Bolt Embedded Accounts SFRA

Version 22.1.1

Table of Contents

Intended Audience

This document is for technical personnel assigned to integrate Bolt embedded accounts into an existing Salesforce Commerce Cloud solution.

1. Overview

1.1 High-Level Overview

Bolt embedded account is a headless version of Bolt's checkout which allows merchants and partners to build their own one-click checkout experience while taking advantage of the Bolt network via server-to-server APIs and an even more limited set of Bolt UI components. With embedded accounts, shoppers with a Bolt account can grant access to their Bolt account data, such as addresses and payment methods, to a merchant via OAuth. Then, the merchant is able to access the data via REST APIs.

1.2 Use Cases

There are two supported checkout processes: 1. Shopper does not have a Bolt account and checks out as a guest, 2. Shopper has a Bolt account and we use their data stored in Bolt for checkout.

1.2.1 Guest Shoppers Checkout (shoppers without a Bolt account):

This flow utilizes Bolt's embedded payments component and tokenize() function to enable checkout on your storefront for shoppers with no existing Bolt account. During this flow, shoppers can opt-in to create a Bolt account for future purchases.
Steps:

1. Shopper enters their email address, a request is sent to Bolt but finds no account exists for the given email.
2. Shopper continues checkout, enters shipping address, and selects a delivery method with merchant's UI.
3. Shopper enters their payment information. They will also be shown a checkbox to get their consent for creating a Bolt account. Once the shopper submits the payment, we

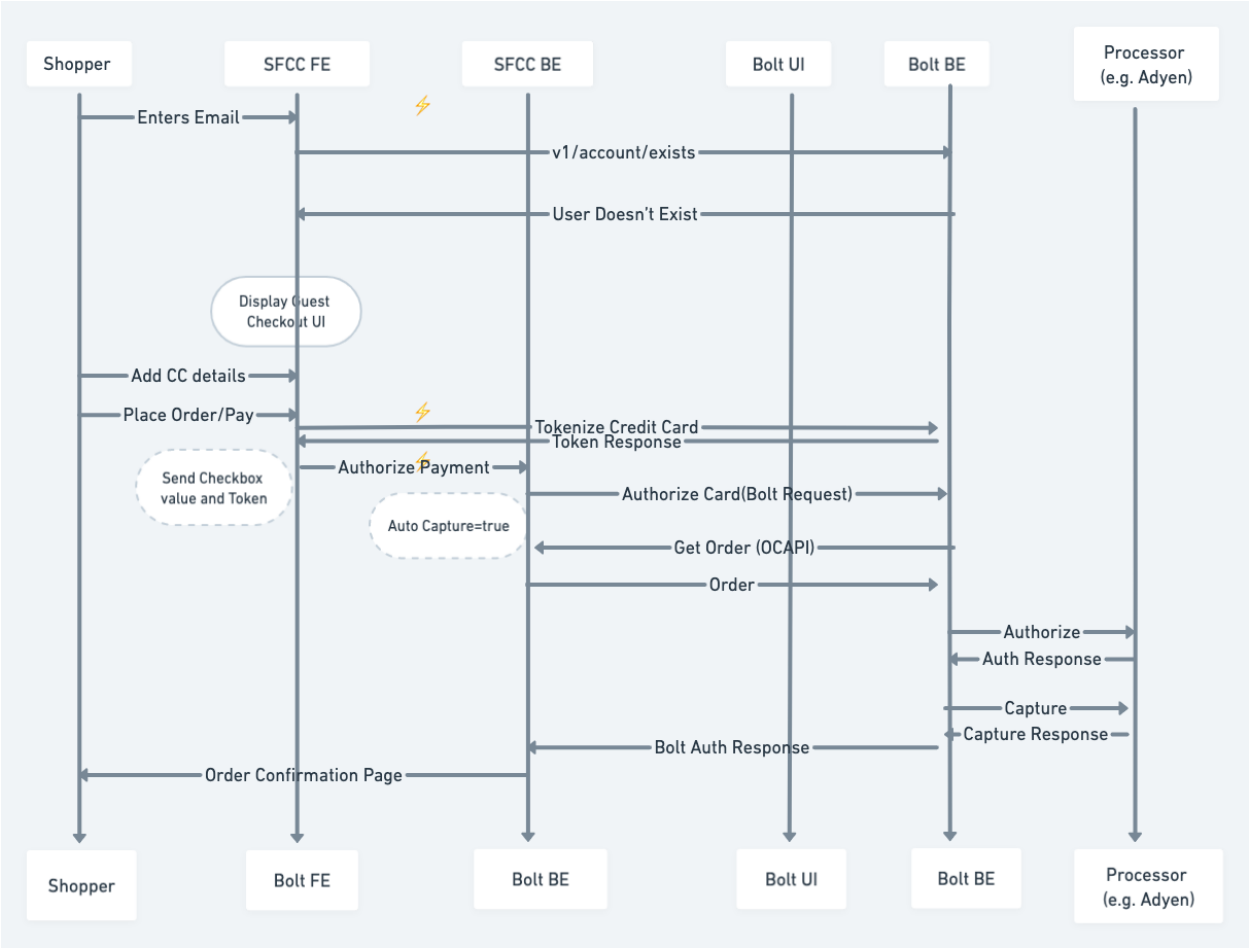
tokenize the card.

☑ Save my information and create a Bolt account

- Easy order tracking and more
- Shop hundreds of brands with one login
- Check out faster with saved information
- Bank level security

4. Shopper clicks the **“place order”** button, and a REST call is sent to Bolt to authorize the card. Show order confirmation page if the credit card is authorized.

The following sequence diagram shows the guest checkout process. The endpoints marked by ⚡ are the ones implemented in our Embedded Cartridge.

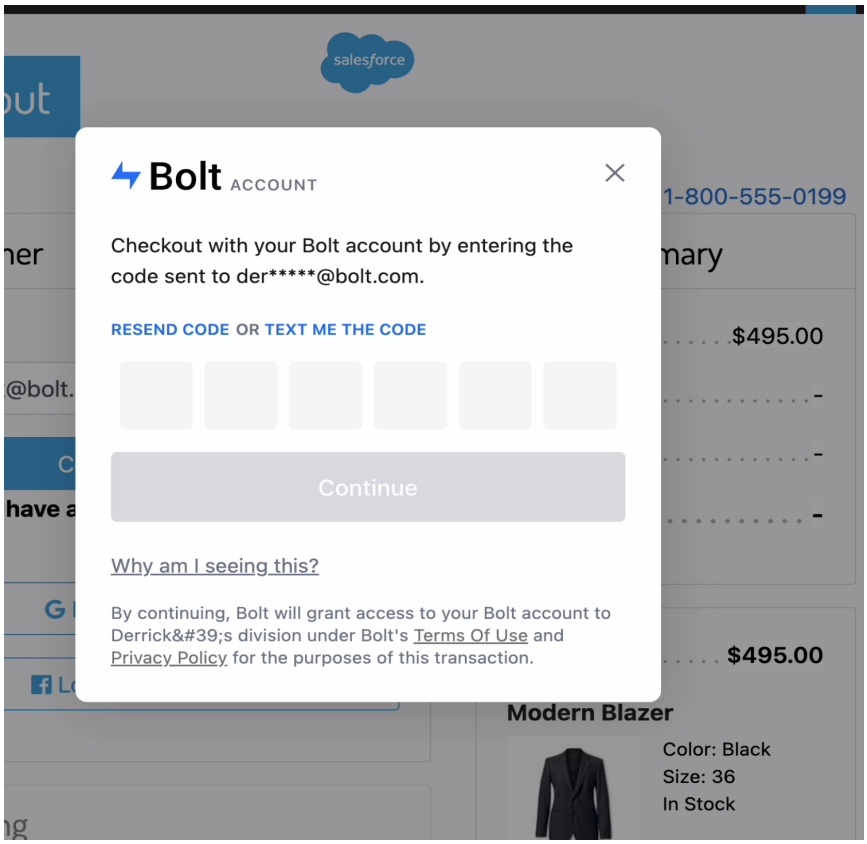


1.2.2 Bolt Returning Shoppers Checkout (shoppers with a Bolt account):

This flow utilizes Bolt's *checkAccountAndFetchDetail()* function and authorization component to determine whether a shopper has a Bolt account, and gives them the option to log in via a one-time password and checkout.

Steps:

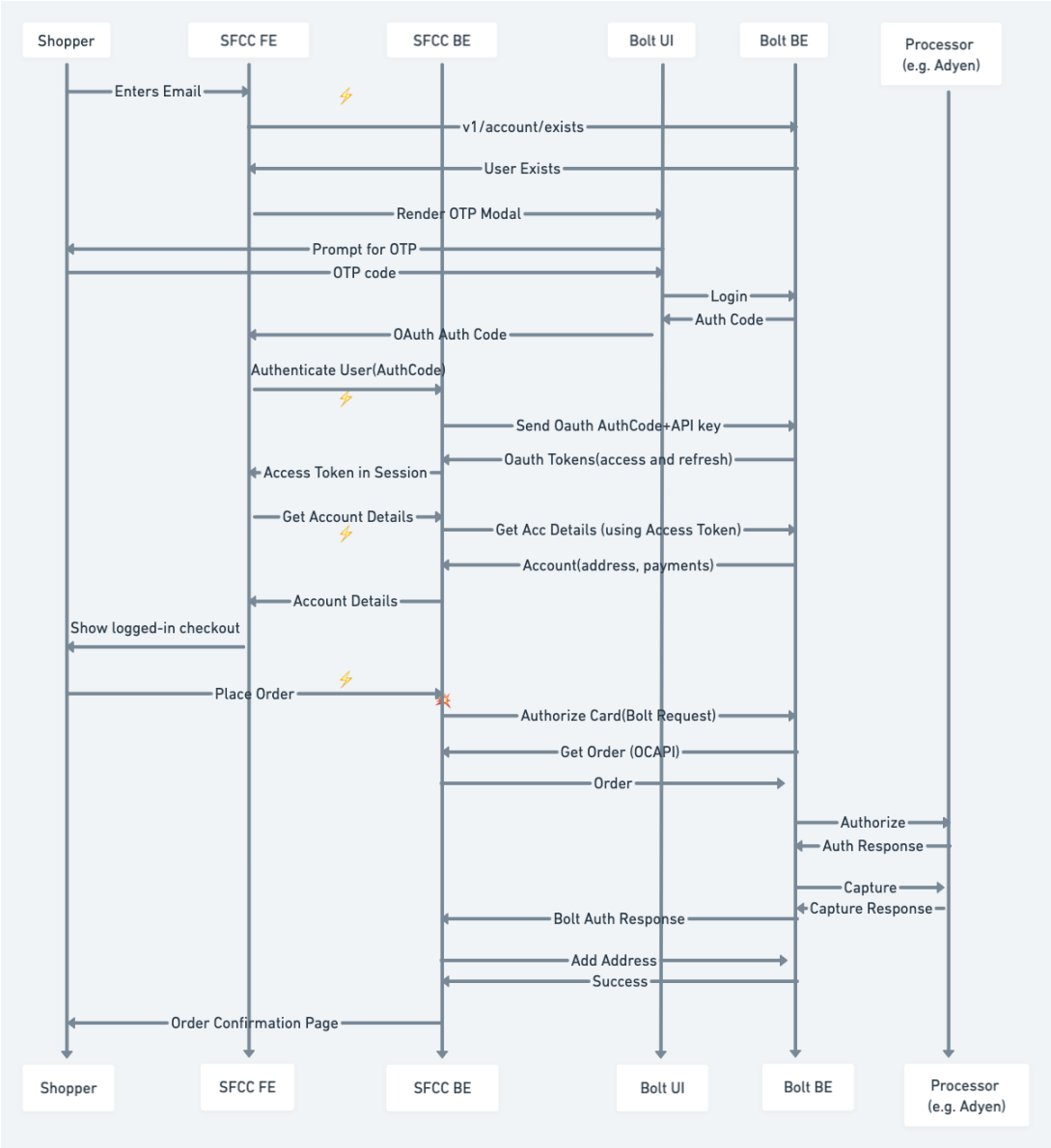
1. The shopper enters their email address, check account exists request sent from the merchant frontend to Bolt. If the given email has an account, the OTP modal is shown to the shopper.



2. Shopper enters the OTP code and gets authenticated.
3. Shipping address is automatically filled with default shipping addresses from their account, default delivery method is selected.
   Note: if their account is missing information, they will be redirected to the corresponding step in the checkout process to fill in the missing information.
4. Shopper selects one of the credit cards used previously to pay, and the corresponding billing address will be populated. They have the option to add a new credit card also.
5. Shopper reviews the order and can change the shipping address and delivery method.

6. Shopper clicks the‘ place order’ button, and a REST call is sent to Bolt to authorize the card. Show order confirmation page if the credit card is authorized.
7. If the shopper added a new shipping address, edited an existing shipping address, or added a new payment, we automatically save that to the shopper’s Bolt account once authorization succeeds.

The following sequence diagram shows the Bolt returning shopper checkout process. The endpoints marked by ⚡ are the ones implemented in our Embedded Cartridge. If shopper has added (or modified) a new credit card, there will be a separate call to Bolt backend happens, which is marked by 💥in the diagram below.



1.3 Privacy

The Bolt embedded accounts integration requires access to the following data:

- cart details
- billing address
- customer email

1.4 Compatibility

The Bolt embedded account cartridge is compatible with Storefront Reference Architecture version 6.1.0, and 22.7 compatibility mode. The testing locale is en_US(by default).

1.5 Supported Payment types

The Bolt embedded account cartridge supports following payment types:

1. Visa
2. Master
3. American Express
4. Discover

The Bolt embedded account cartridge does not support:

1. ApplePay
2. SFCC Gift Certificate

1.6 Testing

Use Cases	Steps	Expecation
Guest Checkout	1. Add product to the cart. 2. Open the checkout page. 3. Fill up the email field and click the “Continue as guest” button. 4. Fill up the shipping form and click the “Next: Payment” button. 5. Fill up the billing form, check the checkbox	The order is placed successfully. A bolt user account is created with the shipping and payment information in the order.

	“Save my information with Bolt for faster checkout” and click the “Next: Place Order” button. 6. Click the “Place Order” button.	
Bolt User Checkout with a existing card	1. Add product to the cart. 2. Open the checkout page. 3. Fill up the email field with the email that is linked to a bolt account(make sure the account has shipping and payment information). 4. A modal will pop up and ask for the OTP(one time passcode) which is sent to an email/phone. 5. Input the OTP and automatically redirect to the order review page 6. Click the “Place Order” button.	The order is placed successfully.
Bolt User Checkout with a new card	1. Add product to the cart. 2. Open the checkout page. 3. Fill up the email field with the email that is linked to a bolt account(make sure the account has shipping and payment information). 4. A modal will pop up and ask for the OTP(one time passcode) which is sent to an email/phone. 5. Input the OTP and automatically redirect to the order review page . 6. Click the “Edit” button in the payment section. 7. Click “Add a new card” and fill in the credit card information. Then click the “Next: Place Order” button. 8. Click the “Place Order” button.	The order is placed successfully. The new credit card information is added to the bolt user account.
Bolt User Checkout with a new address	1. Add product to the cart. 2. Open the checkout page. 3. Fill up the email field with the email that is linked to a bolt account(make sure the account has shipping and payment information). 4. A modal will pop up and ask for the OTP(one time passcode) which is sent to an email/phone. 5. Input the OTP and automatically redirect to the order review page . 6. Click the “Edit” button in the shipping section. 7. Click “Add New” and fill in the new shipping address. Then click the “Next: Payment” button. 8. Click the “Next: Place Order” button. 8. Click the “Place Order” button.	The order is placed successfully. The new shipping address is added to the bolt user account.

2. Integration Guide

2.1 Upload the cartridge to SFCC DigitalServer

Prerequisite: This document expects that the merchant has storefront base cartridges up and knows the running code version.

The Bolt cartridge zip can be found here:
https://github.com/BoltApp/bolt-demandware-embedded/releases

1. In Business Manager, Administration > Sites > Manage Sites > Settings tab, add `int_bolt_embedded_sfra` to the cartridge path.
2. Upload the code via VS code or command line.

2.2 Business Manager Configurations

Metadata, site preferences, service, payment methods, and OCAPI need to be configured in Business Manager.

2.2.1 Importing Metadata

Steps for importing metadata.

1. Go to **Business Manager** > **Administration** > **Site Development** >**Site Import & Export**.
2. Upload and import the metadata (zip file) under *metadata*. Alternatively, use *npm run zipMetadata* and *npm run uploadMetadata.*
3. Navigate to **Business Manager** > **Administration** > **Open Commerce API Settings**
    a. Copy the metadata under *metadata/ocapi/OCAPIshop_embedded.json* to the SHOP API section, replace <<client_id>> with your own client id, then save.
    b. Copy the metadata under *metadata/ocapi/OCAPIdata_embedded.json* to the DATA API section, replace <<client_id>> with your own client id, then save.

2.2.2 Custom Site Preference

Steps for configuring site preference

1. In the Business Manager, navigate to **Merchant Tools** > **Site Preferences** > **Custom Preferences**.
2. Set these values in preference group **Bolt Payment Setting - Embedded**
   API Key: Bolt provided merchant account identifier.
   Bolt Merchant Division ID: The unique ID associated with the merchant's Bolt Account division. Merchants can have different divisions to suit multiple use cases (storefronts, pay-by-link, phone order processing). Use the Bolt Merchant Dashboard to switch

between divisions and find the division ID under Merchant Division Public ID.
Publishable Key: Bolt provided, used to get your Bolt merchant division specific value.
Bolt Environment: String Enum (Sandbox, Staging, Production) Used to switch different Bolt environment endpoint hostname.
Enable Bolt Pay: Enable Bolt Pay or not.
Enable Auto Capture: Enable Auto Capture or not.

2.2.3 Payment Method

Bolt payment method and payment processor should be imported successfully in 2.2.1
In Business Manager, go to **Merchant Tools > Ordering > Payment Methods**, make sure `*BOLT_PAY*` is enabled and the payment processor is configured.

2.2.4 Service

Service used to communicate with Bolt backend should be imported successfully in 2.2.1
In Business Manager, go to **Administration > Operations > Services,** make sure service `bolt.http` is added.

2.3 Customization

The following sections explain how the Bolt embedded cartridge can be used with a customized SFRA website. We have included a sample cartridge (`int_bolt_embedded_sfra_custom`) for your reference on how this customization can be done.

2.3.1 Dependency on native front end component

Bolt relies on the ***$('#email-guest')*** SFCC native front end component to trigger the request for checking if that email has a bolt account. Please change that component ID in `eventListenerRegistration.js` if you apply any customization to this input.

Bolt also uses the keyup callback to trigger the call to render the OTP modal,. When the user types in a valid email address and stops entering anything for 1 second, we will make a call to check if that email has a bolt account and pre-fill all shopper info if there is an associated bolt account. If you'd like a different user experience, feel free to modify eventListenerRegistration.js to change the UX.

2.3.2 Template change

Checkout

In *cartridge/templates/default/checkout/checkout.isml*
Adding following code snippets:

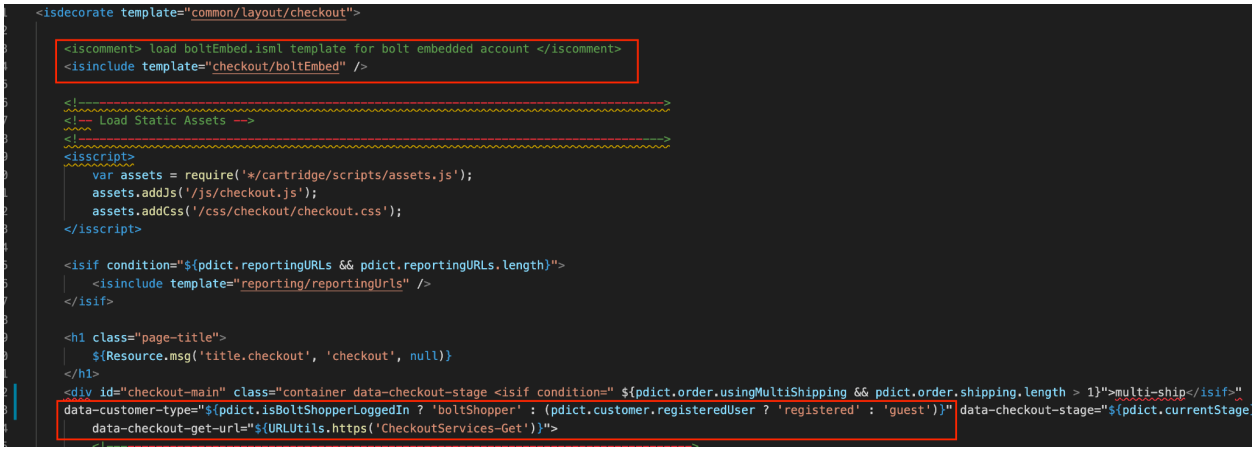
Before the first *<isscript>* tag, include the following template

```
<isinclude template="checkout/boltEmbed" />
```

Add Bolt shopper type check

```
<div id="checkout-main" class="container data-checkout-stage <isif
condition=" ${pdict.order.usingMultiShipping && pdict.order.shipping.length
> 1}">multi-ship</isif>" data-customer-type="${pdict.isBoltShopperLoggedIn
? 'boltShopper' : (pdict.customer.registeredUser ? 'registered' :
'guest')}" data-checkout-stage="${pdict.currentStage}"
       data-checkout-get-url="${URLUtils.https('CheckoutServices-Get')}">
```

For example:



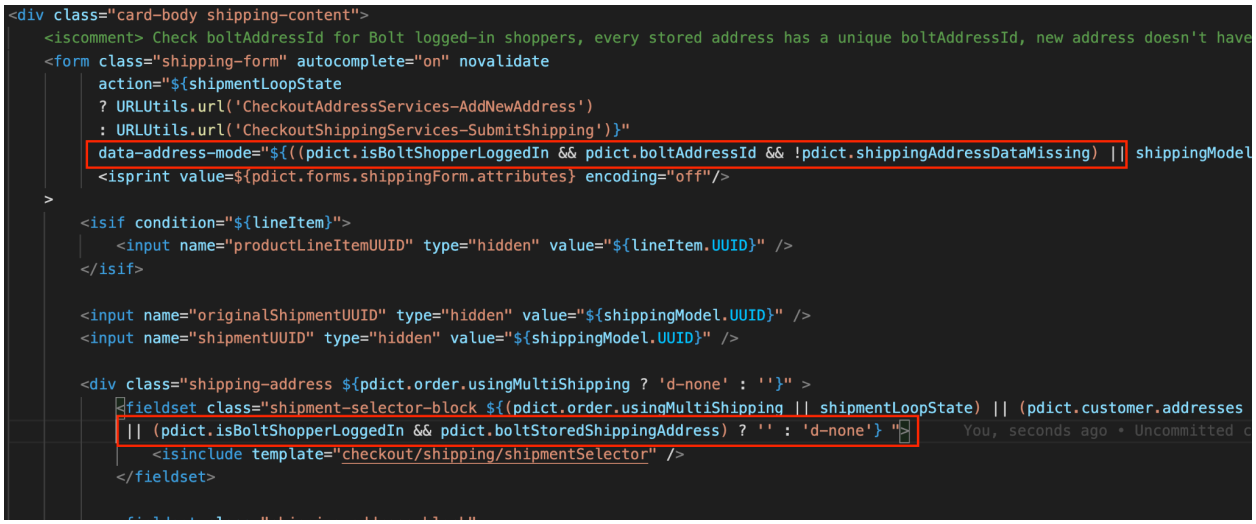
Shipping

In *cartridge/templates/default/checkout/shipping/shipmentCard.isml*
adding following code snippets:

```
data-address-mode="${((pdict.isBoltShopperLoggedIn && pdict.boltAddressId
&& !pdict.shippingAddressDataMissing) || shippingModel.matchingAddressId) ?
'edit' : 'new'}"
```

```
|| (pdict.isBoltShopperLoggedIn && pdict.boltStoredShippingAddress)
```

All together:



In *cartridge/templates/default/checkout/shipping/shipmentSelectorOptions.isml*

Add following code snippet:

```
<isif condition="${pdict.isBoltShopperLoggedIn &&
pdict.boltStoredShippingAddress}">
   <option disabled>${Resource.msg('msg.account.addresses', 'checkout',
null)}</option>
   <isloop items="${pdict.boltStoredShippingAddress}" var="address">
       <option value="${'ab_'+address.id}"
${address.id==pdict.boltAddressId ? 'selected="selected"' : '' }
       data-first-name="${address.first_name||''}"
data-last-name="${address.last_name||''}"
       data-address1="${address.street_address1||''}"
data-address2="${address.street_address2||''}"
       data-city="${address.locality||''}"
data-state-code="${address.region_code||''}"
       data-country-code="${address.country_code||''}"
data-postal-code="${address.postal_code||''}"
       data-phone="${address.phone_number||''}" data-is-gift="${''}"
data-gift-message="${''}"
       data-bolt-address-id="${address.id}">
       ${address.first_name || ''} ${address.last_name || ''}
${address.street_address1 || ''} ${address.street_address2 || ''}
${address.locality || ''}${!address.locality || ','}
${address.region_code||''} ${address.postal_code||''}
```

```
        </option>
    </isloop>
```

in this part of code:

```
<isif condition="${pdict.isBoltShopperLoggedIn && pdict.boltStoredShippingAddress}">
    <option disabled>${Resource.msg('msg.account.addresses', 'checkout', null)}</option>
    <isloop items="${pdict.boltStoredShippingAddress}" var="address">
        <option value="${'ab_'+address.id}" ${address.id==pdict.boltAddressId ? 'selected="selected"' : '' }
        data-first-name="${address.first_name||''}" data-last-name="${address.last_name||''}"
        data-address1="${address.street_address1||''}" data-address2="${address.street_address2||''}"
        data-city="${address.locality||''}" data-state-code="${address.region_code||''}"
        data-country-code="${address.country_code||''}" data-postal-code="${address.postal_code||''}"
        data-phone="${address.phone_number||''}" data-is-gift="${''}" data-gift-message="${''}"
        data-bolt-address-id="${address.id}">
        ${address.first_name || ''} ${address.last_name || ''} ${address.street_address1 || ''} ${address.street_address2 || ''} ${address.locality || ''}${!address.locality || ','}
        </option>
    </isloop>
<iselseif condition="${pdict.customer.addresses && pdict.customer.addresses.length > 0}">
    <option disabled>${Resource.msg('msg.account.addresses', 'checkout', null)}</option>
```

In *cartridge/templates/default/checkout/shipping/shippingAddress.isml*
add following code snippet:

```
<input type="hidden" name="${addressFields.boltAddressId.htmlName}"
value="${pdict.boltAddressId}">
```

to the code here:

```
1   <isset name="addressFields" value="${pdict.forms.shippingForm.shippingAddress.addressFields}" scope="page"/>
2   <isif condition="${shippingModel.shippingAddress}">
3       <isset name="shippingAddress" value="${shippingModel.shippingAddress}" scope="page" />
4   <iselse/>
5       <isset name="shippingAddress" value="${{}}" scope="page" />
6   </isif>
7
8   <input type="hidden" name="${addressFields.boltAddressId.htmlName}" value="${pdict.boltAddressId}">
9
10  <div class="row">
11      <div class="col-sm-6">      You, a week ago • Display stored bolt shipping address (#26)
12          <div class="form-group
13              ${addressFields.firstName.mandatory === true ? 'required' : ''}
14              ${addressFields.firstName.htmlName}">
15              <label class="form-control-label" for="shippingFirstName${lineItem ? lineItem.UUID : 'default'}"  >
16                  ${Resource.msg('field.shipping.address.first.name','address',null)}
```

Payment
In *cartridge/templates/default/checkout/billing/paymentOptions/paymentOptionsTabs.isml*

add following code snippet:

```
<isif condition="${paymentOption.ID === 'BOLT_PAY'}">
    <isinclude template="checkout/billing/paymentOptions/bolt/paymentTab"
/>
</isif>
```

to the code here:

```
<isloop items="${pdict.order.billing.payment.applicablePaymentMethods}" var="paymentOption">
    <isif condition="${paymentOption.ID === 'BOLT_PAY'}">
        <isinclude template="checkout/billing/paymentOptions/bolt/paymentTab" />
    </isif>
</isloop>
```

In *cartridge/templates/default/checkout/billing/paymentOptions/paymentOptionsContent.isml* add following code snippet:

```
<isif condition="${paymentOption.ID === 'BOLT_PAY'}">
    <isinclude
template="checkout/billing/paymentOptions/bolt/paymentContent" />
</isif>
```

to the code here:

```
<isloop items="${pdict.order.billing.payment.applicablePaymentMethods}" var="paymentOption">
    <isif condition="${paymentOption.ID === 'BOLT_PAY'}">
        <isinclude template="checkout/billing/paymentOptions/bolt/paymentContent" />
    </isif>
</isloop>
```

In *cartridge/templates/default/checkout/billing/paymentOptions/paymentOptionsSummary.isml* add following code snippet:

```
<isif condition="${payment.paymentMethod === 'BOLT_PAY'}" />
      <isinclude
template="checkout/billing/paymentOptions/bolt/paymentSummary" />
</isif>
```

add the code here:

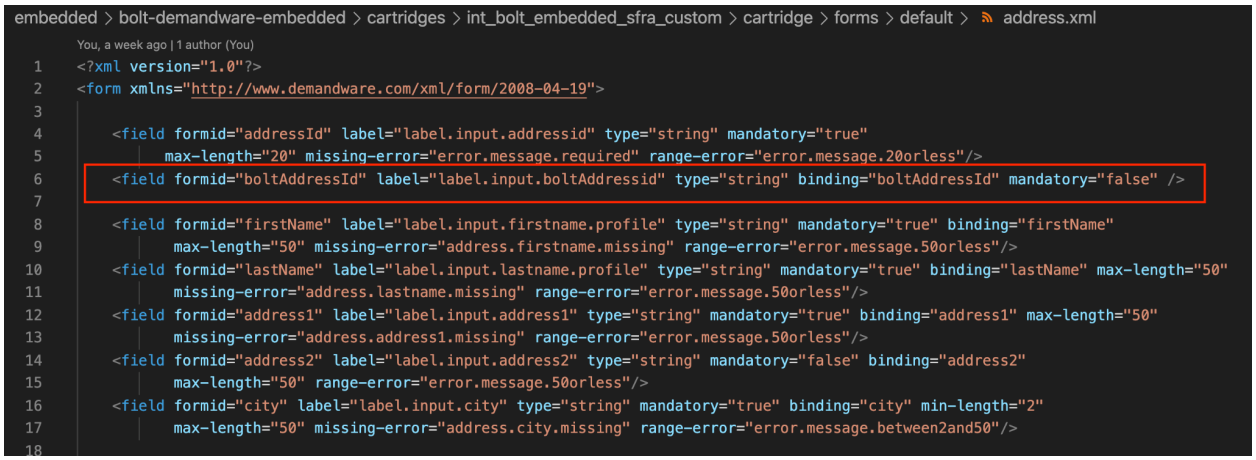
```
<div class="payment-details">
    <isloop items="${pdict.order.billing.payment.selectedPaymentInstruments}" var="payment">
        <isif condition="${payment.paymentMethod === 'BOLT_PAY'}" />
        <isinclude template="checkout/billing/paymentOptions/bolt/paymentSummary" />
        </isif>
    </isloop>
</div>
```

2.3.3 Form Change

In *cartridge/forms/default/address.xml*
add following code snippet:

```
<field formid="boltAddressId" label="label.input.boltAddressid"
type="string" binding="boltAddressId" mandatory="false" />
```

to the code here:



In *cartridge/forms/default/billing.xml*
add following code snippet:

```
<include formid="boltCreditCard" name="boltCreditCard"/>
```

to the code here:

3. Operation, Maintenance

3.1 Data Storage

This integration requires the following system object extensions.

- **OrderPaymentInstrument custom attribute:**
  - **boltCardBin(String)**: Store the credit card bin
  - **boltPaymentMethodId(String)**: Bolt Payment Method ID
  - **boltCreateAccount(Boolean)**: Indicate if shopper choose to create a Bolt account
  - **boltTokenType(String)**: Bolt token type
- **SitePreferences custom attribute:**
  - **boltAPIKey(String)**: Bolt provided merchant account identifier.
  - **boltMultiPublishableKey(String)**: Bolt provided, used to get your Bolt merchant division specific value.
  - **boltEnvironment(String)**: String Enum (Sandbox, Staging, Production) Used to switch different Bolt environment endpoint hostname .
  - **BoltEnable(Boolean)**: Indicate if bolt is enabled
  - **boltMerchantDivisionID(String)**: Bolt merchant division ID
  - **boltEnableAutoCapture(Boolean)**: Indicate if the fund is auto captured

3.2 Logs

This integration introduces a few new custom logs:

- **Bolt Custom Log:**
  Such as the custom-Bolt-blade5-0.mon.demandware.net-0-appserver-20191120.log. This Log file contains all logs from `*int_bolt_embedded_sfra*` cartridge.
- **Service communication Logs:**
  These logs contain every request and response to the Bolt API endpoint. To enable these logs, navigate to **Administration** > **Operations** > **Services** > **bolt**.**http** and check the Communication Log Enabled preference. **getRequestLogMessage** and **getResponseLogMessage** function are implemented to filter sensitive data on Production Environment.

3.3 Availability

If the Bolt service is unavailable the user will not be able to checkout using this payment method. The service availability can be tracked in SFCC using the Service Status in the

Commerce Cloud Business Manager. (Navigate to **Administration** > **Operations** > **Service Status** > **bolt.http**)

3.4 Support

To get help and support from Bolt:

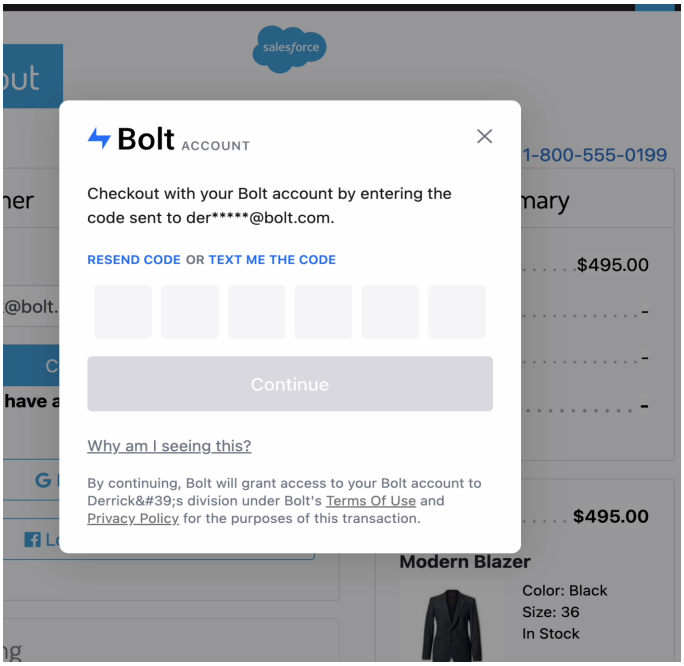
- **Support Knowledge Center:** https://support.bolt.com/hc/en-us/requests/new
- **Status of Bolt Services**: https://status.bolt.com
- **Contact Support**: merchantsupport@bolt.com

3.5 Recovery process

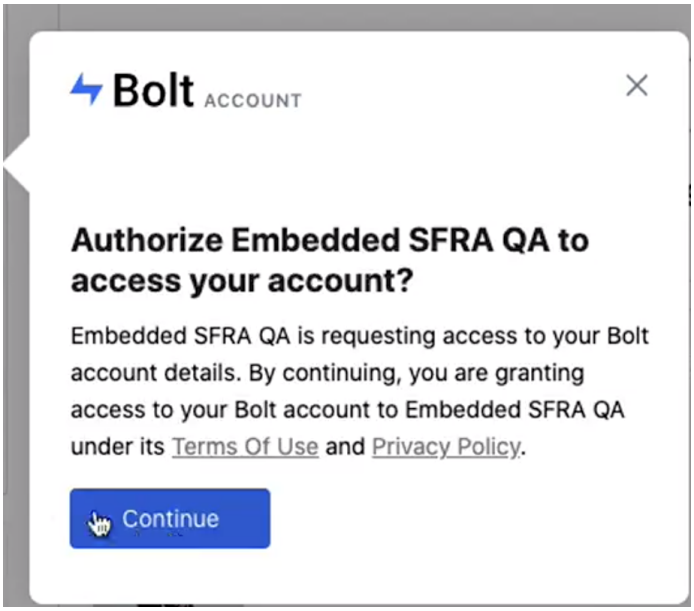
If Bolt service down is identified, refer to Support section(3.4) to get support from Bolt. If the downtime is not expected, go to Merchant Tools-> Ordering -> Payment Methods and disable payment 'BOLT_PAY'. Stay tight with bolt support, once the service is recovered, re-enable it.

4. Limitations

- For SFCC logged-in shoppers if the email address is automatically filled and takes the shopper directly to the shipping step, you may need to do some frontend changes to call ***checkAccountAndFetchDetail*** in `*int_bolt_embedded_sfra/cartridge/client/default/js/account.js*` to check if Bolt account exists.
- Currently do not support combined payments (e.g. gift card and credit card).
- SFCC Session Timeout Behavior
    - SFCC sessions expire after 30 minutes of inactivity where privacy related data such as shopper contact information, shipping address or payment details are cleared. If the shopper tries to checkout after the session expires, they will encounter an error which will prompt them to start the checkout process again. If the shopper decides to restart the checkout process, a new session will be created and the request to check for the Bolt account must be made again.
    - The below behavior is controlled by Bolt:
        - If the shopper has a bolt account and an active bolt session does not exist, then we show the modal where the shopper has to enter their OTP to authorize and continue.



- If the shopper has a bolt account and an active bolt session exists, then we show the modal where the shopper has to simply authorize to continue.



5. Additional Resources

- What is Bolt?
- Bolt Embedded Accounts

6. Release History

Version	Date	Changes
22.1.0	08/01/2022	Initial release
22.1.1	08/31/2022	Remove scripts from templates, filter sensitive data from logs.