Sage 300 Web Screens SDK

Declarative Setup Reports Design

May 2022

The MIT License (MIT)

Copyright © 2022 The Sage Group plc or its licensors. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Contents

[1. Introduction 4](#_Toc104204581)

[2. Objectives 5](#_Toc104204582)

[3. Design 6](#_Toc104204583)

[3.1 Diagram 6](#_Toc104204584)

[3.2 Features 7](#_Toc104204585)

[3.2.1 File Path 7](#_Toc104204586)

[3.2.2 One Report per Configuration 7](#_Toc104204587)

[3.2.3 Menu Screen URL 8](#_Toc104204588)

[3.2.4 HTML Generation 9](#_Toc104204589)

[3.2.5 External Language Resources 9](#_Toc104204590)

[3.2.6 Client-Side Range Validation 10](#_Toc104204591)

[3.2.7 Server Validation 10](#_Toc104204592)

[3.2.8 Security Resources 10](#_Toc104204593)

[3.2.9 Special Validations 11](#_Toc104204594)

[3.2.10 Customizations 12](#_Toc104204595)

[3.2.11 Error Handling 12](#_Toc104204596)

[3.3 New Files 13](#_Toc104204597)

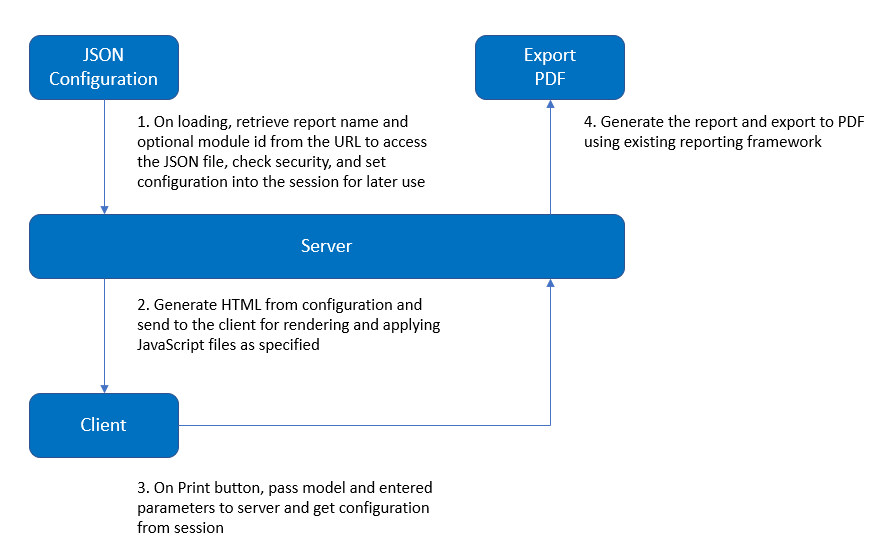
1. Introduction

A web screen for a report prompts for user input, generates a Crystal report, and exports the report to a PDF file. Currently, all web reports are implemented as a separate web screen that requires all the server and client files for a web screen. The code generation wizard generates the template files for the report, but there is still quite a bit of development work required to create the web screen, and there is a lot of maintenance for the numerous files generated for all the reports.

For the Sage 300 2023 release, we have identified close to 100 “simple” setup reports, which have limited to no-user inputs, that need to be created to achieve parity between the desktop and web screens.

This project intends to target web screen reports by creating a framework to simplify the report creation and maintenance of “simple” reports such as these. This framework will dynamically generate the web screen report driven by a JSON configuration.

1. Objectives
2. Target the approximately 100 setup report screens that are still to be created as web screens that have no to limited user inputs
3. Create a framework that reads a report configuration from a JSON file and dynamically generates a web screen for the report
4. Support 3rd party development of simple reports as web screens leveraging this new framework
5. Leverage existing reporting and printing frameworks in the web screens (export to PDF, etc.)
6. Design
   1. Diagram



* 1. Features
     1. File Path

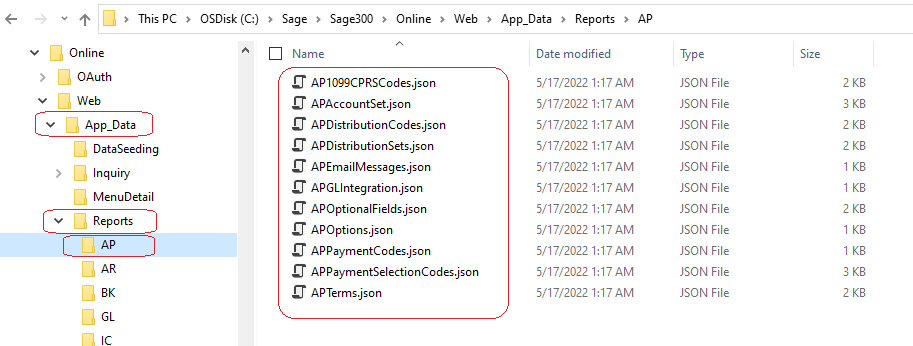
The path to the JSON configuration is configurable in the web.config file:

<add key="DeclarativeReportLocation" value="~/App\_Data/Reports" />

The default is **"~/App\_Data/Reports**" and both absolute and relative paths are supported.

* + 1. One Report per Configuration

A JSON configuration contains one report and can be found in the above noted file path in a ***module Id*** subfolder:



The folder hierarchy is for segregation of reports and to accommodate 3rd party reports

* + 1. Menu Screen URL

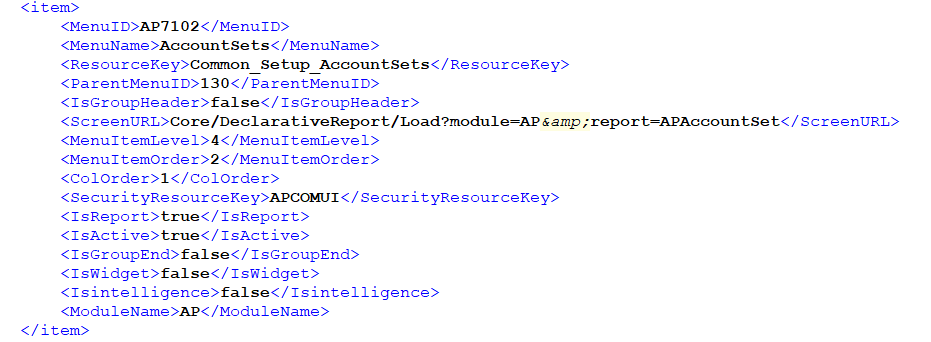
The URL for a declarative report is specified in the module’s XML menu file:

<ScreenURL>Core/DeclarativeReport/Load?module=AP&amp;report=APAccountSet</ScreenURL>

The ***module*** parameter is optional and refers to the folder under the above noted file path. Multiple subfolders may be defined in the module parameter (i.e., ValuedPartner1/AP).

The ***report*** parameter is required and refers to the name of the JSON configuration file without extensions.

An example of the AP Account Sets declarative setup report’s menu item:



* + 1. HTML Generation

With declarative reports, the screen layout must be dynamically generated. The server creates the HTML for each parameter using HTML Helper functions based on the control type of the parameter and returns the HTML to the client with one row per parameter.

If another layout other than this base layout is desired, the ***HtmlTemplate*** property is used to specify the HTML to be rendered with the parameters injected.

The list of supported control types is defined in the ***DeclarativeControlTypeEnum***:

public enum DeclarativeControlTypeEnum

{

Textbox,

Finder,

Dropdown,

Checkbox,

Hidden,

Numeric,

Datepicker,

RadioButtonGroup,

Currency,

Section

}

Some controls require other properties to be defined, such as the finder and dropdown controls.

For details on each property, refer to Declarative Report Interface Document.

* + 1. External Language Resources

The ***ResourceAssemblyPath*** and ***ResourceAssembly*** properties are used to specify the assembly file to load during runtime for localization. This assembly should contain a project with the resx file used for the report. The report and parameter properties suffixed with “Resource” will read from this assembly for the value to display.

The resource name of these properties should include the namespace, followed by a dot, and the resource name.

For details on each property, refer to Declarative Report Interface Document.

* + 1. Client-Side Range Validation

The ***Ranges*** property is used to specify any paired parameters that have client-side range validation. If the value specified in the ***From*** property is greater than the value specified in the ***To*** property, an error message will be displayed containing the ***LabelResource*** property when the user clicks the print button.

Parameter pairs defined in ***Ranges*** property will be rendered horizontally in the layout as opposed to normally being displayed vertically.

* + 1. Server Validation

Upon loading the settings from the JSON configuration, the settings are saved in the .NET session which are retrieved on print for validation that the client has only changed the parameter values or the report name (since multiple RPT files can be printed via a single screen).

This prevents the client side from attempting to print a different report or a report with different parameters than what is configured.

* + 1. Security Resources

The ***SecurityResource*** property is used to verify that the user has permissions to print the report and is interrogated on each parameter and therefore does not generate the parameter for input if access is denied.

This is the same value as the ***SecurityResourceKey*** in the module’s XML menu, which verifies that the user can access the web screen from the menu.

Multiple Security Resources may be specified where a hyphen (-) is “or” and a comma (,) is “and”. For example:

“SecurityResource”: “ICINQRY-ICSTOCKI,ICCOST-ICTRANSI”

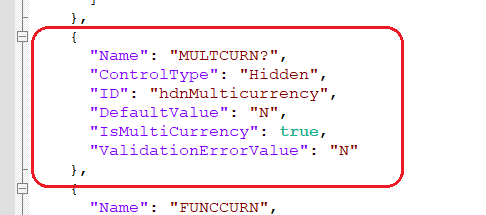
Which reads: ICINQRY || ICSTOCK && ICCOST || ICTRANSI

* + 1. Special Validations

For reports with parameters that set multicurrency, optional fields, or licenses, the ***IsMultiCurrency***, ***LicensesToValidate***, and ***IsOptionalFields*** properties coupled with the ***ValidationErrorValue*** property are used to validate the correct values on the server.

On print, if those properties are set for a parameter, the server will check if those are true on the server. If false, the parameter value will be set to the ***ValidationErrorValue*** instead of the value the client passed in.

For example, the AP Account Set report has a parameter “**MULTCURN?**” that sets whether multicurrency is true or false. The ***IsMultiCurrency*** is set to true, and ***ValidationErrorValue*** is set to “N”. On the client side, this parameter’s value can be set to any value. On print, the server checks that ***IsMultiCurrency*** is true and if so, it checks whether AP Multicurrency is true. If not, it will change the parameter’s value to “N” before passing it to print.



One additional step to validate multicurrency, optional fields, or licenses has been added to Sage300 reports only, and is not available to 3rd parties.

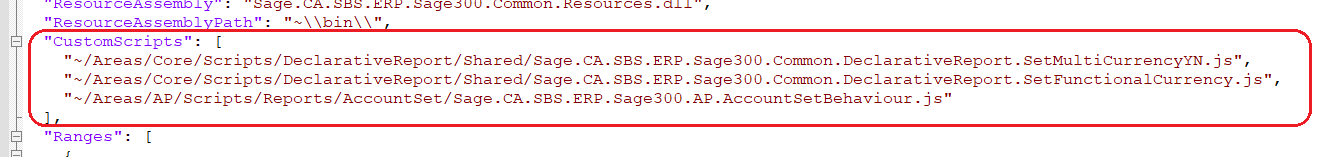
Since the validation above depends on the properties defined in the JSON configuration, they can be removed by the system admin or others with access to the server.

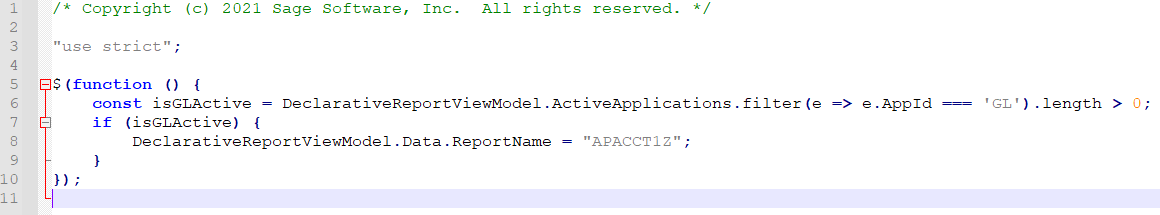
Sage.CA.SBS.ERP.Sage300.Common.Models.Reports.DeclarativeReportValidationList is a class which mirrors the JSON configuration created by Sage300 and contains only the validation properties for each report. This is used to validate those properties, even when they may be missing from the JSON configuration.

* + 1. Customizations

If the report requires additional business logic, such as dynamically changing a controls’ visibility and/or data source, etc., the ***CustomScripts*** property can be used to specify Javascript file(s) to be included.

For example, the AP Account Set report changes the ***ReportName*** property depending on if GL is activated.





The ***RazorViews*** property allows razor views to be included at the bottom of the page, which can be used to add partial views or localization resources.

* + 1. Error Handling

If invalid values are supplied in the JSON configuration, an ***InvalidReportSettingException*** is thrown which will log the report, property, parameter name, and value in the **trace.log** file in the web application’s **Log** folder.

For errors occurring due to JSON deserialization, the **NewtonSoft.JSON** error will be logged, and the user will see the generic error message.

For errors occurring during printing, the **ExportReport** application has been enhanced to log errors from the session (in a separate log file).

* 1. New Files

|  |  |
| --- | --- |
| **Class** | **Description** |
| **Sage.CA.SBS.ERP.Sage300.Common.Models.Reports** | |
| DeclarativeReport | Setting for report. Inherits from ReportBase |
| DeclarativeParameter | Setting for report parameter |
| DeclarativeParameterTextbox | Setting for text parameter. Inherits from DeclarativeParameter |
| DeclarativeParameterFinder | Setting for finder parameter. Inherits from DeclarativeParameter |
| DeclarativeParameterNumeric | Setting for numeric or currency parameter. Inherits from DeclarativeParameter |
| DeclarativeParameterDatepicker | Setting for date parameter. Inherits from DeclarativeParameter |
| DeclarativeParameterDropdown | Setting for dropdown parameter. Inherits from DeclarativeParameter |
| DeclarativeParameterCheckbox | Setting for checkbox parameter. Inherits from DeclarativeParameter |
| DeclarativeParameterRadioButtonGroup | Setting for radio button group parameter. Inherits from DeclarativeParameter |
| DeclarativeParameterSection | Setting for parameter that contains other parameters. Inherits from DeclarativeParameter |
| DeclarativeControlType | Enum of supported html control types |
| DeclarativeRangeParameter | Setting for validating From/To range parameters and displaying them horizontally |
| DeclarativeDataSourceParameter | Setting for dropdown, checkbox, and radio button group parameters’ data source |
| DeclarativeKeyValueParameter | Setting for dictionary type properties |
| DeclarativeControlSectionRenderDirectionEnum | Enum for section parameter to specify horizontal or vertical layout |
| DeclarativeParameterConverter | Overrides ReadJson to allow NewtonSoft.JSON to deserialize different parameter types. Inherits from NewtonSoft.Json.JsonConverter |
| DeclarativeReportValidationList | Defines server-side validation rules for Declarative Reports written by Sage300 |
| **Sage.CA.SBS.ERP.Sage300.Common.Interfaces.Repository.Reports** | |
| IDeclarativeReportRepository | Interface for repository. Inherits from IReportRepository |
| **Sage.CA.SBS.ERP.Sage300.Common.Interfaces.Service.Reports** | |
| IDeclarativeReportService | Interface for service. Inherits from IReportService |
| **Sage.CA.SBS.ERP.Sage300.Common.BusinessRepository.Reports** | |
| DeclarativeReportRepository | Inherits from BaseReportRepository |
| **Sage.CA.SBS.ERP.Sage300.Common.BusinessRepository.Mapper.Reports** | |
| DeclarativeReportMapper | Implements IReportMapper. Dynamically maps report parameters from DeclarativeReport |
| **Sage.CA.SBS.ERP.Sage300.Common.Services.Reports** | |
| DeclarativeReportService | Inherits from BaseReportService |
| **Sage.CA.SBS.ERP.Sage300.Core.Web.Models** | |
| DeclarativeReportViewModel | View model for the page |
| **Sage.CA.SBS.ERP.Sage300.Core.Web.Controllers** | |
| DeclarativeReportControllerInternal | Inherits from ReportControllerInternal. Parses JSON file to generate parameters and read settings |
| DeclarativeReportController | Inherits from ReportController. Load action takes 2 parameters for report and module. Gets report setting path. Overrides Execute action to get settings from cache to print report |
| **Client Side** | |
| ~\Areas\Core\Views\DeclarativeReport\Load.cshtml | |
| ~\Areas\Core\Views\DeclarativeReport\Partials\\_Localization.cshtml | |
| ~\Areas\Core\Views\DeclarativeReport\Partials\\_DeclarativeReport.cshtml | |
| ~\Areas\Core\Scripts\DeclarativeReport\Sage.CA.SBS.ERP.Sage300.Common.DeclarativeReportBehaviour.js | |
| ~\Areas\Core\Scripts\DeclarativeReport\Sage.CA.SBS.ERP.Sage300.Common.DeclarativeReportRepository.js | |