Custom Web Store Integration Points - Hook ShipRush Web to Your Site, Store or App

Like revision histories? So do we! Find it at the end of this document.

- 1. Custom webstores
  - 1.1. Push / Custom-Post-In Diagram
  - 1.2. Custom Web Store - Polling Architecture Chart
- 2. Custom-Post-In: Setup & API
  - 2.1. So What do you Post There?
  - 2.2. Once an order is in ShipRush, how to update it?
  - 2.3. Order Status Explanation (shipped, unshipped, paid, cancelled)
  - 2.4. Shipped/Unshipped/Cancelled
    - 2.4.1. Order based:
    - 2.4.2. Shipment based:
    - 2.4.3. Paid status
    - 2.4.4. Deleted/Cancelled Statuses
      - 2.4.4.1. Cancelled Flag
      - 2.4.4.2. Deleted Flag
  - 2.5. Tagging Shipments
    - 2.5.1. Tags in Action
  - 2.6. Geting tracking numbers from My.ShipRush to your system
    - 2.6.1. Sending in a PostBackURL
    - 2.6.2. Quick Steps to see PostbackUrl in action
    - 2.6.3. My.ShipRush API Users: Steps to see PostbackUrl notification working with interactive and online tools only (no modifications to your server code required)
  - 2.7. Setup Workflow
  - 2.8. Sample HTTP POST (HTTP POST headers and body)
    - 2.8.1. Headers
    - 2.8.2. Order Sample: With order details
    - 2.8.3. Order Sample: The Bare Minimum Order
      - 2.8.3.1. Error Response
- 3. Polling API
  - 3.1.1. getordersbydate
  - 3.1.2. ping
    - 3.1.2.1.1. getserverinfo
    - 3.1.2.1.2. updateshippinginfo
- 4. FAQ
  - 4.1. Can you send me a javascript example for post out and post back?
- 5. Document Revision History

Terms

- **Developer** That is you, the developer who is consuming the ShipRush Web Non Visual API in a web or desktop application
- **"Your Application"** is the app you (the developer reading this) are responsible for. This API supports both desktop and browser applications.
- **Client Application or Client App** same as above
- **Customer or User** This is your customer or user. AKA "end user"
- **HTTP REST** is a standard way of presenting an API on the internet. RFC's, details, etc are discussed in many places on the internet, including wikipedia http://en.wikipedia.org/wiki/Representational_State_Transfer
- **Fiddler** Is a free tool that is very useful for working with and understanding a REST API. It is available at http://www.fiddler2.com/fiddler2/ (Windows)
  - Alternate tool: HTTP Analyzer (Windows)
  - Alternative tool: curl (cross platform)
- **Date Time format**: My.ShipRush uses the ISO 8601 Date/Time Format http://en.wikipedia.org/wiki/ISO_8601

1. Custom webstores

Developers and users who have custom (or heavily customized) systems can interact with ShipRush two ways:

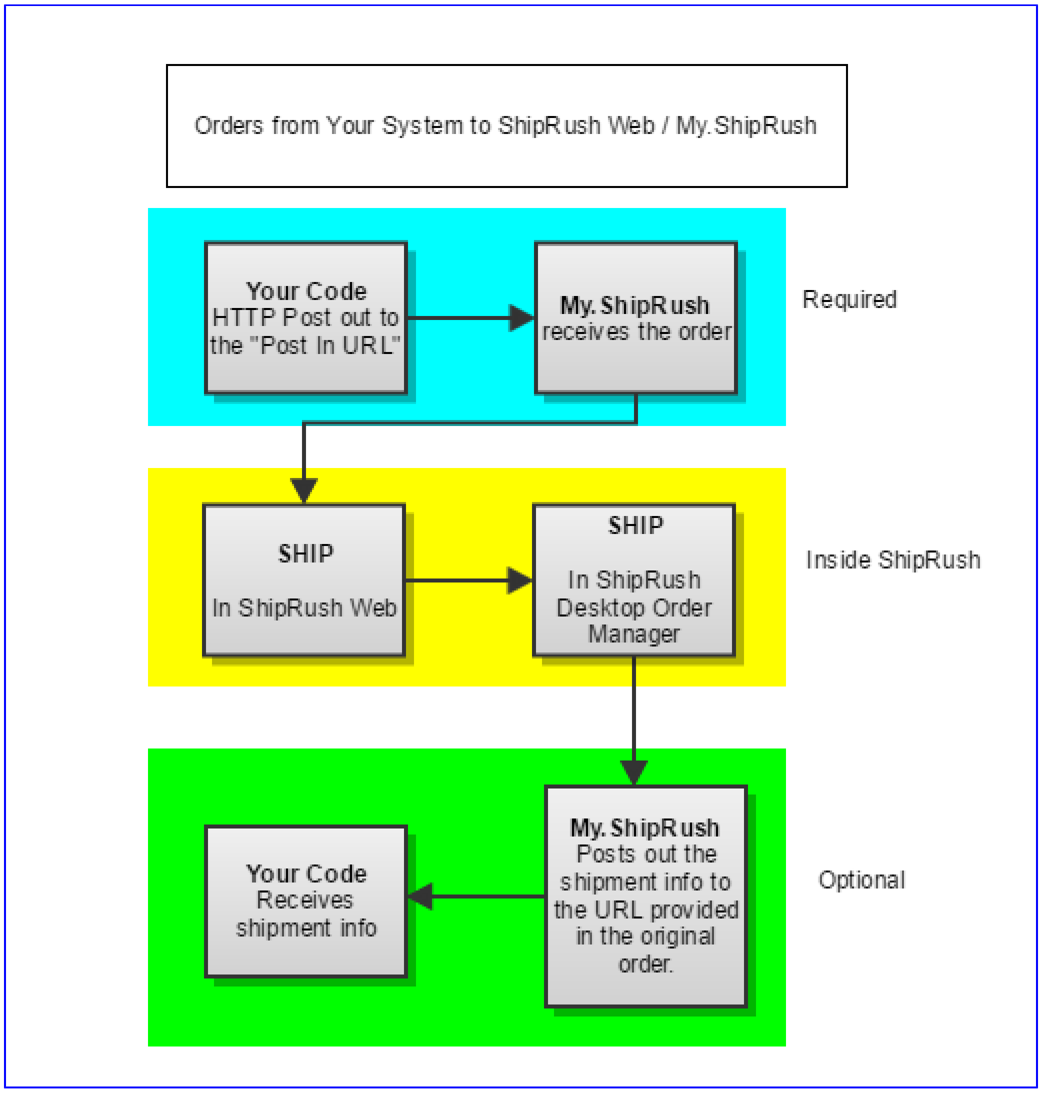
1. "Push" based "Custom Post In" method
2. "Polling" based Custom Web Store method

Both methods expect you to format data in XML. The format of the shipment XML follows the standard ShipRush "core" format. This is the same format used in ShipRush zpk files and the ShipRush SDK. Details can be found in:

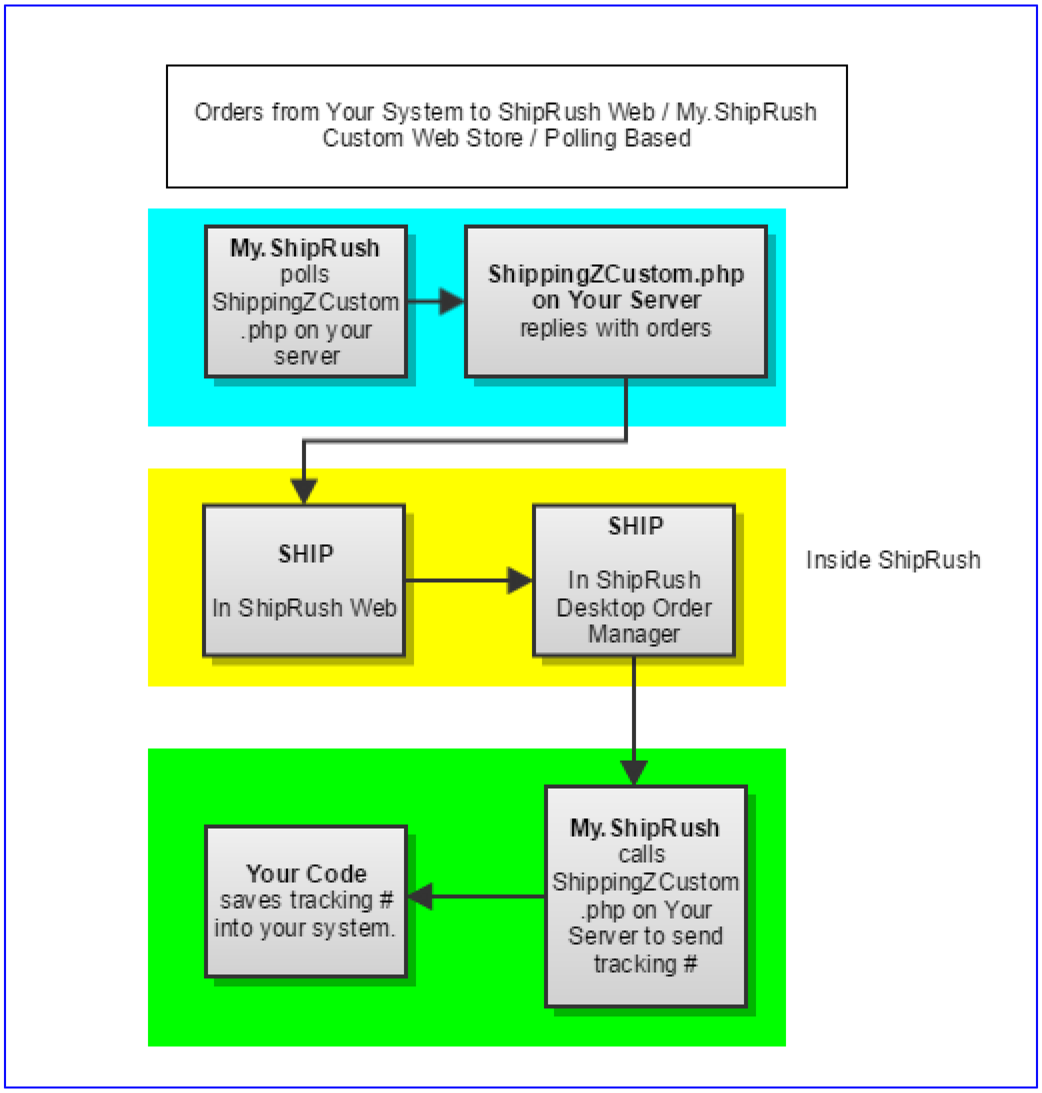
- Using the ShipRush UI to save zpk templates
- The ShipRush Browser Plugin documentation (aka "Clicky," in ShipRush_Developer_Clicky.chm)
- Shipment XSD (ShipRushShipment_Redist.xsd)
    - The XSD also has constants and enums for the various shipping carriers, packaging types, service options, etc.
- ShipRush.SDK.DesktopDemo demo application that can post sample order to PostIn URL
- My.ShipRush PHP webstore integration (including sample Custom Integration) http://my.shiprush.com/assets/ShipRush_ShoppingCart_Integration_Kit__See_README_file.zip

⚠ Reference to the documentation is needed when passing order line items and detailed shipping information (such as shipping service). For super-high level automation, the minimum elements can be sent. See below for detail.

1.1. Push / Custom-Post-In Diagram


Orders from Your System to ShipRush Web / My.ShipRush
Your Code HTTP Post out to the "Post In URL"
My.ShipRush receives the order
Required
SHIP In ShipRush Web
SHIP In ShipRush Desktop Order Manager
Inside ShipRush
Your Code Receives shipment info
My.ShipRush Posts out the shipment info to the URL provided in the original order.
Optional

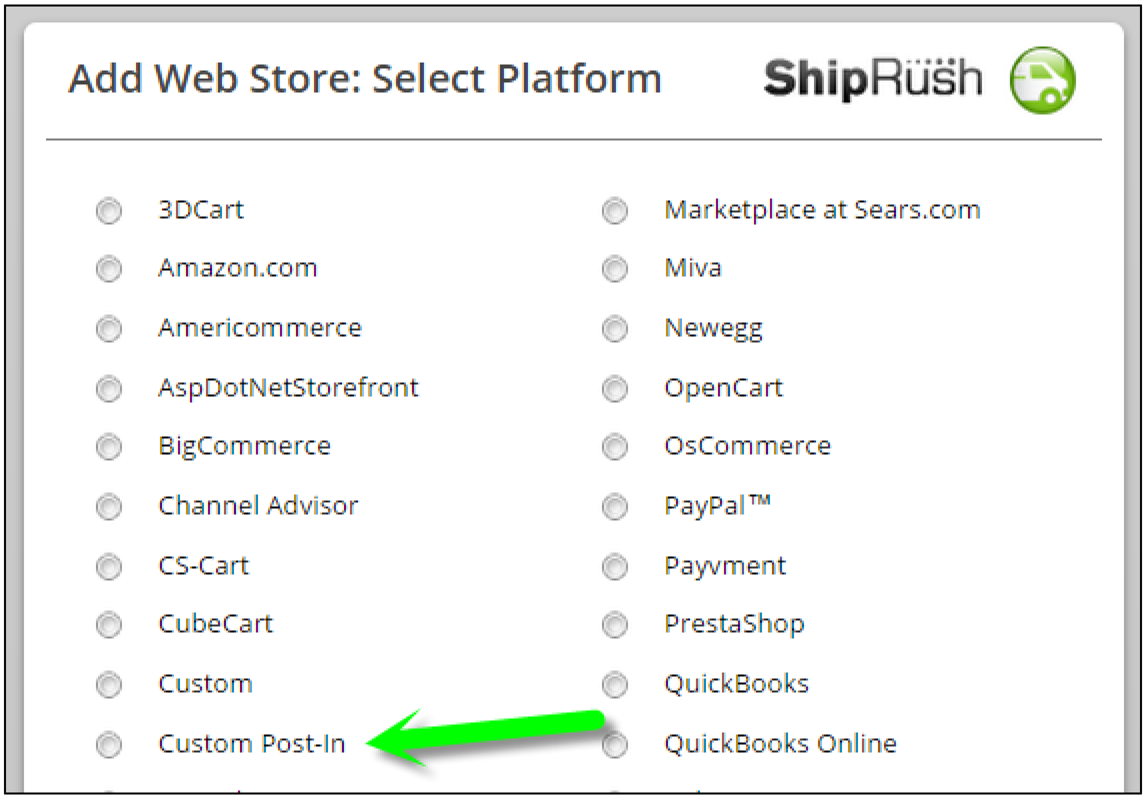
1.2. Custom Web Store - Polling Architecture Chart


Orders from Your System to ShipRush Web / My.ShipRush Custom Web Store / Polling Based
My.ShipRush polls ShippingZCustom.php on your server
ShippingZCustom.php on Your Server replies with orders
SHIP In ShipRush Web
SHIP In ShipRush Desktop Order Manager
Inside ShipRush
My.ShipRush calls ShippingZCustom.php on Your Server to send tracking #
Your Code saves tracking # into your system.

2. Custom-Post-In: Setup & API

Webstore type: **postin**

In ShipRush, when adding a web store, choose Custom Post In as shown below.



With Post-In API you/your developer pushes orders into My.ShipRush via HTTP POST. During **Custom Post-In** webstore set up (the screens that come later in the wizard shown above) you are given a unique URL to this effect:

```
https://api.my.shiprush.com/IntegrationService.svc/XXqoomtXOU28wJyUAPgFig/UtCQqjsGeU6l###
#######/order/add
```

This URL is your username, password and a place to post new sales to My.ShipRush. For each new sale on your site your code would HTTP POST a ShipRush-compatible XML to this URL.

This URL is a password. Store it in encrypted storage. It should not be displayed in clear in your application. In your logging system, it should be masked so as not to expose these elements.

2.1. So What do you Post There?

See below for example order XML.

2.2. Once an order is in ShipRush, how to update it?

If the order is cancelled / deleted / changed on your system, and you wish to update ShipRush, simply post it into ShipRush again, with a different status.

2.3. Order Status Explanation (shipped, unshipped, paid, cancelled)

2.4. Shipped/Unshipped/Cancelled

In the case of a My.ShipRush consuming developer, **focus on the Order** and ignore the information below RE shipment.

2.4.1. Order based:

For "payload" which has BOTH Shipment and Order the information about shipping status will be duplicated in 2 places

To put an order in the Ready to Ship folder: Set: Shipment.ShipmentType == Pending AND Order.IsShipped == FALSE

To put an order in the Shipped folder: Set: Shipment.ShipmentType == History AND Order.IsShipped == TRUE

2.4.2. Shipment based:

A Shipment can be Shipped or Unshipped.

We suggest to always send: Shipment.ShipmentType == Pending

As this will put orders in the Ready to Ship folder.

- Shipped (the Shipped folder): Shipment.ShipmentType == History
- Unshipped (Ready to Ship folder): Shipment.ShipmentType == Pending

2.4.3. Paid status

Payment status only applies to the Order. Shipment has no concept of "payment". Information about payment status always comes via Order.IsPaid

When sending orders to ShipRush, there are two ways to handle data that is not paid:

- **Ignore it**: Do not send to ShipRush orders that are not paid.
- **Flag it**: Send unpaid orders to ShipRush, which will now appear in the Unpaid area.
- **Recommended:** Only send orders to ShipRush that are paid and "ship able"

2.4.4. Deleted/Cancelled Statuses

These are different for Shipment and Order. We suggest to **focus on the Order** and ignore the information below RE shipment.

Since "Deleted" and "Cancelled" represent different concepts - it is possible to have combination of Order/Shipment where Cancelled and Deleted do not match.

2.4.4.1. Cancelled Flag

The Cancelled flag always comes from you by updating the order (there is no way to modify it in My.ShipRush) and is controlled via the Order.IsCancelled field.

To update an order in ShipRush, post in with: Order.IsCancelled == True

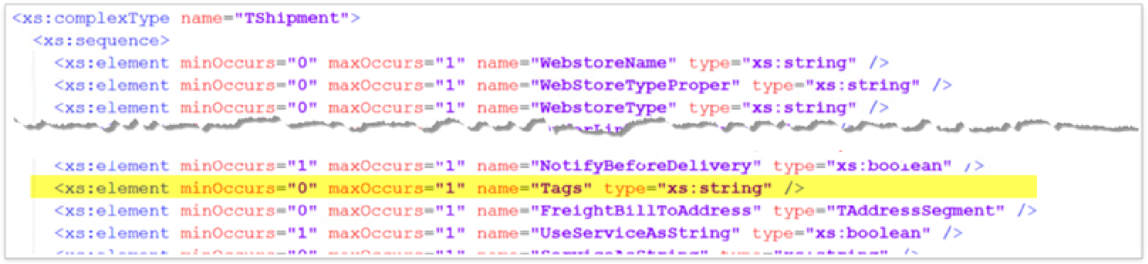
When the shipper refreshes the list of shipments, the shipment will move to the Cancelled folder.

2.4.4.2. Deleted Flag

The "Deleted" flag applies to the Shipment. It indicates that the user of ShipRush deleted a shipping label that he already printed. The Deleted flag is available via Shipment.ShipmentStatus == Deleted.

2.5. Tagging Shipments

Shipments coming into ShipRush can be tagged by using the "Tags" element of the TShipment payload



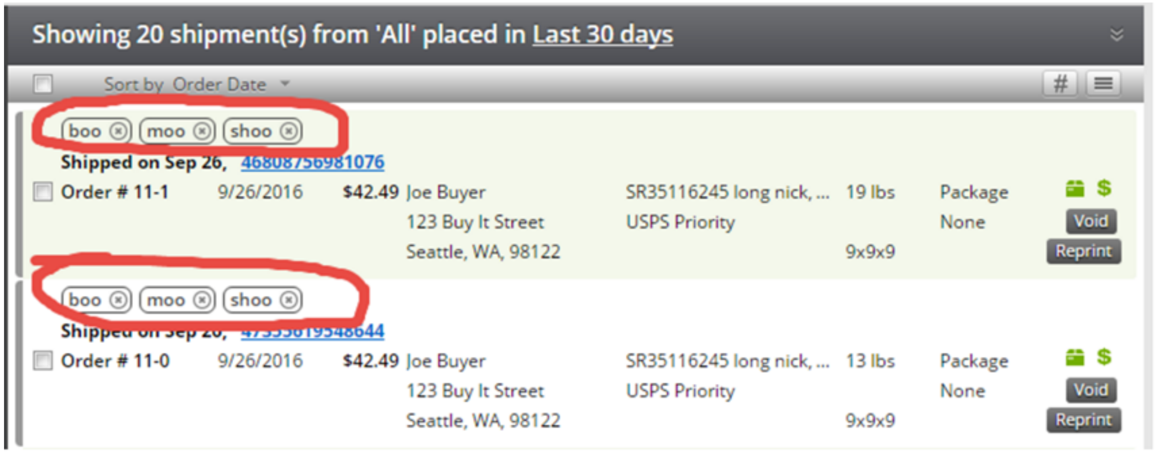
```
<xs:complexType name="TShipment">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="WebstoreName" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="WebStoreTypeProper" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="WebstoreType" type="xs:string" />

    <xs:element minOccurs="1" maxOccurs="1" name="NotifyBeforeDelivery" type="xs:boolean" />
    <xs:element minOccurs="0" maxOccurs="1" name="Tags" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="FreightBillToAddress" type="TAddressSegment" />
    <xs:element minOccurs="1" maxOccurs="1" name="UseServiceAsString" type="xs:boolean" />
```

Tags should be comma or space-separated.

The total length of all tags should not exceed 100 characters. If it does they are truncated at 100.

2.5.1. Tags in Action

The shipment grid can be filtered by tag. This shows the tags visually:



Showing 20 shipment(s) from 'All' placed in Last 30 days

Sort by Order Date

boo moo shoo

Shipped on Sep 26, 46808756981076

Order # 11-1	9/26/2016	$42.49	Joe Buyer 123 Buy It Street Seattle, WA, 98122	SR35116245 long nick, ... USPS Priority	19 lbs 9x9x9	Package None

boo moo shoo

Shipped on Sep 26, 47335619548644

Order # 11-0	9/26/2016	$42.49	Joe Buyer 123 Buy It Street Seattle, WA, 98122	SR35116245 long nick, ... USPS Priority	13 lbs 9x9x9	Package None

2.6. Geting tracking numbers from My.ShipRush to your system

There are two way you can get tracking number from My.ShipRush to your system.

- Subscribe to shipping notifications by providing a notification URL (on your server) via Shipment.PostbackUrl
  - This is simple and automatic. When you push the order into ShipRush, include a url (in the Shipment.PostbackUrl element) and ShipRush will push ShipRush xml to that url when there is a tracking number to report.
- Advanced, harder way: My.ShipRush Ecommerce Gateway (aka the My.ShipRush API): This lets you poll for tracking numbers. (Push notifications can also be configured.) However this is a different code path and requires significant additional development by you.
  - NOTE: This requires full use of the My.ShipRush API, which is significant additional effort. (However it does give more global control.)

2.6.1. Sending in a PostBackURL

My.ShipRush will do an HTTP post to the provided URL with the full Shipment/Order data (with <Request> as root XML element - same format you used for posting data to My.ShipRush). We recommend using online HTTP-post catching service like http://requestb.in/ for testing PostbackUrl functionality.

My.ShipRush API users: ShipRush.Test.GUI SDK demo shows how to add a shipment with a PostbackUrl (under "Shipments and orders" tab)

Note: For shipments processed with Web Shipping there will be at least 2 notifications coming to PostbackUrl. One is instant and another one in a few seconds. If your server is down My.ShipRush will keep retrying for 2 days every few hours. For shipments processed in ShipRush desktop there will be at least 1 notification a few seconds after operation completed. (For shipments marked via UpdateShipment REST API in the My.ShipRush API, the post out is within a few seconds of the shipment.)

The max length for a postbackurl is 55 characters

Make sure to pad PostbackUrl with some kind of authentication token to prevent someone else posting invalid data to your system. We strongly suggest that this be a dynamic, changing token. NOT a global token.

Good Example: hash order # + dynamic password (for example, each 10 day window of time generates a new pw)

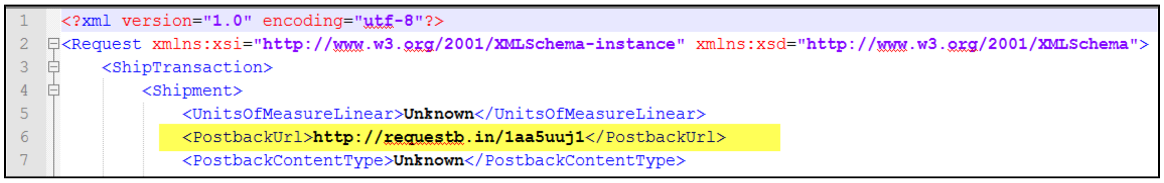
Weak Example: hash the Order # + salt: (problem: The salt is constant across all orders, hack once, hack all)

- order 234234234: https://my.systemsite.com/updateorder/ABDD003BDDD1F3DD
- order 234234235: https://my.systemsite.com/updateorder/37237DEFFACC8238

Bad Example (please do not emulate!)

- order 234234234: https://my.systemsite.com/updateorder/234234234
- order 234234235: https://my.systemsite.com/updateorder/234234235

Best Practice: The PostbackUrl, on your system, should expire. E.g. after some number of days (say 7), it should not allow any input.



```
<?xml version="1.0" encoding="utf-8"?>
<Request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <ShipTransaction>
        <Shipment>
            <UnitsOfMeasureLinear>Unknown</UnitsOfMeasureLinear>
            <PostbackUrl>http://requestb.in/1aa5uuj1</PostbackUrl>
            <PostbackContentType>Unknown</PostbackContentType>
```

```
<?xml version="1.0" encoding="utf-8"?>
<Request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <ShipTransaction>
  <Shipment>
   <PostbackUrl>http://requestb.in/1aa5uuj1</PostbackUrl>
   ...
   <Package>
    <PackageActualWeight>7</PackageActualWeight>
    <PackagingType>02</PackagingType>
    <PackageReference1>Order # 61-0</PackageReference1>
    ...
    </Package>
   <DeliveryAddress>
    <Address>
     <FirstName>Joe</FirstName>
     <LastName>Buyer</LastName>
     <Address1>123 Buy It Street</Address1>
     <City>Seattle</City>
     <State>WA</State>
     ...
     </Address>
   </DeliveryAddress>
   <ShipperAddress>
    <UPSAccountNumber>123555</UPSAccountNumber>
    <Address>
     <FirstName>Alex</FirstName>
     <LastName>Test</LastName>
     <Company>ACME</Company>
     <Address1>120 Lakeside ave</Address1>
     ...
     </Address>
   </ShipperAddress>
  </Shipment>
  <Order>
   <BillingAddress>
    <FirstName>Joe Bogus</FirstName>
    <Company>Z-Firm LLC</Company>
    <Address1>120 lakeside ave</Address1>
    ...
   </BillingAddress>
   <ShippingAddress>
    <FirstName>Joe Bogus</FirstName>
    <Company>Z-Firm LLC</Company>
    <Address1>120 lakeside ave</Address1>
    ...
```

2.6.2. Quick Steps to see PostbackUrl in action

Using systems like requestb.in exposes you to risk in that any data sent to such an endpoint is public. For example, if you send a live order there, you may be exposing to the public the real name and PII (Personally Identifiable Information) of a customer. If the payload were to include a postback url or the Custom Post In Url, those urls contain passwords / security elements that should not be exposed.

Please think it through and use caution.

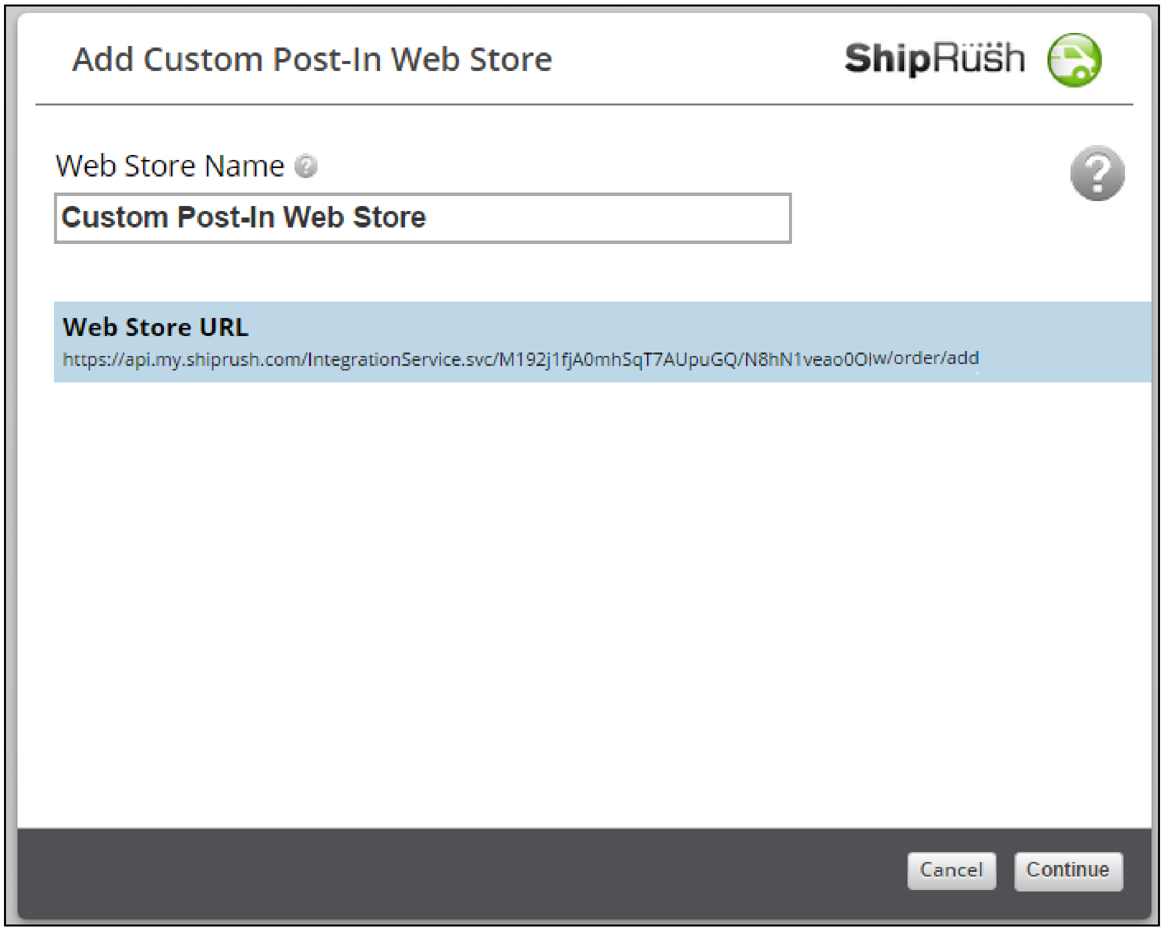
1. Create a post in store on the My.ShipRush site
2. Go to "http://requestb.in/" and create new capture url
3. Use Fiddler to send in an order with a PostbackUrl set to the capture url created above
4. Ship the order (use a plain "USPS" account in ShipRush to get quick setup for shipping without a credit card of carrier account)
5. At this point instant HTTP POST notification will go through
6. Go to to "http://requestb.in/1aa5uuj1?inspect" (token in your URL will be different) and look at payload

2.6.3. My.ShipRush API Users: Steps to see PostbackUrl notification working with interactive and online tools only (no modifications to your server code required)

- Start ShipRush.Tests.GUI.exe (from My.ShipRush SDK package)
- Create new user (or grab token for existing user)
- Go to "webstores" tab. Create new webstore of type "postin". After browser pops - complete registration (click "next" few times)
- Go to "http://requestb.in/" and create new capture url
- Go back to ShipRush.Tests.GUI to "Shipments and orders" tab, paste capture url to "Postback Url" field
- Click "Add pending shipments" <- this simulates your webstore posting shipments to My.ShipRush
- Go to My.ShipRush Web Shipping (http://my.shiprush.com)
- Go to "Ready To Ship" \ "Custom PostIn Webstore" folder
- Ship one of the pending shipments <- at this point instant HTTP POST notification will go through
- Go to to "http://requestb.in/1aa5uuj1?inspect" (token in your URL will be different) and look at payload
- Wait few seconds <- at this point second background HTTP POST notification will fire

2.7. Setup Workflow

The user starts in Order Manager (or My.ShipRush), creates a My.ShipRush ID, and creates the web store. During the web store creation, the user is given the URL which gets entered on the host system configuration.



Add Custom Post-In Web Store

Web Store Name

Custom Post-In Web Store

Web Store URL

https://api.my.shiprush.com/IntegrationService.svc/M192j1fjA0mhSqT7AUpuGQ/N8hN1veao0OIw/order/add

For developers with many users, the Web Store Type listed in My.ShipRush can be customized, to name the host system clearly, and offer specific help content to assist with setup. For example, if you publish a system called SuperDuperCart, you can request that this be listed as the store type in the list of store types.

2.8. Sample HTTP POST (HTTP POST headers and body)

Note that the Content-Type header is required.

2.8.1. Headers

```
HTTP/1.0
Content-Type: application/xml
Host: api.my.shiprush.com
Content-Length: 1817
```

2.8.2. Order Sample: With order details

```
<?xml version = "1.0"?>
<Request>
 <ShipTransaction>
  <Order>
   <!-- See MyShipRushService.xsd for enums and explanations -->
   <OrderNumber>ZF-0034</OrderNumber>
   <Total>199.95</Total>
   <PaymentStatus>2</PaymentStatus>  <!-- 2=Paid It is IMPORTANT to set this to Paid -->
   <ShipmentOrderItem>
    <Name>iPhone Nano 8GB</Name>
    <Price>199.98</Price>
    <Quantity>1</Quantity>
   </ShipmentOrderItem>
   <BillingAddress>
    <FirstName>John Stewart</FirstName>
    <LastName></LastName>
    <Address1>101 Lakeside AVE APT 4703</Address1>
    <City>KILLEEN</City>
    <State>TX</State>
    <PostalCode>76541</PostalCode>
    <Country>US</Country>
    <Phone>5054144980</Phone>
   </BillingAddress>
  </Order>
  <Shipment>
   <UPSServiceType>U02</UPSServiceType>   <!-- Optional. This is USPS Priority Mail. Sets
shipping service.  -->
   <ShipmentChgType>PRE</ShipmentChgType> <!-- Optional. Sets shipment payment/charge
(prepaid, third party)  -->
   <PartnerId>Amazon</PartnerId>
   <PostbackUrl>http://requestb.in/1aa5uuj1</PostbackUrl>
   <Package>
    <PackageActualWeight>4</PackageActualWeight>
    <PackagingType>02</PackagingType>
    <PackageReference1></PackageReference1>
    <InsuranceAmount>55</InsuranceAmount>  <!-- Optional. to set insurance.  -->
   </Package>
   <DeliveryAddress>
    <Address>
     <FirstName>John Stewart</FirstName>
     <LastName></LastName>
     <Address1>101 Lakeside AVE APT 4703</Address1>
     <City>KILLEEN</City>
     <State>TX</State>
     <PostalCode>76541</PostalCode>
     <Country>US</Country>
     <Phone>5054144980</Phone>
    </Address>
   </DeliveryAddress>
  </Shipment>
 </ShipTransaction>
</Request>
```

2.8.3. Order Sample: The Bare Minimum Order

Just an address. No order information other than an order number.

```
Suggested minimum elements for <Order>
```

```
OrderNumber     // to map to your system

PaymentStatus   //  send 2  (means paid, ready to ship)

ShipmentType    // Set to Pending (means not shipped)

OrderDate       // Set to appro date, remember to use appro ISO encoding as mentioned above in this
document
```

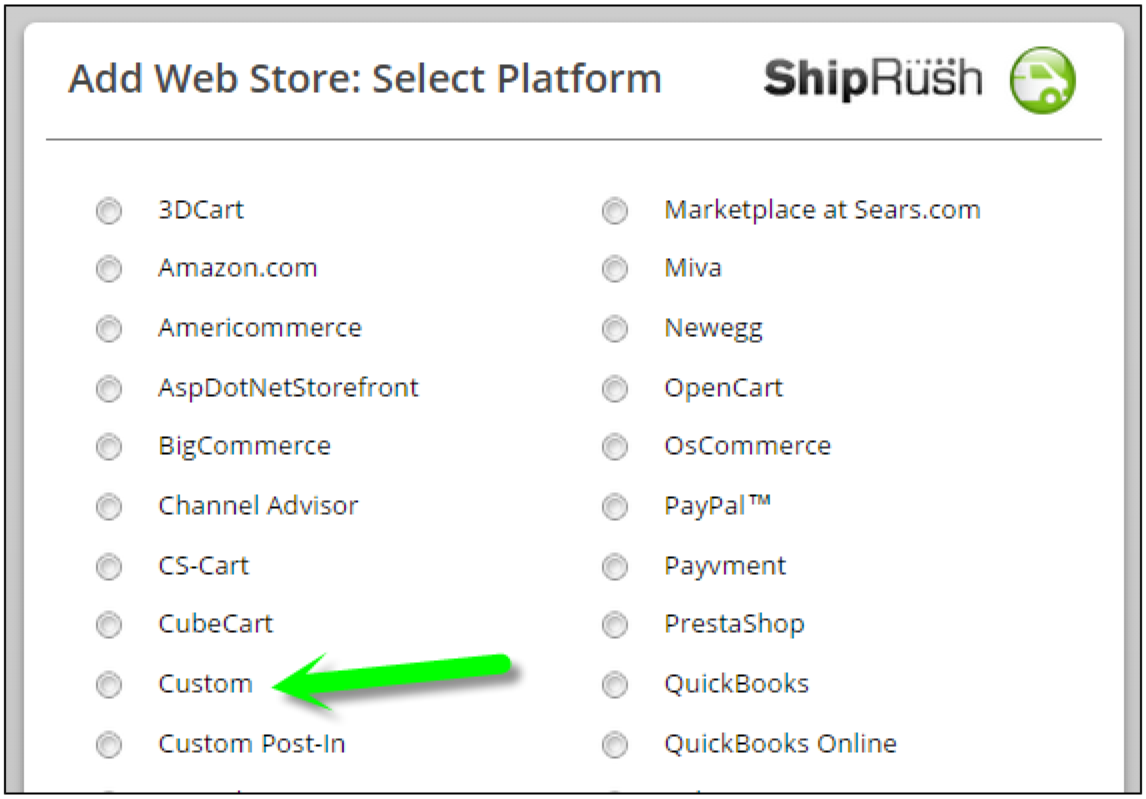
```
<?xml version = "1.0"?>
<Request>
 <ShipTransaction>
  <Order>
   <OrderNumber>ZF-0040</OrderNumber>
  </Order>
  <Shipment>
   <Package>
    <PackageReference1>Widgets</PackageReference1>
   </Package>
   <DeliveryAddress>
    <Address>
     <FirstName>John Stewart</FirstName>
     <Address1>101 Lakeside AVE APT 4703</Address1>
     <City>KILLEEN</City>
     <State>TX</State>
     <PostalCode>76541</PostalCode>
     <Country>US</Country>
     <Phone>5054144980</Phone>
    </Address>
   </DeliveryAddress>
  </Shipment>
 </ShipTransaction>
</Request>
```

2.8.3.1. Error Response

```
<?xml version="1.0" encoding="utf-8"?>
<Error xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Message>Invalid security token 'a2a87a5d-576b-4d39-bcc0-###########'. Token not found.
[0013].</Message>
  <Details/>
  <CallContextId>caf7d793-7ef0-46b2-a278-###########</CallContextId>
</Error>
```

3. Polling API

Webstore type: **custom**



With Polling API My.ShipRush will call your website periodically to see if there are new sales available. Your website needs to have an entry point for My.ShipRush calls and return ShipRush-friendly XML. My.ShipRush will pass query parameters in the URL parameters and will expect response in the HTTP body.

Z-Firm build several PHP implementations based on this API. ZenCart, Magento, OsCommerce integrations use this API to get orders from these shopping carts. You can use our code as a reference point for implementing your integration. Download latest package from http://my.shiprush.com/assets/ShipRush_ShoppingCart_Integration_Kit__See_README_file.zip

You need to implement following commands. My.ShipRushs expects HTTP 200 for all successful requests and HTTP 500 for all failed requests. Failed requests will be retried in 2-4 hours.

Url Parameters	Description
cmd	Command
shipping_access_token	Security token that identifies caller as My.ShipRush.
datefrom	Get orders From date in UTC
dateto	Get orders To date in UTC

3.1.1. getordersbydate

Return all sales that fit date range http://mywebstore.com/MyShipRushEntry.php?cmd=getordersbydate&shipping_access_token=ABCD1234SUPER&datefrom=2000-01-01T01:01:01Z&dateTo=2020-01-01T01:01:01Z

3.1.2. ping

Check setup and return basic server info as XMLhttp://mywebstore.com/MyShipRushEntry.php?cmd=ping&shipping_access_token=ABCD1234SUPER

```
<Response>
<Message>Success. Access token database access verified.</Message>
<Version>1.0.0.36236</Version>
</Response>
```

3.1.2.1.1. getserverinfo

Return expanded server info as text blobhttp://mywebstore.com/MyShipRushEntry.php?cmd=getserverinfo&shipping_access_token=ABCD1234SUPER

3.1.2.1.2. updateshippinginfo

Update order in your system with tracking number and name of the shipping service. http://mywebstore.com/MyShipRushEntry.php?cmd=updateshippinginfo&shipping_access_token=ABCD1234SUPER&OrderNumber=ZF999&TrackingNumber=1Z00000000000&Carrier=UPS&Service=UPS+Ground

4. FAQ

4.1. Can you send me a javascript example for post out and post back?

It is true that you could implement the code to do post out/post in in node.js, and this would be javascript code. However, you cannot interact with ShipRush Web via js in a browser. Why? Because it is a cross-domain-scripting approach, which browsers forbid due to security concerns.

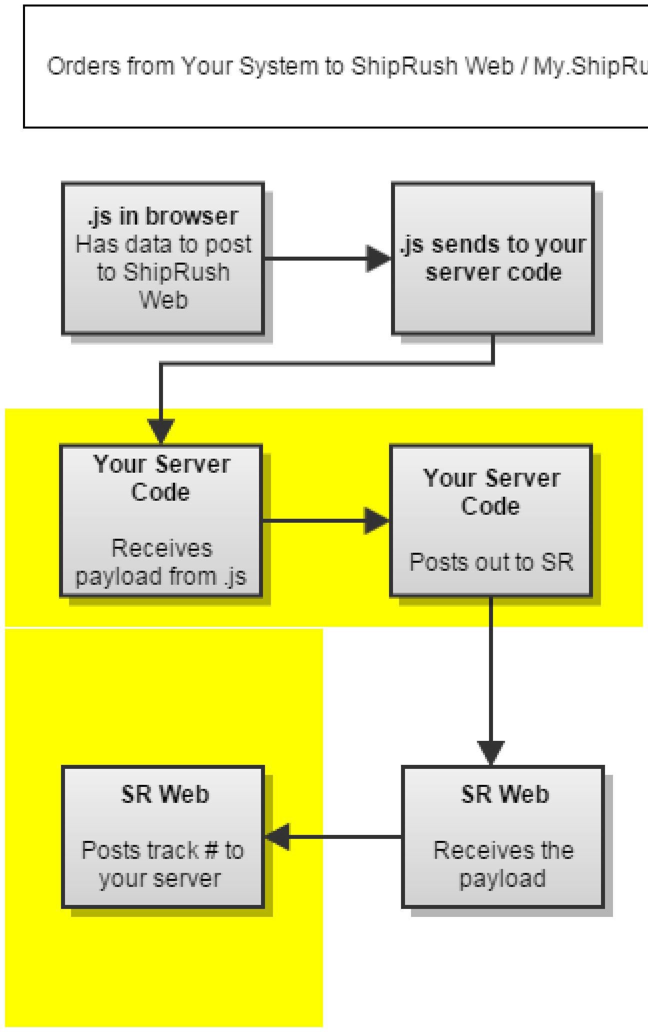
The solution: Go from .js in browser to your site, and from your site (server side code) out to SR.

Then, (obviously) SR is going to post back to a site (not to a browser).

More about cross site scripting (aka XSS):

- https://danielmiessler.com/study/xss/
- http://security.stackexchange.com/q/1368/43858

The needed flow is:


Orders from Your System to ShipRush Web / My.ShipRush
.js in browser Has data to post to ShipRush Web
.js sends to your server code
Your Server Code Receives payload from .js
Your Server Code Posts out to SR
SR Web Receives the payload
SR Web Posts track # to your server

5. Document Revision History

Date	My.ShipRush Build	ZF Case (if applicable)	Comment
April 3, 2017	101392	ZF 57520, 57478	Updated XSD in the kit
September 26, 2016	95469	Case 54007	New section on how to add tags to incoming shipments.
August 29			Added new "Suggested minimum elements for <Order>"
July 29, 2016			Improved example in section "Order Sample: With order details"
July 13, 2016		Case 52720	Added section on how to update order statuses
July 12, 2016			New charts and explanations added to docs

July 11, 2016			Added new Terms section
Sept 3, 2015			`Expanded detail RE PostBackURL`
Sept 2, 2015			New FAQ section
August 2015			Original release