Sage 300 Web Screens SDK

Web Help

May 2021

The MIT License (MIT)

Copyright © 2021 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](#_Toc71283980)

[2. Web Help Framework 6](#_Toc71283981)

[2.1 Sage 300 Screen Help 6](#_Toc71283982)

[2.2 Third-Party Screen Help 6](#_Toc71283983)

[3. MenuHelp.xml Layout 8](#_Toc71283984)

[3.1 Example 9](#_Toc71283985)

[3.2 Elements and Attributes 11](#_Toc71283986)

[4. Localization Resource 12](#_Toc71283987)

[4.1 Visual Studio 12](#_Toc71283988)

[4.1.1 Create a Solution and Project 12](#_Toc71283989)

[4.1.2 Create English Resource File 12](#_Toc71283990)

[4.1.3 Mark English Resource File as Public 12](#_Toc71283991)

[4.1.4 Create Non-English Resources as Required 12](#_Toc71283992)

[4.1.5 Create Resource Names and Values 13](#_Toc71283993)

[4.1.6 Compiling 13](#_Toc71283994)

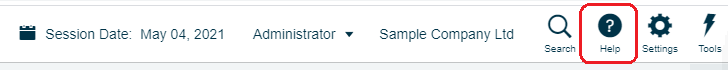
[4.1.7 References in the XML file 13](#_Toc71283995)

[5. Deployment 14](#_Toc71283996)

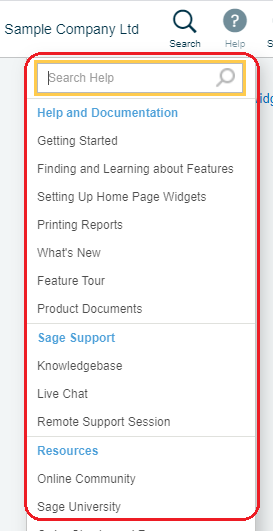
[6. Sample 15](#_Toc71283997)

1. Introduction

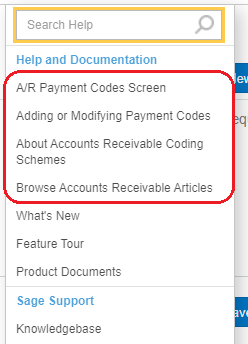
The Sage 300 web screens have a ***Help*** icon in the banner section of the web application for providing application and context sensitive (screen specific) help:



The topics listed under the ***Help and Documentation*** section are provided by Sage 300’s UX Content team:



The web framework allows for up to four (4) optional “screen specific items” (although there is no validation for this limit!) to be added to the list of topics directly under the ***Help and Documentation*** section (the AR Payment Codes screen is displayed below):



Third parties have requested the ability to participate in this optional context sensitive area of the web help system, but the current implementation does not allow it.

This has changed in Sage 300 2022 and third parties can now seamlessly integrate their web screen help topics along with Sage 300 help topics.

The implementation and deployment will be covered in this document, but it is important to note that this implementation is for providing the hooks required to display the help content.

Sage 300 is not hosting or validating the third-party help content.

This enhancement will provide for seamless integration, consistency, and a good user experience.

1. Web Help Framework

The Web Help framework is responsible for displaying Sage 300 help.

We will look at how the Sage screen and the new third-party screen context help is discovered and rendered.

The Sage Screen’s MenuHelp.xml file is dramatically different from the new third party’s MenuHelp.xml file. The format of the Sage help URL is known and can resolve the tokens in a known fashion whereas the format and replacement for the third-party tokens must be described in the XML.

* 1. Sage 300 Screen Help

Whether the customer is on the homepage or a web screen, when the **Help** icon is clicked, the process of finding the Menu Help (help) begins.

If the help has not already been cached (for performance on subsequent invocations), the following are performed:

* The application will ensure that the Help Menu Configuration filename (**~/App\_Data/MenuHelp.xml**) has been established in the web.config file.
  + An error will be thrown if the configuration is not found, but this is unlikely.
* The XML file will be read to retrieve the elements to build a complete list of help topics.
* The list is cached.

Reading from the cache, the help items for the underlying screen (homepage or web screen) are retrieved. Since help is optional for any screen, if no screen help items are discovered, then there simply will not be any screen specific help displayed.

If topics are discovered, the URL is built by replacing several tokens and the screen help links are displayed.

* 1. Third-Party Screen Help

Once the Sage 300 help topics are discovered by reading Sage’s Menu Help XML file, and before it returns to cache these topics, the framework needs to determine if any third-party help files exist:

* The framework will search for External Help Menu Configurations in the ***ExternalHelp*** folder (**~/App\_Data/*ExternalHelp***).

Since multiple third-party help files may be present on a customer site, each third-party will create/install their own unique folder under the ExternalHelp folder (i.e., ~/App\_Data/ExternalHelp/Partner1, ~/App\_Data/ExternalHelp/Partner2, etc.).

The third-party folder will contain the External Help Menu files:

* + **MenuHelp.XML**
    - Required. Contains the menu help topics for the third-party
  + {***ResourceFile***}.**dll**
    - Optional: English is the default language
  + **es/{*ResourceFile*}.resources.dll** 
    - Optional: if Spanish translation is provided
  + **fr/{*ResourceFile*}.resources.dll** 
    - Optional: if French translation is provided
  + **zh-Hans/{*ResourceFile*}.resources.dll** 
    - Optional: if Chinese Simplified translation is provided
  + **zh-Hant/{*ResourceFile*}.resources.dll** 
    - Optional: if Chinese Traditional translation is provided
* The MenuHelp.xml file will be read to retrieve the required elements. It is at this time that the URL and Description will be resolved since the elements in the XML will require immediate resolution/replacement as opposed to Sage’s implementation where this information is resolved at display time.
* The list is added to the cache.

Reading from the cache, the help items for the underlying screen (homepage or web screen) are retrieved. Since help is optional for any screen, if no screen help items are discovered, then there simply will not be any screen specific help displayed.

If topics are discovered, the URL which has already been built will display the screen help links.

1. MenuHelp.xml Layout

As mentioned, the XML layout for third parties will differ from the Sage 300 XML layout. This section will only describe the third-party layout.

* 1. Example

The following is an example of a MenuHelp.xml for a third-party. The URL specified is **based** upon Sage’s help URL, but this is only used for example purposes as third parties are in control of their URL format, tokens, etc. Likewise, the naming of the resource and resource keys are purely for illustration.

<?xml version="1.0" encoding="utf-8" ?>

<!-- MenuHelp.xml for Third Party menu items -->

<!-- 'root' (required) - The root element -->

<root>

<!-- 'url' (optional) - The url for all items, if not specified, every item must specify a url)

'value' (required) - The url for all items and can be overridden by the item -->

<url value="https://help.sage300.sageinternal.com/{0}/2022/web/index.htm#cshid={1}?utm\_source=helpmenu&amp;utm\_medium=inproduct&amp;utm\_campaign=sage300" />

<!-- 'resource' (optional) - Resource for localizing descriptions

'name' (required) - The name of a resource dll

'class' (required) - The class in the resource dll -->

<resource name="ValuedPartner.dll" class="ValuedPartner.Localization" />

<!-- 'menu' (required) - The list of menus (screens) to be added to help list

'id' (required) - Identifies the menu (screen) and comes from XXMenuDetails.xml -->

<menu id="TU4001" > <!-- Third party menu (screen) with two menu items -->

<!-- 'item' (required) - The list of items for the menu (screen)

'resourceKey' (optional) - Key to localization resource

'description' (required) - Description of item if not localized or if localization fails

'url' (optional) - If specified, will override url specified for all items -->

<item resourceKey="Clone" description="Pay Code Clone" url = "">

<!-- 'token' (optional) - The list of tokens to be replaced in url

'token' (required) - The token to be replaced (i.e. {0} or {whatever} or FOO or ...)

'value' (optional) - The value to be used for the token

If 'local' == "true", the 'value' will be ignored

If 'locale' not specified, a 'value' must be specified

'locale' (optional) - Flag to specify whether to use the user's web locale -->

<token token="{0}" locale="true" />

<token token="{1}" value="18003001" />

</item>

<item resourceKey="Google" description="Google" url = "http://Google.com" />

</menu>

<menu id="AR5107" > <!-- Sage 300 A/R Payment Codes menu (screen) with 1 additional menu item -->

<item resourceKey="Google" description="Google" url = "http://Google.com" />

</menu>

</root>

Here is a verbalization of what this xml layout is saying:

*“Here is an xml menu help file for a third party.*

*In the “url” element is defined a URL in the “value” attribute for all menu items unless a URL is specified at the menu item element which will then be used instead.*

*In the “resource” element a resource has been specified to be used to translate the description for the menu item.*

*In the first “menu” element is a screen that will have menu items that will be added to Sage’s screen help.*

*In the first “item” element is this menu item’s elements, which are a resource key that will be used to access the resource to provide a localized description for the menu item, a description that can be used in case the localization fails, and there is no override or specific URL for this item so the URL for all menu items will be used.*

*In the first and second “token” elements are tokens and values that will be used to identify tokens in the URL for all menu items and their replacement values. Since the “locale” element is set to true, this means that this token is a special token and instead of providing a value, the user’s locale in which they are currently using in the web screen will be used as a replacement value whereas if a “value” element is specified then it will be used as the replacement value.*

*In the second “item” element is this menu item’s elements, which are a resource key that will be used to access the resource to provide a localized description for the menu item, a description that can be used in case the localization fails, and there is an override URL specified in the “url” attribute so this URL will be used instead of the URL for all menu items.*

*In the second “menu” element is a Sage 300 screen that will have a partner specified menu item added to it.*

*In the final “item” element is this menu item’s elements, which are a resource key that will be used to access the resource to provide a localized description for the menu item, a description that can be used in case the localization fails, and there is an override URL specified in the “url” attribute so this URL will be used instead of the URL for all menu items.”*

* 1. Elements and Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Element | Description | Attribute | Description |
| root | ***Required***. The root element |  |  |
| url | ***Optional***. The url element which identifies the url for all menu items. If not specified, a URL must be identified for every menu item | value | ***Required***. The URL for all menu items, but can be overridden by the menu item’s url |
| resource | ***Optional***. The resource element which identifies a resource for localizing descriptions | name | ***Required***. The name of the resource dll. |
| class | ***Required***. The class in the resource dll. |
| menu | ***Required****.* The menu element which identifies a screen requiring help menu items | id | ***Required***. The id of the screen which comes from the XXMenuDetails.XML file. |
| item | ***Required.*** The item element which identifies a menu item for the screen | resourceKey | ***Optional.*** The resource key in the localization resource. |
| description | ***Required.*** The description of the item if not localized or if the localization fails. |
| url | ***Optional.*** The url if specified will override the url specified for all menu items |
| token | ***Optional.*** The token element which identifies the list of tokens to be replaced in the url | token | ***Required.*** The token to be replaced in the url |
| value | ***Required/Optional.*** The value to be used for the token.  If the locale attribute is true, the value attribute will be ignored and can be omitted.  If the locale attribute is not specified, a value attribute must be specified. |
| locale | ***Optional.*** A special attribute that indicates whether to use the user’s web locale |

1. Localization Resource

A localization resource, if supplied, will localize the description of the displayed menu item. This resource is optional if only an English description is required since the description must be provided in the XML as default.

This section will describe the steps required to create a localization resource using Visual Studio 2019.

* 1. Visual Studio
     1. Create a Solution and Project
* Open Visual Studio and create a new project (File 🡪 New 🡪 Project)
* Select ***Class Library (.Net Framework)*** from list of project templates
* Select ***Next***
* Name the project (i.e., ***Partner1***) and location appropriately, target 4.8 Framework, and select ***Create*** 
  + 1. Create English Resource File

Before creating the resource file, it is important to delete the ***Class1.cs*** file that is created by default when a class library project is created. Select the ***Class1.cs*** file and Select ***Delete***.

Right-Click on the Solution, Select ***Add*** 🡪 ***New Item*** 🡪Select ***Resources File*** from the list and name the RESX file appropriately (i.e., ***Localization.resx***). This will create the main resource which by default is English.

* + 1. Mark English Resource File as Public

After the resource file in the previous step has been created, it must be marked as ***Public*** for the Resource Manager to drive the resource access.

With the resource file displayed in the IDE, select the ***Access Modifier*** drop down and change from ***Internal*** to ***Public***.

* + 1. Create Non-English Resources as Required

Once the main or English resource is created, if non-English resources are required, right-Click on the Solution, Select ***Add*** 🡪 ***New Item*** 🡪Select ***Resources File*** from the list and name the RESX file with the same name as the English resource but with the appropriate locale in the name (i.e., ***Localization.es.resx***). This will create a non-English resource.

The locales to be used are:

* ***es*** (Spanish)
* ***fr*** (French)
* ***zh-Hans*** (Chinese Simplified)
* ***zh-Hant*** (Chinese Traditional)

By default, when the non-English files are created the ***Access Modifier*** will default to ***“No code generation”*** which is correct since only the English resource is the “driver” and all other resources are subjugate to it. Therefore, no changes are required to the ***Access Modifier***.

* + 1. Create Resource Names and Values

Starting with the English resource, enter ***Names*** (keys) in the resource and the ***Values*** for the names (i.e., ***TU\_XX\_1***, ***Localized description***) where ***TU\_XX\_1*** is the value of the ***resourceKey*** attribute in the XML file for a menu item and ***Localized description*** is the localized value to be used for the description of the menu item.

Once the English resource has been populated, the **same** Names (keys) will need to be created in any other non-English resource files. The Values in the non-English resource files will require translation.

* + 1. Compiling

Once the resource files have been populated with the appropriate ***Names*** and ***Values***, the project can be compiled.

In looking at the project’s ***bin*** folder (based upon the examples above), you will find a ***Partner1.dll*** file, a Partner1.pdb file (can be ignored), and a folder called ***es***, which contains a ***Partner1.resources.dll*** (for the Spanish satellite resource).

* + 1. References in the XML file

Since a resource has been created for the purpose of localizing the help menu descriptions, the resource element in the XML file now can be created with the appropriate values added to the attributes. Based upon the examples above, the XML element and attributes would look like:

<resource name="Partner1.dll" class="Partner1.Localization" />

See section 3.2 for Elements and Attributes.

1. Deployment

Once a MenuHelp.xml file and an optional resource if localization is required have been created, the XML file and optional resource files can be delivered to the customer.

As mentioned in section 2.2, third party help is in the ***ExternalHelp*** folder (**~/App\_Data/*ExternalHelp***). Since multiple third-party help files may be present on a customer site, each third-party will create/install their own unique folder under the ExternalHelp folder (i.e., ***ExternalHelp/Partner1***, ***ExternalHelp/Partner2***, etc.).

Based upon the examples above, the ***ExternalHelp*** folder will look like this:

* + **Partner1** (folder)
    - **es** (folder)
      * **Partner1.resources.dll** (localization satellite assembly)
    - **MenuHelp.XML** (xml file)
    - **Partner1.dll** (localization assembly)

Based upon the examples above if localization was not required, the ***ExternalHelp*** folder will look like this:

* + **Partner1** (folder)
    - **MenuHelp.XML** (xml file)

1. Sample

Located in the ***samples/WebHelp*** folder of the Web SDK you will find samples of a MenuHelp.xml file as well as Visual Studio project for localization.