

### Requires

- module:N/file
- module:SS_Search



# Members

INNER CONSTANT

Type: `string`

### MODULE

Module name for logging

View Source common/SS_File.js, line 31

### Example

```
MODULE = 'outside.File'
```

# Methods

INNER

### getType(options) → {string}

Get the file type based on the file name

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | | |
| name | `string` | |

View Source common/SS_File.js, line 41

Returns: Type: `string`

### Example

```
getType({ name: 'file.txt' });
```

INNER

### parseLast(name) → {string}

Parse the last part of a string

### Parameters:

| Name | Type | Description |
|---|---|---|
| name | `string` | |

View Source common/SS_File.js, line 79

Returns: Type: `string`

### Example

```
parseLast('file.txt');
```

MODULE

# SS_Format

Wrapper module for N/format

View Source common/SS_Format.js, line 1

Returns:

| | |
|---|---|
| SS_Format | Type: `Object` |
| SS_Format.clean | Type: `function` |
| SS_Format.dateToString | Type: `function` |
| SS_Format.format | Type: `function` |
| SS_Format.isEmpty | Type: `function` |
| SS_Format.normalizeDate | Type: `function` |
| SS_Format.parse | Type: `function` |

### Requires

- module:N/format



# Members

INNER CONSTANT

Type: `string`

## MODULE

Module name for logging

View Source common/SS_Format.js, line 32

### Example

```
MODULE = 'outside.Format'
```

# Methods

INNER

## cleanObject(param) → {Object}

Clean an object by removing any empty properties

### Parameters:

| Name | Type | Description |
|---|---|---|
| param | `Object` | |

View Source common/SS_Format.js, line 43

Returns: Type: `Object`

### Example

```
cleanObject({ key: 'value', empty: '' });
```

INNER

## cleanString(value, delimiter) → {string}

Clean a string by removing spaces, slashes, pipes, and underscores

### Parameters:

| Name | Type | Description |
|---|---|---|
| value | `string` | |
| delimiter | `string` | |

View Source common/SS_Format.js, line 63

Returns: Type: `string`

### Examples

```
cleanString('This is a string');
```

```
cleanString('This is a string', '_');
```

```
cleanString('This is a string', '-');
```

INNER

### dateToString(date) → {string}

Convert a date to a string

#### Parameters:

| Name | Type | Description |
| --- | --- | --- |
| date | Date | |

View Source common/SS_Format.js, line 87

Returns: Type: string

#### Examples

```
dateToString(new Date());
```

```
dateToString(new Date('2021-01-01'));
```

INNER

### formatCurrency(options) → {string}

Format a currency value

#### Parameters:

| Name | Type | Description |
| --- | --- | --- |
| options | Object | |
| symbol | string | |
| value | number | |
| hideSymbol | boolean | |

View Source common/SS_Format.js, line 100

Returns: Type: string

#### Examples

```
formatCurrency({ value: 1000 });
```

```
formatCurrency({ value: 1000, symbol: '£' });...
```

INNER

### formatDate(options) → {string}

Format a date value

#### Parameters:

| Name | Type | Description |
| --- | --- | --- |
| options | Object | |
| value | Date | |

View Source common/SS_Format.js, line 122

Returns: Type: string

#### Examples

```
formatDate({ value: new Date() });
```

```
formatDate({ value: new Date('2021-01-01') });
```

```
formatDate({ value: null });
```

INNER

## formatFloat(options) → {string}

Format a floating point number

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | Object | |
| value | number | |

View Source common/SS_Format.js, line 136

Returns: Type: string

INNER

## formatInteger(options) → {string}

Format an integer

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | Object | |
| value | number | |

View Source common/SS_Format.js, line 147

Returns: Type: string

### Examples

```
formatInteger({ value: 1000 });
```

```
formatInteger({ value: 1000.00 });
```

MODULE

# SS_Lib_CustomerStatements

Library functions for Customer Statements

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 1

| Returns: | | |
|---|---|---|
| | SS_Lib_CustomerStatements | Type: `Object` |
| | SS_Lib_CustomerStatements.buildUI | Type: `function` |
| | SS_Lib_CustomerStatements.createStatementTask | Type: `function` |
| | SS_Lib_CustomerStatements.getRequestParameters | Type: `function` |
| | SS_Lib_CustomerStatements.getTableData | Type: `function` |

### Requires

- module:SS_Constants
- module:SS_File
- module:SS_Record
- module:SS_Script
- module:SS_Search
- module:SS_String
- module:SS_UI
- module:SS_Url
- module:SS_Transaction

# Members

INNER CONSTANT

### FORM

Type: `Object`

Form configuration for Customer Statements

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 62

INNER CONSTANT

### MODULE

Type: `string`

Module name for logging and debugging purposes

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 53

INNER CONSTANT

### OPERATOR

Type: `Object`

Search operators

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 72

INNER CONSTANT

### REQUEST_PARAMS

Type: `Array`

Request parameters for Customer Statements

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 81

INNER CONSTANT

### SCRIPT_PARAMS

Type: `Object`

Script parameters for Customer Statements

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 92

INNER CONSTANT

### TRANSACTION

Type: `Object`

Transaction constants

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 103

# Methods

INNER

### buildSearchFilters(options) → {Array}

Build search filters for Customer Statements

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | |
| balance | Number | |
| cas | Number | |
| condition | String | |
| customer | Number | |
| lastcomm | Date | |
| level | Number | |
| subsidiary | Number | |
| searchObject | Object | |

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 112

Returns: filters Type: Array

INNER

### buildUI(options) → {Object}

Build the User Interface for Customer Statements

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | |
| request | Object | |

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 202

Returns: { status, object } Type: Object

INNER

### createStatementTask(options) → {Object}

Create a Map/Reduce task for generating Customer Statements

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | Object | |
| parameters | Object | |

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 245

Returns: { status, data } Type: Object

**Example**

```
createStatementTask({ parameters });
```

INNER

### createStatementTaskRecord(options) → {Object}

Create a Map/Reduce task record for generating Customer Statements

**Parameters:**

| Name | Type | Description |
|---|---|---|

| | |
|---|---|
| options | Object |
| parameters | Object |

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 285

Returns: { status, data } Type: Object

**Example**

```
createStatementTaskRecord({ parameters });
```

INNER

## getRequestParameters(options) → {Object}

Get the request parameters for Customer Statements

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | | |
| request | Object | |
| request.parameters | Object | |
| request.parameters.action | String | |
| request.parameters.balance | String | |
| request.parameters.cas | String | |
| request.parameters.condition | String | |
| request.parameters.customer | String | |
| request.parameters.lastcomm | String | |
| request.parameters.level | String | |
| request.parameters.subsidiary | String | |

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 321

Returns: output Type: Object

**Example**

```
getRequestParameters({ request });
```

INNER

## getTableData(options) → {Object}

Get the data for the Customer Statements table

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | | |
| parameters | Object | |
| parameters.balance | String | |
| parameters.cas | String | |
| parameters.condition | String | |
| parameters.customer | String | |
| parameters.lastcomm | String | |
| parameters.level | String | |
| parameters.subsidiary | String | |

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 356

Returns: { columns, rows } Type: Object

INNER

### getUrlValues() → {Object}

Get the URL values for Customer Statements

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 432

Returns: Type: `Object`

**Example**

```
getUrlValues();
```

INNER

### loadSearch(options) → {Object}

Load a saved search

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | | |
| searchId | String | |

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 447

Returns: search Type: `Object`

**Example**

```
loadSearch({ searchId });
```

INNER

### mapParametersToFields(options) → {Object}

Map the request parameters to the form fields

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | | |
| parameters | Object | |

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 467

Returns: output Type: `Object`

INNER

### writeDataTable(options)

Write the data table to the form

**Parameters:**

| Name | Type | Description |
|---|---|---|
| options | | |
| data | Object | |
| form | Object | |
| template | String | |

View Source DataTableSuitelets/SS_Lib_CustomerStatements.js, line 499

**Example**

```
writeDataTable({ data, form, template });
```

MODULE

# SS_Record

This module is a wrapper for the N/record module

View Source common/SS_Record.js, line 1

Returns: Record Type: `Object`

## Requires

- module:N/record
- module:SS_Constants

## Classes

Record

MODULE

# SS_Query

Wrapper module for N/query

View Source common/SS_Query.js, line 1

Returns: - Module functions Type: `Object`

### Requires

- module:N/query

# Members

INNER CONSTANT

Type: `string`

### CURRENCIES

SQL query to get all active currencies from the Currency table

View Source common/SS_Query.js, line 23

INNER CONSTANT

Type: `string`

### CUSTOM_LISTS

SQL query to get all active custom lists

View Source common/SS_Query.js, line 65

INNER CONSTANT

Type: `string`

### CUSTOM_LIST_VALUES

SQL query to get all values from a custom list

View Source common/SS_Query.js, line 45

INNER CONSTANT

Type: `string`

### STATE_COUNTRIES

SQL query to get all countries and states

View Source common/SS_Query.js, line 84

MODULE

# SS_Request

Wrapper module for serverRequest object

View Source common/SS_Request.js, line 1

Returns:

| | |
|---|---|
| SS_Request | Type: `Object` |
| SS_Request.getParameter | Type: `function` |
| SS_Request.getAllParameters | Type: `function` |

# Members

INNER

Type: `string`

## MODULE_NAME

Module name for logging

View Source common/SS_Request.js, line 23

MODULE

# SS_Runtime

Wrapper module for N/runtime

View Source common/SS_Runtime.js, line 1

| | | |
|---|---|---|
| Returns: | SS_Runtime | Type: `Object` |
| | SS_Runtime.AccountId | Type: `string` |
| | SS_Runtime.Environment | Type: `string` |
| | SS_Runtime.ExecutionContext | Type: `string` |

## Requires

- module:N/runtime

MODULE

# SS_SL_DataTables

This file contains the code for the Suitelet that will render the DataTables Suitelet.

View Source DataTableSuitelets/SS_SL_DataTables.js, line 1

### Requires

- module:SS_DataTableSuitelet

# Members

INNER

Type: `string`

### MODULE

Constant variable to store the module name.

View Source DataTableSuitelets/SS_SL_DataTables.js, line 20

# Methods

INNER

### onRequest(options) → {void}

Function that will be called when the Suitelet is accessed.

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | `Object` | |
| request | `ServerRequest` | |
| response | `ServerResponse` | |

View Source DataTableSuitelets/SS_SL_DataTables.js, line 27

Returns: Type: `void`

MODULE

# SS_SL_DataTables_Backend

This file contains the code for the Suitelet that will render the DataTables Suitelet.

View Source DataTableSuitelets/SS_SL_DataTables_Backend.js, line 1

### Requires

- module:SS_DataTableBackend

# Members

INNER

Type: `string`

### MODULE

Constant variable to store the module name.

View Source DataTableSuitelets/SS_SL_DataTables_Backend.js, line 22

# Methods

INNER

### onRequest(context) → {void}

Function that will be called when the Suitelet is accessed.

### Parameters:

| Name | Type | Description |
|---|---|---|
| context | `Object` | Suitelet context |
| request | `ServerRequest` | Suitelet request |
| response | `ServerResponse` | Suitelet response |

View Source DataTableSuitelets/SS_SL_DataTables_Backend.js, line 31

Returns: Type: `void`

MODULE

# SS_Script

Wrapper module for N/runtime - Script object

View Source common/SS_Script.js, line 1

| Returns: | | |
|---|---|---|
| | SS_Script | Type: `Object` |
| | SS_Script.ApiVersion | Type: `number` |
| | SS_Script.BundleIds | Type: `Array` |
| | SS_Script.DeploymentId | Type: `number` |
| | SS_Script.Id | Type: `number` |
| | SS_Script.LogLevel | Type: `string` |
| | SS_Script.RemainingUsage | Type: `number` |
| | SS_Script.Script | Type: `Object` |
| | SS_Script.Triggers | Type: `Object` |
| | SS_Script.UserEventTypes | Type: `Object` |

### Requires

- module:N/runtime



## Members

INNER CONSTANT

Type: `string`

### MODULE

Module name for logging

View Source common/SS_Script.js, line 32

INNER CONSTANT

### USER_EVENT_TYPES

User event types

View Source common/SS_Script.js, line 42

### Example

```
{{Scheduled: {}, ClientScript: {SAVE_RECORD: string, PAGE_INIT: string, VALIDATE_FIELD: string, FIELD_CHANGED: string}, RESTlet: {}, WorkflowAction: {ACTION: string}, Suitelet: {
```

MODULE

# SS_Search

Wrapper module for N/search

View Source common/SS_Search.js, line 1

| Returns: | | |
|---|---|---|
| | SS_Search | Type: `Object` |
| | SS_Search.Operator | Type: `Object` |
| | SS_Search.Sort | Type: `Object` |
| | SS_Search.Summary | Type: `Object` |
| | SS_Search.Type | Type: `Object` |
| | SS_Search.cast | Type: `function` |
| | SS_Search.create | Type: `function` |
| | SS_Search.load | Type: `function` |
| | SS_Search.loadFromParameter | Type: `function` |
| | SS_Search.lookup | Type: `function` |

## Requires

- module:N/search
- module:SS_Script
- module:SS_String

## Classes

Search

# Members

INNER CONSTANT

Type: `string`

### MODULE

Module name for logging and debugging purposes

View Source common/SS_Search.js, line 42

### Example

```
{{readonly Operator: *, readonly Type: *, readonly LogicalOperator: {OR: string, AND: string}, readonly Sort: *, readonly Summary: *}}
```

MODULE

# SS_String

Utility functions for strings

View Source common/SS_String.js, line 1

Returns:

| | |
|---|---|
| SS_String | Type: `Object` |
| SS_String.generateRandomString | Type: `function` |
| SS_String.normalize | Type: `function` |

# Members

**INNER** **CONSTANT**

Type: `string`

## MODULE

Module name for logging and debugging purposes

View Source common/SS_String.js, line 22

MODULE

# SS_Task

Wrapper module for N/task

View Source common/SS_Task.js, line 1

Returns: SS_Task — Type: `Object`

SS_Task.createMapReduceTask — Type: `function`

### Requires

- module:N/task

# Members

**INNER** **CONSTANT** — Type: `string`

### MODULE

Module name for logging and debugging purposes

View Source common/SS_Task.js, line 27

MODULE

# SS_Transaction

Wrapper module for Transaction constants

View Source common/SS_Transaction.js, line 1

| Returns: | | |
|---|---|---|
| | SS_Transaction | Type: `Object` |
| | SS_Transaction.Fields | Type: `Object` |
| | SS_Transaction.SublistFields | Type: `Object` |
| | SS_Transaction.Sublists | Type: `Object` |
| | SS_Transaction.Types | Type: `Object` |

### Requires

- module:N/record

# Members

**INNER** **CONSTANT**

Type: `Object`

## FIELDS

Transaction fields

### Properties:

| Name | Type | Description |
|---|---|---|
| ACCOUNT | string | |
| ALT_NAME | string | |
| AMOUNT_REMAINING | string | |
| BALANCE | string | |
| CLASS | string | |
| CLEARED | string | |
| CLEARED_DATE | string | |
| CREATED_FROM | string | |
| CURRENCY | string | |
| CURRENCY_NAME | string | |
| CURRENCY_SYMBOL | string | |
| CUSTOM_FORM | string | |
| DAYS_OVERDUE | string | |
| DEPARTMENT | string | |
| ENTITY | string | |
| EXCHANGE_RATE | string | |
| EXTERNAL_ID | string | |
| ID | string | |
| INTERNAL_ID | string | |
| IS_BASE_CURRENCY | string | |
| JOB | string | |
| LAST_MODIFIED_DATE | string | |
| LOCATION | string | |
| MEMO | string | |
| NAME | string | |
| OTHER_REF_NUM | string | |
| PAYMENT | string | |

| | |
|---|---|
| POSTING_PERIOD | `string` |
| PREPAYMENT_ACCOUNT | `string` |
| PRINT_VOUCHER | `string` |
| PURCHASE_ORDER | `string` |
| STATUS | `string` |
| SUBSIDIARY | `string` |
| SUPERVISOR_APPROVAL | `string` |
| TERMS | `string` |
| TO_BE_PRINTED | `string` |
| TRAN_DATE | `string` |
| TRAN_ID | `string` |
| TRANSACTION_NUMBER | `string` |
| VENDOR | `string` |

View Source common/SS_Transaction.js, line 28

**INNER** **CONSTANT**

Type: `Object`

## SUBLISTS

Transaction sublists for record

### Properties:

| Name | Type | Description |
|---|---|---|
| ACCOUNTING_BOOKS | `string` | |
| EXPENSE | `string` | |
| ITEM | `string` | |

View Source common/SS_Transaction.js, line 120

**INNER** **CONSTANT**

Type: `Object`

## SUBLIST_FIELDS

Transaction sublist fields

### Properties:

| Name | Type | Description |
|---|---|---|
| ACCOUNTING_BOOK | `string` | |
| AMOUNT | `string` | |
| CLASS | `string` | |
| DEPARTMENT | `string` | |
| DESCRIPTION | `string` | |
| EXCHANGE_RATE | `string` | |
| ITEM | `string` | |
| PORT | `string` | |
| QUANTITY | `string` | |
| RATE | `string` | |
| VOYAGE | `string` | |

View Source common/SS_Transaction.js, line 138

**INNER** **CONSTANT**

Type: `Object`

## TYPES

Transaction types

**Properties:**

| Name | Type | Description |
| --- | --- | --- |
| CASH_REFUND | string | |
| CASH_SALE | string | |
| CUSTOMER_DEPOSIT | string | |
| CUSTOMER_PAYMENT | string | |
| CUSTOMER_REFUND | string | |
| DEPOSIT | string | |
| DEPOSIT_APPLICATION | string | |
| INVOICE | string | |
| ITEM_FULFILLMENT | string | |
| JOURNAL_ENTRY | string | |
| PURCHASE_ORDER | string | |
| SALES_ORDER | string | |
| VENDOR_BILL | string | |
| VENDOR_PAYMENT | string | |
| VENDOR_PREPAYMENT | string | |

View Source common/SS_Transaction.js, line 173

MODULE

# SS_UI

SS_UI module for creating and managing SuiteScript 2.0 User Interface

View Source common/SS_UI.js, line 1

| Returns: | | |
|---|---|---|
| | SS_UI | Type: `Object` |
| | SS_UI.addButton | Type: `function` |
| | SS_UI.addField | Type: `function` |
| | SS_UI.addSublist | Type: `function` |
| | SS_UI.buildButtons | Type: `function` |
| | SS_UI.buildFieldGroups | Type: `function` |
| | SS_UI.buildFields | Type: `function` |
| | SS_UI.buildForm | Type: `function` |
| | SS_UI.buildSublists | Type: `function` |
| | SS_UI.getDeploymentSublists | Type: `function` |
| | SS_UI.getField | Type: `function` |
| | SS_UI.getSublist | Type: `function` |
| | SS_UI.getSublistColumnKeyByLabel | Type: `function` |
| | SS_UI.mapParametersToFields | Type: `function` |
| | SS_UI.readQueryParameters | Type: `function` |
| | SS_UI.writeSublistData | Type: `function` |

### Requires

- module:N/ui/serverWidget
- module:SS_File
- module:SS_Format
- module:SS_String
- module:SS_Script

## Members

INNER CONSTANT

### MODULE_NAME

Type: `string`

Module name for logging

View Source common/SS_UI.js, line 52

MODULE

# SS_Url

Wrapper module for N/url

View Source common/SS_Url.js, line 1

| | | |
|---|---|---|
| Returns: | SS_Url | Type: `Object` |
| | SS_Url.resolveRecord | Type: `function` |
| | SS_Url.resolveScript | Type: `function` |

## Requires

- module:N/url

CLASS

# DataTable

## SS_DataTable~DataTable(options)

CONSTRUCTOR

### new DataTable(options)

DataTable class for rendering a DataTable

### Parameters:

| Name | Type | Description |
| --- | --- | --- |
| options | Object | The options object for the DataTable |
| columns | Array | The columns for the DataTable |
| rows | Array | The rows for the DataTable |

View Source common/SS_DataTable.js, line 39

### Example

```
const dataTable = new DataTable({
    columns: [
```

# Record

## SS_Record~Record(options) → {Object}

CONSTRUCTOR

### new Record(options) → {Object}

Wrapper class for N/record

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | `Object` | |
| record | `Object` | |
| record.id | `string` | |
| record.type | `string` | |

View Source common/SS_Record.js, line 40

Returns: Record Type: `Object`

### Example

```
new Record({ record: record });
```

# Search

## SS_Search~Search(options)

CONSTRUCTOR

### new Search(options)

Wrapper class for N/search

### Parameters:

| Name | Type | Description |
|---|---|---|
| options | Object | |
| columns | Array | |
| filters | Array | |
| search | Object | |
| type | string | |

View Source common/SS_Search.js, line 70

### Example

```
const search = new Search({ columns: [], filters: [], search: {}, type: 'customer' });
```