Sage 300 Web Screens SDK

Server-Side Report Callback

February 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](#_Toc64390236)

[2. Server-Side Callback 5](#_Toc64390237)

[2.1 The Report Class 5](#_Toc64390238)

[2.1.1 ReportProcessType Property 5](#_Toc64390239)

[2.1.2 AssemblyName Property 6](#_Toc64390240)

[2.1.3 TypeName Property 6](#_Toc64390241)

[2.1.4 ReportModel Property 6](#_Toc64390242)

[2.2 Ship to Location Report Example 7](#_Toc64390243)

[2.2.1 ShipToLocationReportMapper Class 7](#_Toc64390244)

[2.2.2 ShipToLocationExample Class 8](#_Toc64390245)

[2.2.3 UpdateCustomerAfterPrinting Class 9](#_Toc64390246)

[2.3 The Server-Side Callback Framework 11](#_Toc64390247)

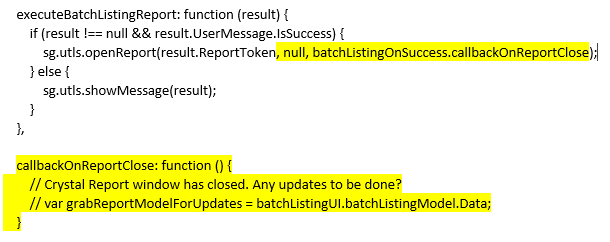
[2.3.1 The Process routine 11](#_Toc64390248)

[3. Summary 12](#_Toc64390249)

1. Introduction

Several releases prior to Sage 300 2021.2, partners requested the ability to be informed (called-back) when a Web Report was closed in the browser. So, functionality was added to the ***sg.utls.openReport*** function to append a function (callback) to be invoked when the report was closed by the customer.

Here is the A/R Batch Listing Report that was modified to illustrate this implementation:



Partners could use this mechanism to update the printed flag or perform other updates.

However, with the release of Sage 300 2021.2, we have changed the way reports are generated, displayed, and closed. The enhancement is called Print to PDF. Instead of rendering reports in either an iFrame or another Tab Page via the Crystal Reports Viewer, we render the report directly to a PDF file where a customer chooses to open the PDF to either print or save or whatever.

Thus, the above implementation is now obsolete as this ability to be informed when a report has closed on the client is no longer possible.

But there is another option!

This document is intended to provide information on the ability for partners to be notified on the server when the report is done generating since the opening, printing, and closing of a report is outside of Sage 300’s control.

1. Server-Side Callback

The ability to be notified when a report has completed generating has been part of the application for a while and screens within the Sage 300 application, such as A/R invoice Printing, use this mechanism.

This section will demonstrate this ability with a simple example:

* The A/R Ship to Location Report will be modified with the ability to have a server-side callback.
* The callback will invoke a new service endpoint with a new object containing a Customer Number.
* The service will invoke the Customer repository with the Customer Number.
  1. The Report Class

The ***Report*** class (*Sage.CA.SBS.ERP.Sage300.Common.Models.Reports*), which is the main class used in the Web Reports, contains the properties that must be set for the callback to be invoked. Let us look at these properties.

* + 1. ReportProcessType Property

The ***ReportProcessType*** property identifies if a callback is to be requested on either the *Load* or the *Close*of the report. This property is defaulted to *Load* in the Report constructor.­

/// <summary>

/// ReportProcessType

/// </summary>

public ReportProcessType ReportProcessType { get; set; }

/// <summary>

/// Constructor

/// </summary>

public Report()

{

Parameters = new List<Parameter>();

ReportProcessType = ReportProcessType.OnLoad;

}

This property is used in conjunction with the following properties: ***AssemblyName***, ***TypeName***, and ***ReportModel*** which will be defined in the following sections. Therefore, this property is moot IF these properties are not set.

At of 2021.2, the *Load* and *Close* callbacks, if specified, are invoked is sequence. In other words, they are the same at this time (always after the report is generated). However, in the future, the Load callback will be placed in the correct order of the report process. Therefore, setting the *ReportProcessType* correctly will be important for future releases.

* + 1. AssemblyName Property

The ***AssemblyName***property identifies the name of the assembly containing the type to be invoked on the callback.­

/// <summary>

/// Name of the assembly.This is required if anything needs to be processed

Before/after the report is generated.

/// </summary>

public string AssemblyName { get; set; }

This property must be set for the callback to be invoked. An example:

Sage.CA.SBS.ERP.Sage300.AR.Services

* + 1. TypeName Property

The ***TypeName***property identifies the type in the assembly to be invoked on the callback by the Process method.­

/// <summary>

/// The Process method of this type is called if there is anything to process

after report is generated

/// </summary>

public string TypeName { get; set; }

This property must be set for the callback to be invoked. An example:

Sage.CA.SBS.ERP.Sage300.AR.Services.Process.UpdateCustomerAfterPrinting

* + 1. ReportModel Property

The ***ReportModel***property identifies the model to be used in the *Process* method for the callback.

/// <summary>

/// Model - This is required if anything needs to be processed after the report

is generated.

/// </summary>

public ModelBase ReportModel { get; set; }

This property must be set for the callback to be invoked. An example:

new ShipToLocationExample(){ CustomerNumber = model.FromCustomerNumber }

* 1. Ship to Location Report Example

The previous section presented the properties of the *Report* class that must be set for a callback to occur when a report is either Loaded or Closed. This section will provide an example of the callback process using the A/R Ship to Location Report.

* + 1. ShipToLocationReportMapper Class

The report mapper class is where the Report class referenced above is instantiated and the various properties for the report are established. In the *Map* routine, notice the following properties that have been set to inform the reporting framework that a callback will be requested when the report is closed (generation has completed):

/// <summary>

/// Map a report

/// </summary>

/// <param name="model">Model to be converted to report</param>

/// <returns>Mapped Report</returns>

public Report Map(T model)

{

…

// In order to invoke a callback on the server when the report is done

// being generated, the following properties must be set

report.ReportProcessType = Common.Models.Enums.ReportProcessType.OnClose;

report.AssemblyName = "Sage.CA.SBS.ERP.Sage300.AR.Services";

report.TypeName = "Sage.CA.SBS.ERP.Sage300.AR.Services.Process.UpdateCustomerAfterPrinting";

report.ReportModel = new ShipToLocationExample(){ CustomerNumber = model.FromCustomerNumber };

…

* + 1. ShipToLocationExample Class

As noted in the *ShipToLocationReportMapper* class, the *ReportModel* property was assigned an object that will contain the customer number that will be “acted upon” when the report callback is invoked. Since this model does not exist, it must be created.

This class will be the parameter to the Process method and thus must inherit from ***ModelBase.***

Here is the class that was created for this example for us to pass the customer number:

// Copyright (c) 2021 Valued Partner All rights reserved.

#region Namespace

using Sage.CA.SBS.ERP.Sage300.Common.Models;

#endregion

namespace Sage.CA.SBS.ERP.Sage300.AR.Models.Process

{

/// <summary>

/// Partial class for ShipToLocation Example

/// </summary>

public partial class ShipToLocatioExample : ModelBase

{

public string CustomerNumber { get; set; }

}

}

As noted by the namespace, this class was created in the *Process* folder, but that is not required. It was done here since the service will also be created in the Process folder, but once again, this is not a requirement.

* + 1. UpdateCustomerAfterPrinting Class

As noted in the *ShipToLocationReportMapper* class, the *TypeName* property was assigned a type name that will contain the type that will be instantiated when the report callback is invoked. Since this class does not exist, it must be created.

This class must contain the ***Process*** method with a parameter of a class that inherits from ***ModelBase.***

Here is the class that was created for this example for us to pass the customer number:

// Copyright (c) 2021 Valued Partner All rights reserved.

#region Namespace

using Sage.CA.SBS.ERP.Sage300.Common.Models;

using Sage.CA.SBS.ERP.Sage300.Common.BusinessRepository.Utilities;

using Sage.CA.SBS.ERP.Sage300.AR.Interfaces.BusinessRepository;

using Sage.CA.SBS.ERP.Sage300.AR.Models;

using Sage.CA.SBS.ERP.Sage300.Common.Interfaces.Bootstrap;

using ShipToLocationExample = Sage.CA.SBS.ERP.Sage300.AR.Models.Process.ShipToLocationExample;

#endregion

namespace Sage.CA.SBS.ERP.Sage300.AR.Services.Process

{

/// <summary>

/// Update customer after printing

/// </summary>

public class UpdateCustomerAfterPrinting

{

private Context \_context;

/// <summary>

/// Constructor

/// </summary>

/// <param name="context"></param>

public UpdateCustomerAfterPrinting(Context context)

{

\_context = context;

}

/// <summary>

/// Process method is invoked by the reporting framework

/// </summary>

/// <param name="model">Model containing customer number</param>

/// <returns></returns>

public void Process(ShipToLocationExample model)

{

\_context.Container = BootstrapTaskManager.Container;

using (var customerRepository = Helper.Resolve<ICustomerEntity<Customer,

CustomerOptionalFieldValues>>(\_context))

{

customerRepository.UpdatePrintedFlag(model.CustomerNumber);

}

}

}

}

As noted by the namespace, this class was created in the *Process* folder, but that is not required.

The ***UpdatePrintedFlag*** is a fictional method that was created for this example.

* 1. The Server-Side Callback Framework

In the proceeding sections, the Report class properties were explained and the Ship to Location Report was modified to invoke a callback. This next section will present the routine that invokes the callback, if specified.

* + 1. The Process routine

The Process routine of Sage 300’s Export Report Controller is invoked for the report based upon the *ReportProcessType* value (Load, Close):

// Invoke additional processing if the report process type is set to OnLoad

if (report.ReportProcessType.HasFlag(ReportProcessType.OnLoad))

{

Process(report);

}

// Invoke additional processing if the report process type is set to OnClose

if (report.ReportProcessType.HasFlag(ReportProcessType.OnClose))

{

Process(report);

}

The Process routine will evaluate the other flags set to the Report class in the report’s mapper class to determine if a callback has been requested:

/// <summary>

/// This method does the processing before/after the report is generated.

/// </summary>

/// <param name="report">Report</param>

private static void Process(Report report)

{

// If report has not specified an assembly or type name, no callback has

been requested.

if (string.IsNullOrEmpty(report.TypeName) ||

string.IsNullOrEmpty(report.AssemblyName))

{

return;

}

// Instantiate the instance specified in the report object

var qualified = string.Format("{0}, {1}", report.TypeName,

report.AssemblyName);

var processService = Type.GetType(qualified, true);

var processObject = Activator.CreateInstance(processService, new object[] {

report.Context });

// Invoke the Process method of the specified type with the ReportModel

property

var mi = processService.GetMethod("Process");

mi.Invoke(processObject, new object[] { report.ReportModel });

}

At this time in the reporting framework, the method is hardcoded to ***Process***. However, in a future release, we will add a property to the ***Report*** class (i.e., MethodName) to remove this hardcoded value and let the developer assign the name that best describes the purpose of the callback.

1. Summary

This document presented the client-side callback that has been deprecated in the reporting framework as a result of our enhancement to the reporting framework to print a report directly to PDF.

It presented that a server-side callback mechanism has existed in the application and will be used to replace the deprecated client-side callback.

It presented properties in the *Report* class that are required to be set in order to provide for a server-side callback from the reporting framework.

It provided an example of a report that was modified to set the required properties for a call back from the reporting framework.