> **Document type**: How-to guide
> **Audience**: Merchant and agency developers
> **Purpose**: This document provides instructions for generating and updating Subresource Integrity (SRI) hash values in a Salesforce Commerce Cloud cartridge to ensure secure script loading during checkout processes.

Update Subresource Integrity Hash Values for Scripts in the Cartridge

To help prevent script tampering and ensure that checkout processes work correctly, when you modify any file in the cartridge, generate and update the Subresource Integrity (SRI) hash values for the scripts in the cartridge.

Automatically generate new SRI hash values when building the cartridge

To automatically generate new SRI hash values when you build the cartridge, take the following steps.

1. Install the following packages. In version 0.13.0 and later, the cartridge includes the packages by default.

   ```
   npm install webpack-assets-manifest@4.0.0 --save
   npm install webpack-subresource-integrity@1.5.2 --save
   ```

2. In your `webpack.config.js` configuration file, require the `SubresourceIntegrityPlugin` and `WebpackAssetsManifest` plugins. In version 0.13.0 and later, the cartridge includes the plugins by default.

   ```
   var path = require('path');
   const SubresourceIntegrityPlugin = require('webpack-subresource-integrity');
   const WebpackAssetsManifest = require('webpack-assets-manifest');
   ```

3. In the `plugins` array of the Javascript configuration object, add instances of the `SubresourceIntegrityPlugin` and `WebpackAssetsManifest` plugins as follows. In version 0.13.0 and later, the cartridge includes the instances by default.

```
plugins: [
     new SubresourceIntegrityPlugin({
         hashFuncNames: ['sha384']
     }),
     new WebpackAssetsManifest({
         integrity: true,
         integrityHashes: ['sha384'],
         writeToDisk: true,
         output: 'js-asset-manifest.json'
     })
 ]
```

Generate the `js-asset-manifest.json` file

The `webpack-assets-manifest` package generates `js-asset-manifest.json`, which includes SHA-384 SRI hash values for each affected script. To generate the `js-asset-manifest.json` file, do the following.

1. At a command prompt, open your cartridge directory and build the cartridge.
2. After you build the cartridge, find the `js-asset-manifest.json` file in `link_amazon_bwp/cartridges/int_buywithprime/cartridge/static/`.

Update SRI hash values for scripts in the cartridge

When the cartridge loads a checkout script, it automatically verifies the SRI of the script by checking whether the new SRI hash value matches the expected SRI hash value. The following sections show how to update the SRI hash values for scripts in the cartridge to ensure that checkout processes work correctly.

Update the SRI hash value for `bwpSummary.js`

To update the SRI hash value for the `bwpSummary.js` script, take the following steps.

1. Open the following file in a text editor:

   ```
   link_amazon_bwp/cartridges/int_buywithprime_sfra/cartridge/templates/default/checkout/checkout.isml
   ```

2. In the Load Static Assets section, find the line that references the `bwpSummary.js` script.

   ```
   assets.addJs('/js/bwpSummary.js', 'sha384-xyZ123dEf456GhI789jKl+012MnO345pQr678sTu901vWx234yZa567bCd890EfG');
   ```

3. From the `js-asset-manifest.json` file, get the new SRI hash value for the `bwpSummary.js` entry under the `integrity` key. The following example contains a placeholder SRI hash value.

```
"default/js/bwpSummary.js": {
  "src": "default/js/bwpSummary.js",
  "integrity": "sha384-def123abC456GhI789jKl+012MnO345pQr678sTu901vWx234yZa567bCd890EfG"
},
```

4. In the `checkout.isml` file, replace the existing SRI hash value with the new SRI hash value from the `js-asset-manifest.json` file and keep the `sha384-` prefix.
5. Save the `checkout.isml` file.

Update the SRI hash values for `clickstreamLoader.js` and `buywithprime.js`

To update the SRI hash values for the `clickstreamLoader.js` and `buywithprime.js` scripts, take the following steps.

1. Open the following file in a text editor:

```
link_amazon_bwp/cartridges/int_buywithprime/cartridge/scripts/amazon/hooks/htmlHooks.js
```

2. Find the following script references in the `htmlHead` function:

```
// clickstreamLoader.js hash
Velocity.render('<script defer src="$url" integrity="sha384-
abC123dEf456GhI789jKl+012MnO345pQr678sTu901vWx234yZa567bCd890EfG"></script>', {
     url: URLUtils.staticURL('/js/clickstreamLoader.js')
 });

// buywithprime.js hash
Velocity.render('<script defer src="$url" integrity="sha384-
abC123dEf456GhI789jKl+012MnO345pQr678sTu901vWx234yZa567bCd890EfG"></script>', {
     url: URLUtils.staticURL('/js/buywithprime.js').toString()
 });
```

3. From the `js-asset-manifest.json` file, get the new SRI hash values for the `clickstreamLoader.js` and `buywithprime.js` entries under the `integrity` key. The following example contains placeholder SRI hash values.

```
"default/js/clickstreamLoader.js": {
  "src": "default/js/clickstreamLoader.js",
  "integrity": "sha384-abC123dEf456GhI789jKl+012MnO345pQr678sTu901vWx234yZa567bCd890EfG"
},
"default/js/buywithprime.js": {
  "src": "default/js/buywithprime.js",
  "integrity": "sha384-xyZ123aBC456GhI789jKl+012MnO345pQr678sTu901vWx234yZa567bCd890EfG"
},
```

4. In the `htmlHooks.js` file, replace both SRI hash values with their corresponding new SRI hash values from the `js-asset-manifest.json` file and keep the `sha384-` prefix.
5. Save the `htmlHooks.js` file.

Generate new SRI hash values manually

If your configuration requires you to build the cartridge separately, you might have to generate new SRI hash values manually. For details about how to generate new SRI hash values manually, in the Mozilla documentation see Subresource Integrity.