

Authors: Ben Mockford

Date: 20/02/2017

VERSION: 2.1

STATUS: PUBLISHED

CLASSIFICATION: CONFIDENTIAL

Local Collect Widget: Technical Implementation Guide

This document is the property of Mosaic Online Systems. If found, please return it to:

Information Security Team, Mosaic Online Systems, Unit A1 Yeoman Gate, Yeoman Way, Worthing, West Sussex, BN13 3QZ together with your name and address.

Letter postage need not be prepaid.

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Revision Date | Implemented by | Change |
| 0.1 | 16/09/2016 | Ben Mockford | Initial draft version |
| 0.2 | 23/09/2016 | Ben Mockford | Example CSS files added |
| 1.0 | 19/01/2017 | Ben Mockford | Final version. Revised pending Production deployment. |
| 2.0 | 16/02/2017 | Andrea Cooke | Final Version after RM requested updates. |
| 2.1 | 20/02/2017 | Carl Roberts | Clarification of Widget Config Object and addition of Information on editable attributes in CSS file |

**Document Sign Off**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Name | Dept/Role | Date | Comments |
| 0.2 | Carl Roberts | Project Manager | 27/09/2016 | Draft for Royal Mail Internal Use |
| 1.0 | Carl Roberts | Project Manager | 19/01/2017 | Approved |
| 2.0 | Carl Roberts | Project Manager | 16/02/2017 | Approved |
| 2.1 | Carl Roberts | Project Manager | 20/02/2017 | Approved |
|  |  |  |  |  |

**Document Distribution**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Name | Reason | Date |
| 0.2 | Kelly Holmes | Draft To Assist with Demonstrating Widget to other Royal Mail Staff | 27/09/2016 |
| 1.0 | Kelly Holmes | Final version, ready for publishing to Royal Mail customers | 19/01/2017 |
| 2.0 | Kelly Holmes | Final version, ready for publishing to Royal Mail customers | 16/02/2017 |
| 2.1 | Kelly Holmes | Final version, ready for publishing to Royal Mail customers | 20/02/2017 |
|  |  |  |  |

**Contents**

[1.0 Overview 3](#_Toc475358062)

[2.0 Implementation 4](#_Toc475358063)

[2.1 Embedding the Widget 4](#_Toc475358064)

[2.2 Displaying the Delivery Options 5](#_Toc475358065)

[2.3 Calling the Widget 5](#_Toc475358066)

[2.4 Displaying Local Collect Locations 7](#_Toc475358067)

[2.4.1 Grid View 9](#_Toc475358068)

[2.4.2 Map View 10](#_Toc475358069)

[2.4.3 Drop-down View 11](#_Toc475358070)

[2.5 Continuing with the Checkout Process 12](#_Toc475358071)

[3.0 Enrolling to use the Widget 12](#_Toc475358072)

[3.1 Overview 12](#_Toc475358073)

[3.2 Widget Configuration 12](#_Toc475358074)

[3.3 Browser Compatibility 14](#_Toc475358075)

[4.0 Error Handling 14](#_Toc475358076)

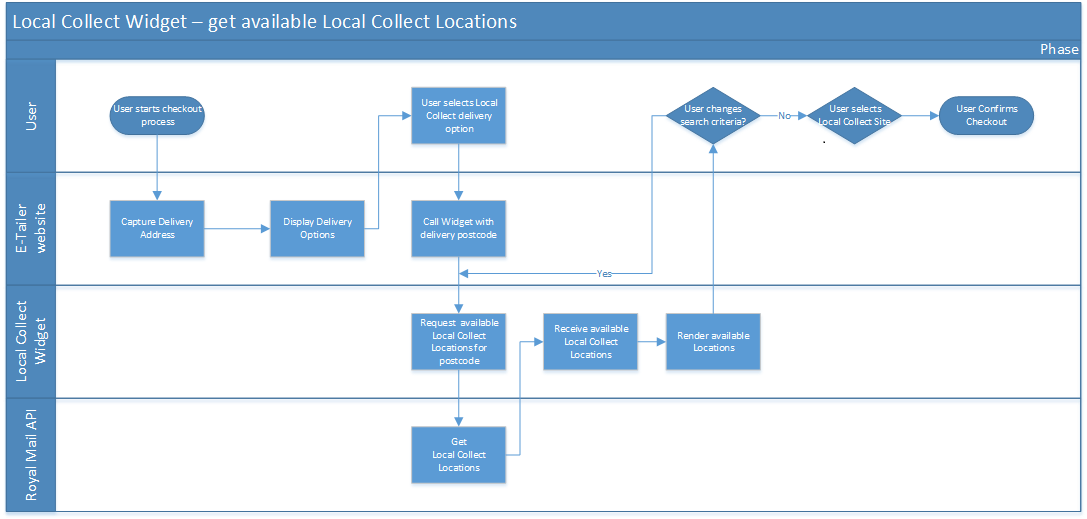
1. Overview

Royal Mail is looking to extend the reach of Local Collect by improving the integration options available to the web retailer to utilise the Local Collect capability. The current Royal Mail solution relies on the retailer’s ability to implement a web service integration with the Local Collect API’s exposed by Royal Mail.

In order to increase adoption, Royal Mail is therefore seeking an integration option for e-tailers using a Royal Mail ‘Local Collect Widget’ that can be simply embedded into the e-tailer’s website checkout process, which will integrate with the Local Collect API and remove intricacies of dealing with the API from the e-tailer.

The Widget will perform all the integration tasks such as branch look up and then return the selected Local Collect location to the e-tailer’s website as the delivery address where Local Collect is selected as the preferred delivery option.

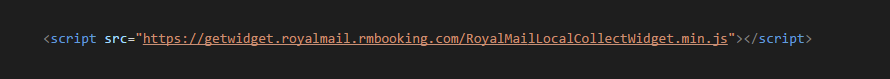
Below is an overview of the process from starting the checkout process to confirming their order:



1. Implementation
   1. Embedding the Widget

Royal Mail will provide you with all the access details including the relevant portal information to allow you to follow the implementation process below.

The Widget will be embedded within the Customer website by including a reference to a JavaScript external component, and styling sheet as follows:





The web server on which the e-tailer’s website is hosted will automatically retrieve the Local Collect Widget from the servers when it needs to be used to generate a website impressing using functionality provided by the Local Collect Widget.

**In order to determine how Local Collect information will be presented to the user, a named placeholder (div) is inserted into the web page by the e-tailer. This will be the anchor for any additional information provided by the Widget to the user.**

<body>

…

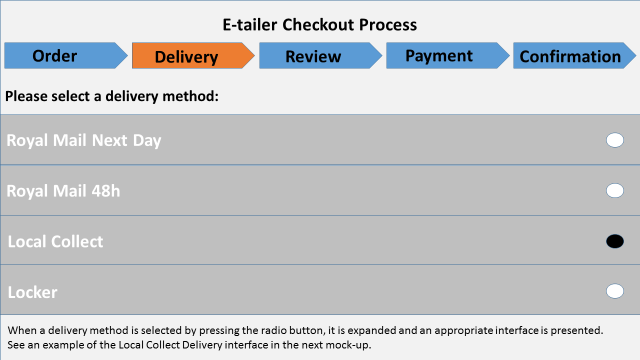
<div id='royalMailLocalCollectPlaceHolder'></div>

…

</body>

* 1. Displaying the Delivery Options

In order to utilise the Widget, the e-tailer will need to extend the list of delivery options presented to the user, to include Local Collect as a valid delivery option. The manner in which the available delivery option(s) are presented to the user will vary between websites, and may be controlled by radio buttons, drop down lists etc.



* 1. Calling the Widget

Once the ‘Local Collect’ delivery option has been added to the website delivery options by the e-tailer, selecting this delivery option should fire an additional event ‘localCollectSelected’ to instruct the Local Collect Widget to render additional information to the user.

<body> …

<input type="button"  value="Load Widget" onclick="localCollectSelected(widgetConfigObj)viewingOptions:['Grid']" ( {

{

    royalMail: { getRMId: function () { return 'MyId' },

getRMKey: function () { return 'MyKey' } },

 getMapsKey: function () { return 'MyKey' },

onLocationSelected : function(localCollect) // a call back when a Local Collect

{ // location has been successfully selected

console.log(location.organisationName);

},

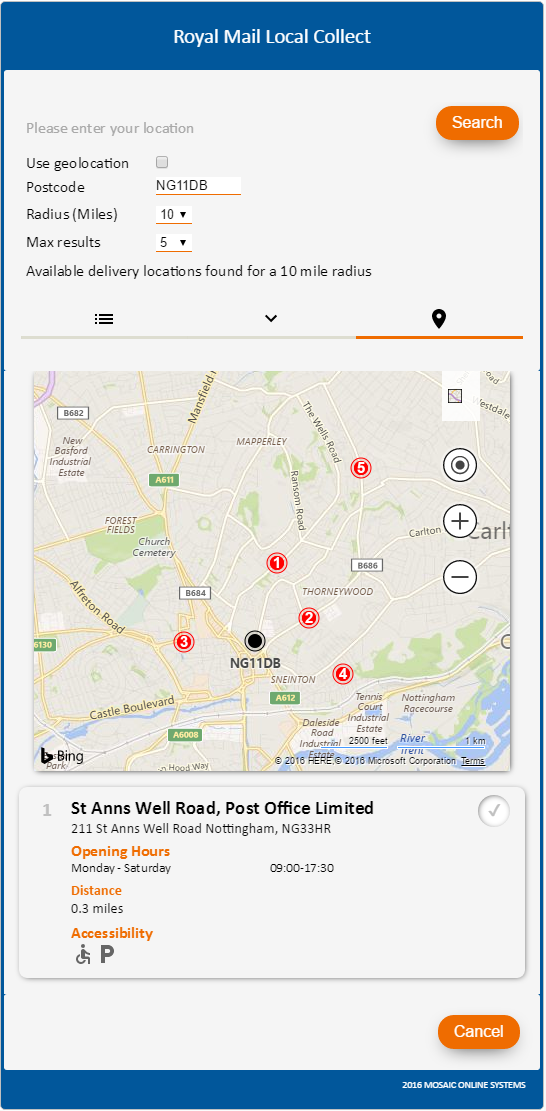
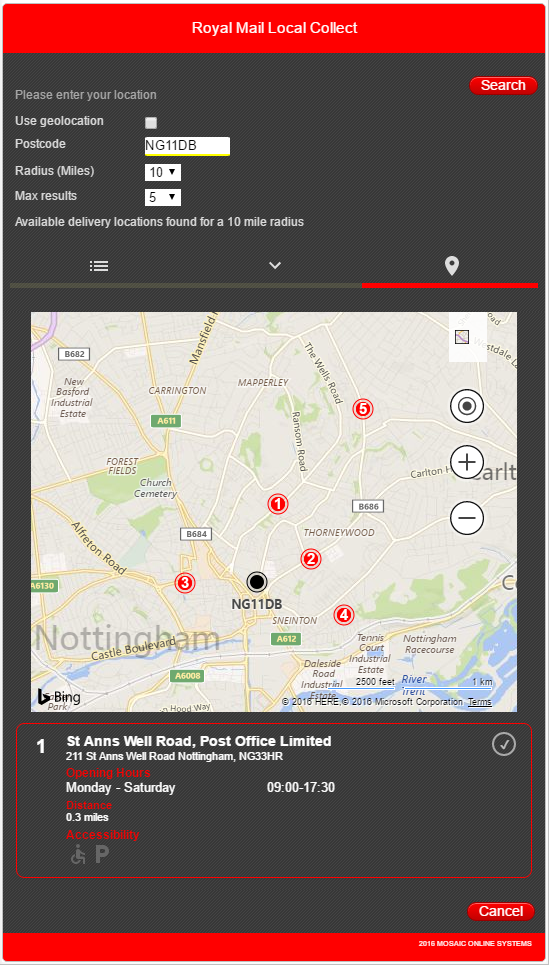
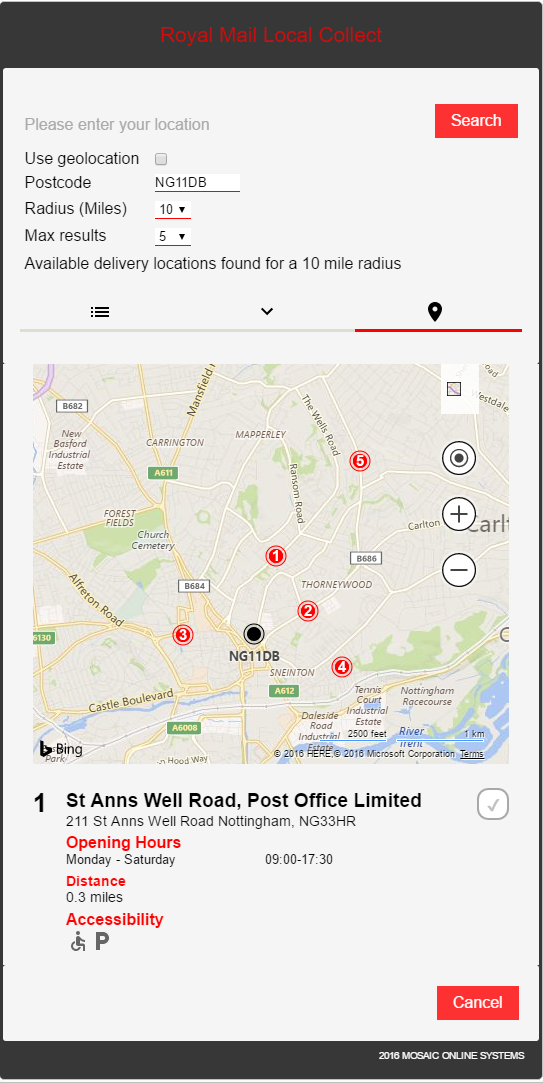
onCancelSelection : function() {/\*a call back in case of cancelation\*/}});"/>

…

</body>

In response to this ‘localCollectSelected’ method being called, the Widget will render additional controls onto the user’s web page, within a named div (<div id='royalMailLocalCollectPlaceHolder'></div>) defined by the e-tailer’s website as per the example on the bottom of page 4.

The manner in which the information is presented will be controlled by a CSS stylesheet. In addition to the provided themes (blue, dark and red theme), the e-tailer can alter this CSS file to maintain same look and feel between the Widget and the e-tailer’s website.

Below are some sample CSS files for each of the above themes. These CSS files can be edited and embedded in to the e-tailers website to customise the appearance of the widget.

Althought there are clearly a number of classes that can be edited in the CSS file, it is recommended that the following only are changed, to avoid any detrimental effects on the Widget’s performance:

|  |  |
| --- | --- |
| Class Name | Description |
| .widgetTitle | The colour of this class can be amended. |
| #notificationRegion | The colour of this class can be amended. |
| .widgetInput | The colour of this class can be amended, as can border width. |
| .widgetSelect | The colour of this class can be amended, as can border width. |
| .widgetbutton | The colour of this class can be amended, as can font type, colour and size. |
| .widgetbutton:hover | The colour of this class can be amended. |
| .widgetbutton:disabled | The colour of this class can be amended. |
| .disabled | The background colour of this class can be amended. |
| .widgetPlaceHolder | The background and border colour of this class can be amended, as can font type, colour and size. |

In addition to the styling customisation, other facilities will be provided to enable the e-tailer to apply more personalisation including:

* Default search parameters like default radius and default maximum number of results.
* The used terms such as the Widget title, the headings, etc.
* Viewing options such as grid, drop down list or maps.

As part of the parameters passed into the Widget by the triggering event will be the recipient’s delivery address. This will be used by the Widget to determine the default search area for identifying Local Collect locations. Additional parameters that can be passed in are the planned delivery day and search radius. If these are not specified then they will be defaulted to the day after the current date, and a search radius of 10 miles.

The Widget will then contact the Royal Mail Local Collect REST API and will perform a GET against the LocalCollectOffices endpoint, specifying the recipient’s postcode, the planned delivery date, and the search radius.

The call to the API will also pass in the appropriate authentication headers [See section on authentication].

* 1. Displaying Local Collect Locations

Once a response is received from the Local Collect API, the Widget will cache the returned details, including the booking references, and will then render the results within the named div on the e-tailer’s web page. Options for the manner in which the results will be presented are:

* Tabular form
* Drop down list with detail panel
* Map view with detail panel

The manner in which the results will be presented can be controlled by the e-tailer’s website, so that the e-tailer can select the most appropriate approach for the website and available screen real estate.

The e-tailer would need to create a Javascript object with the widget configuration settings 'widgetConfigObj', as per the below example:  
  
<input type="button" class="etailerbutton" value="Select a local collect" onclick="localCollectSelected(widgetConfigObj)">   
  
The e-tailer would then need to specify the relevant 'viewingOptions':  
['Grid'] or  
['DropDown'] or  
['Maps']

An example of the Widget Config Object is below:

<script> …

widgetConfigObj = {

viewingOptions:['Grid'],

royalMail: { getRMId: function () { return '4d3f61bd-babf-4849-93d0-ce91f06d78c1' } },

deliveryPostCode: 'NG11DB',

defaultRadius: 10,

defaultNbrOfResults: 10,

onOpenWidget: function () {

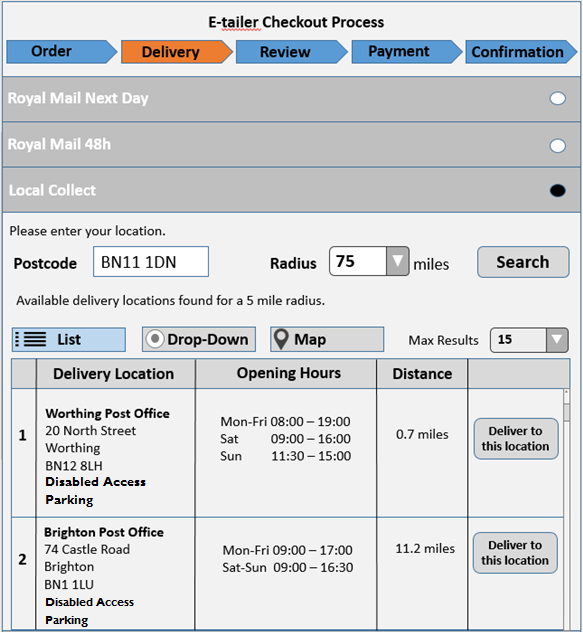
document.getElementById('checkout').disabled = true;

}…

|  |  |
| --- | --- |
| Class Name | Description |
| viewingOptions | ['Grid'],or ['DropDown'], or ['Maps'] |
| royalMail | This is the ‘Client-ID’ obtained from <https://developer.royalmail.net/api> |
| deliveryPostCode | This will need to be populated with Post Code details entered, by the e-tailer’s customer. |
| defaultRadius | Delivery and Enquiry Offices within this radius from the designated Post Code will be displayed by the Widget. The default and recommended radius is 10 miles. |
| defaultNbrOfResults | This is the number of Delivery and Enquiry Offices within the designated radius that will be displayed by the Widget, the default is 10. |

* + 1. Grid View

In the event that a grid view is selected, then the number of rows to be displayed can be controlled by the e-tailer. Each row will include the Local Collect location name and address, opening hours, distance from the recipient’s postcode, and a ‘Deliver to this location’ button to enable selection. The returned delivery locations will be numbered ascendingly by distance, showing the nearest locations at the top of the list.



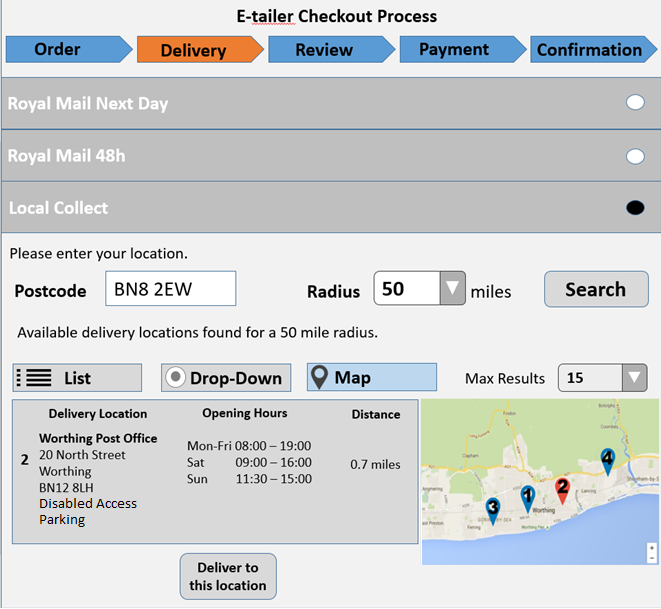
If the results cannot all be displayed within the web page, a user will have the option of viewing further results e.g. by using a scrollbar.

* + 1. Map View

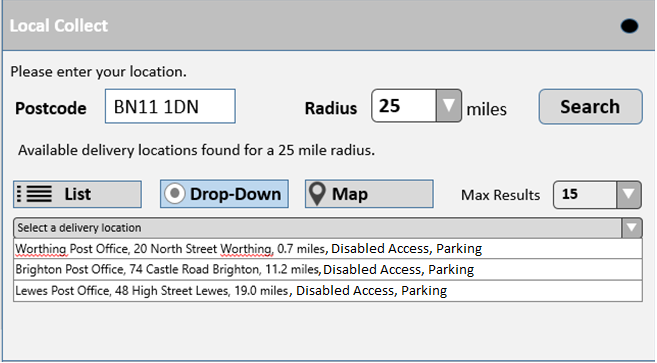
In the event that a map view has been requested, then the locations will be displayed as Pins on a map, and selecting a pin will display details of that Local Collect location in the detail panel, including location name and address, opening hours, distance from the recipient’s postcode and ‘Deliver to this location’ button to enable selection.

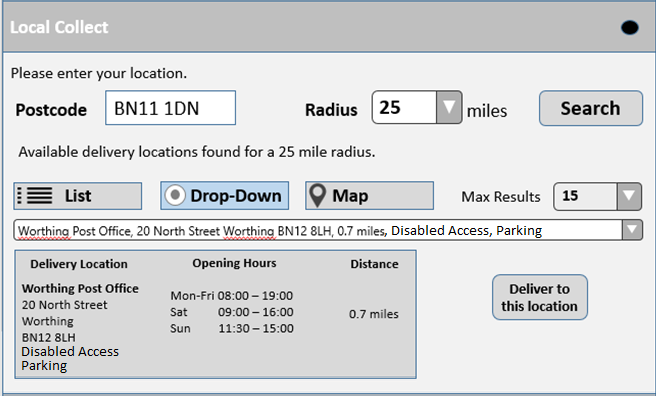
The pins will be numbered ascendingly by distance. The map will be centralised to focus on the nearest delivery location and the zoom will be auto-adjusted to present all available locations in the map view.

The map will support interactive navigation and zoom adjustment.



* + 1. Drop-down View

In the event that the user is space constrained on the checkout page, then a third option of listing the available Local Collect locations in a drop-down list will be available. In this option, the drop-down list will include the location name, partial address and distance from the search postcode, and a ‘Select a delivery location’ option. 

When the user selects a particular location in the drop down list, the detail panel will be populated with the location name, full address, opening hours and a ‘Deliver to this location’ button.

The user will have the option to adjust the search radius, and also the postcode used for the search. This will enable the poster to look for available Local Collect locations that are near to an address other than the recipients address (e.g. Local Collect locations near work).

* 1. Continuing with the Checkout Process

When the user is ready to proceed with the checkout process, the user will interact with the parent website in some manner (e.g. automatically move to the next step, clicking on Next button, Confirm Order etc). As this interaction will be with the parent website rather than the Widget itself, the e-tailer’s website must check the state’s consistency of the selected delivery option. Once the call back is triggered the retailer must retrieve the location data and store within the customer order to create the label.

Checking the state of validity is very simple when Local Collect is the actual selected delivery option. In fact, while configuring the widget, the e-tailers website provides the widget with two functions to deal with the local collect selection and cancelation. In this two functions (call-backs from the widget), the website should check the state of validity of the checkout form. Then, enable/disable the ‘Checkout’ button depending on that.

1. Enrolling to use the Widget
   1. Overview

The Royal Mail Customer Take on Team will provide a welcome pack with all relevant urls’

In order to enrol a new e-tailer client onto the Local Collect Widget, a number of steps need to be completed. Each client needs to register in order to obtain Royal Mail Local Collect login credentials. An e-tailer can easily embed the Local Collect Widget into their webpage and customize the Widget in order to adapt the application’s functionality to the e-tailer’s unique checkout process.

The Widget can be embedded into an e-tailer’s website by simply inserting a reference into the checkout process of the webpage. The source code of the Widget will be downloaded from the servers and subsequently appear within the e-tailer’s website.

* 1. Widget Configuration

The Widget exposes a number of configuration settings, allowing an e-tailer to control how the Widget is presented within their checkout process. Where the configuration settings are not specified, default values will be used.

An e-tailer can also expose a number of input parameters to allow the Widget to fully utilise the Royal Mail Local Collect API functionality. An e-tailer can enable their website to expose the configuration settings and map the parameters to the as a set of options exposed by the Widget. The available configuration settings and input parameters are listed below:

|  |  |  |
| --- | --- | --- |
| Configuration / Parameter | Description | Default (if not specified) |
| Client ID | A valid Client ID must be provided for authentication. | n/a |
| Bing key | A valid Bing key must be provided to the widget to be able to use Bing Maps. | n/a |
| Postcode | The recipient’s postcode. Where available, the Postcode element will be used for the initial delivery location search. | n/a |
| Default View | The views available to the end-user. | Grid |
| Views | Controls which views are presented to the user. | Grid, Map, Drop-down |
| Default Radius | The default radius for the search (miles). Must be between 1 and 99. | 10 miles |
| Show Radius | Boolean configuration which determines if the ‘Radius’ filter will be shown on the screen. Available options: Enabled/Disabled. When enabled, the ‘Radius’ drop-down will be shown. When disabled, it will be hidden and the Widget will only use the default radius value. | Enabled |
| Radius Values | Provides the e-tailer an option of specifying the selectable values in the ‘Radius’ drop-down. | 1, 5, 10, 20, 50, 75, 99 |
| Default Max Results | The default number of maximum results to show in the output. | 10 |
| Max Results Values | The values available in the ‘Max Results’ drop-down. | 1, 5, 10, 15, 20 |
| The used terms | Any shown term on the widget can be replaced by a different one provided by the e-tailer website. | Default terms |

**Widget Styling**

Please see example below.

script> …

widgetConfigObj = {

viewingOptions:['Grid'],

royalMail: { getRMId: function () { return '4d3f61bd-babf-4849-93d0-ce91f06d78c1' } },

deliveryPostCode: 'NG11DB',

defaultRadius: 10,

defaultNbrOfResults: 10,

onOpenWidget: function () {

document.getElementById('checkout').disabled = true;

}…

…

* 1. Browser Compatibility

|  |  |  |
| --- | --- | --- |
|  | Desktop | Mobile |
| [JQuery](https://jquery.com/browser-support/) | *Internet Explorer*: 9+ *Chrome*: supported, *Edge*: supported *Firefox*: supported *Safari*: supported *Opera*: supported | *Stock browser* onAndroid 4.0+, *Safari* on iOS 7+ |
| [Bootstrap](http://v4-alpha.getbootstrap.com/getting-started/browsers-devices/) | *Internet Explorer*: 9+ *Chrome*: supported, *Edge*: supported *Firefox*: supported *Safari*: supported *Opera*: supported | Stock browser on Android 5.0+ *Safari* on iOS |
| *Use of http custom-headers*   * *X-IBM-Client-Id* | *Internet Explorer*: 10+ *Chrome*: supported, *Edge*: supported *Firefox*: supported *Safari*: supported *Opera*: supported | Stock browser on Android 5.0+ *Safari* on iOS |

The Widget will do a browser detection, and where possible will degrade gracefully, indicating that the Local Collect feature is unavailable in this browser, rather than completely disrupting the customer’s checkout process.

1. Error Handling

The Local Collect Widget will handle error conditions gracefully so that the customer’s checkout experience is not adversely disrupted when an error occurs. Below are some errors that may appear along with the associated behaviour that the error would generate:

|  |  |
| --- | --- |
| Error | Error behaviour |
| Widget not supportable in users browser version | Notify user that Local Collect service is not available on this browser version and disable Widget. |
| Local Collect API cannot be contacted. | Notify user that Local Collect service is currently unavailable, and to either retry or select another delivery option. |
| Local Collect API ClientID refused. | Notify user that Local Collect service is currently unavailable, and to select another delivery option. |
| Booking timed out | Retry request to refresh booking timeout. |
| Local Collect site originally requested now unavailable | Notify user that the Local Collect location now no longer available, and to select a different Local Collect location. |
| Booking refused | Notify user that the Local Collect booking was unsuccessful, and to either retry, or select another delivery option. |