



Toshi Integration for SiteGenesis

22.1.0



[Table of Contents](#)

1. Component Overview	3
1.1. Functional Overview.....	3
1.2. Use Cases	3
1.3. Compatibility.....	3
1.4. Privacy, Payment.....	3
2. Implementation Guide.....	4
2.1. Before you start	4
2.2. Cartridge Upload & Assignment	5
2.3. Setup of Business Manager.....	7
2.3.1. Activate the cartridge in Business Manager.....	7
2.3.2. Importing metadata with a single site import	8
2.4. Configuration	13
2.4.1. Managing custom site preferences	13
2.4.2. Site Preferences configuration table.....	14
2.4.3. Store configuration	15
2.4.4. Shipping Method Configuration.....	15
2.5. Custom Code changes	16
2.5.1. Controllers.....	16
2.5.2. Scripts.....	16
2.5.3. Forms	16
2.5.4. Client JS	17
2.5.5. ISML changes.....	18
2.6. Jobs	20
2.6.1. toshiSendOrders.....	20
3. Testing	20
4. Release History	21

1. Component Overview

TOSHI collaborates with renowned global brands to offer their customers the convenience of in-store clienteling from the comfort of their own homes, on-demand.

1.1. Functional Overview

This guide provides a summary of all features implemented in the cartridge, as well as guidance on its installation within a Salesforce B2C Commerce environment. It also includes details about the required configurations and a checklist to ensure that the cartridge is properly installed and configured.

1.2. Use Cases

The cartridge provides implementation of a delivery method - 'Try Before you Buy', which is specifically designed for the integration of the booking service from TOSHI.

1.3. Compatibility

This cartridge has been tested against API Version: 22.6 (Compatibility Mode: 19.10) & SiteGenesis Controllers version 105.1.1

1.4. Privacy, Payment

This integration requires access to the following customer data elements: Billing Address, Shipping Address, Order Details

2. Implementation Guide

2.1. Before you start

To ensure a successful implementation, it is essential to have *SiteGenesis Controllers* installed and properly configured in a Salesforce B2C Commerce instance. If developers encounter issues with the setup of SiteGenesis, we recommend visiting the Salesforce support center for assistance.

If developers have not yet obtained the necessary account details for implementation, they should reach out to team TOSHI for further assistance. The account details are vital for the execution of the integration process.

2.2. Cartridge Upload & Assignment

The SiteGenesis package includes two cartridges which need to be uploaded on a Salesforce B2C Commerce instance to extend the native SiteGenesis functionality with the capabilities provided by the cartridge:

- ***int_toshi*** – encompasses a wide range of general functionalities, including essential Helpers, Services, and Jobs.
- ***int_toshi_sitegenesis*** - utilization of SiteGenesis Controllers, such as COBilling, COShipping, and COSummary, which have been modified to accommodate the requirements of Toshi. Additionally, SiteGenesis forms have been implemented specifically for Toshi, incorporating necessary adjustments.

A step-by-step guide on how to add the cartridges - *int_toshi* and *int_toshi_sitegenesis*, to an existing SFCC codebase using Visual Studio Code (VS Code) is outlined below:

1. Open your SFCC codebase in VS Code

Launch VS Code and navigate to the root folder of your SFCC codebase using the "File" > "Add Folder to Workspace" option.

2. Upload cartridges into a Salesforce B2C Commerce instance

Using the WebDAV extension in VS Code, navigate to the appropriate location in your SFCC instance where the cartridges need to be uploaded.

In VS Code, right-click on ***int_toshi*** and ***int_toshi_sitegenesis*** cartridges in the local folder where they were extracted and select '*Deploy to WebDAV*' from the context menu.

Select the target location of your Salesforce B2C Commerce instance where you want to upload the cartridges and click '*Deploy*'. This will upload the cartridges to the instance.

3. Add the cartridges to the cartridge path

Once the cartridges are uploaded to your SFCC instance, you need to add them to the cartridge path in your SFCC codebase. Open the '*dw.json*' file in the root folder of your codebase and add the following entries to the '*cartridges*' section:

```
"cartridges": [  
  "int_toshi",  
  "int_toshi_sitegenesis"  
]
```

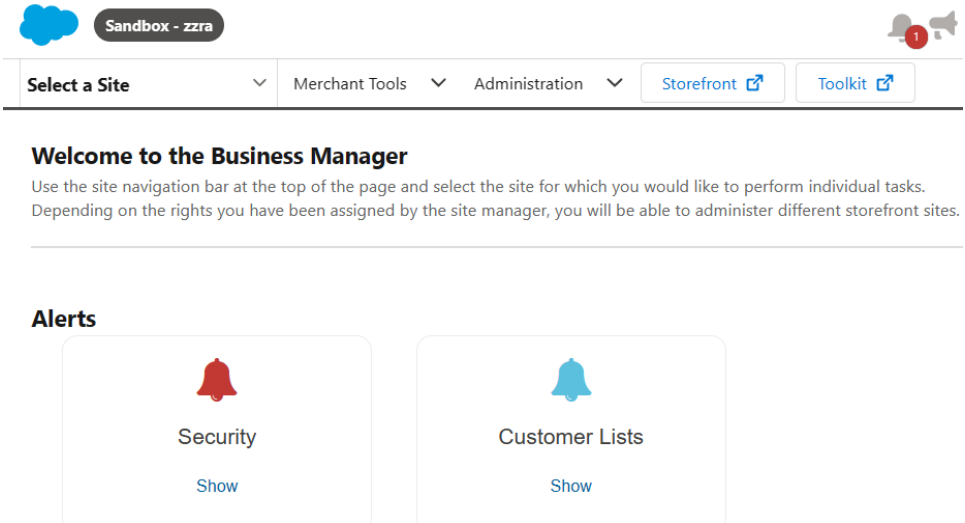
4. Save the "dw.json" file.

2.3. Setup of Business Manager

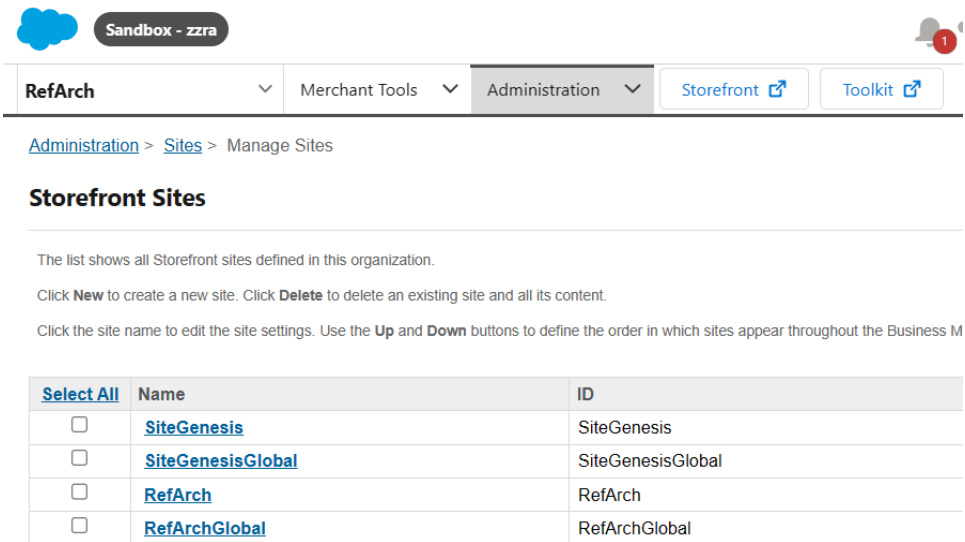
2.3.1. Activate the cartridge in Business Manager

To enable the functionality, developers must add the cartridges to the cartridge path of the storefront sites as follows:






1. Log in to the Salesforce B2C Commerce Business Manager:



2. Go to *Administration > Sites > Manage Sites*



3. Click on the site name.

 **Sandbox - zzra**    

RefArch ▼ Merchant Tools ▼ Administration ▼ [Storefront](#) [Toolkit](#)

[Administration](#) > [Sites](#) > Manage Sites

Storefront Sites

The list shows all Storefront sites defined in this organization.

Click **New** to create a new site. Click **Delete** to delete an existing site and all its content.

Click the site name to edit the site settings. Use the **Up** and **Down** buttons to define the order in which sites appear throughout the Business Manager.

Select All	Name	ID	Status
<input type="checkbox"/>	SiteGenesis	SiteGenesis	Online
<input type="checkbox"/>	SiteGenesisGlobal	SiteGenesisGlobal	Online
<input type="checkbox"/>	RefArch	RefArch	Online
<input type="checkbox"/>	RefArchGlobal	RefArchGlobal	Online

- The Settings tab must be selected, and the following details added to the cartridge path:

`int_toshi_sitegenesis:int_toshi:app_storefront_controllers:app_storefront_core`

[Administration](#) > [Sites](#) > [Manage Sites](#) > SiteGenesis - Settings

General **Settings** Cache Site Status Page Meta Tag Rules

SiteGenesis - Settings

Click **Apply** to save the details. Click **Reset** to revert to the last saved state.

Instance Type: Sandbox/Development ▼

Deprecated. The preferred way of configuring HTTP and HTTPS hostnames is by using new features of the site aliases configuration ("SEO > Aliases configuration" and are intended only to support an older configuration style).

HTTP Hostname:

HTTPS Hostname:

Instance Type: All

Cartridges:

Effective Cartridge Path:

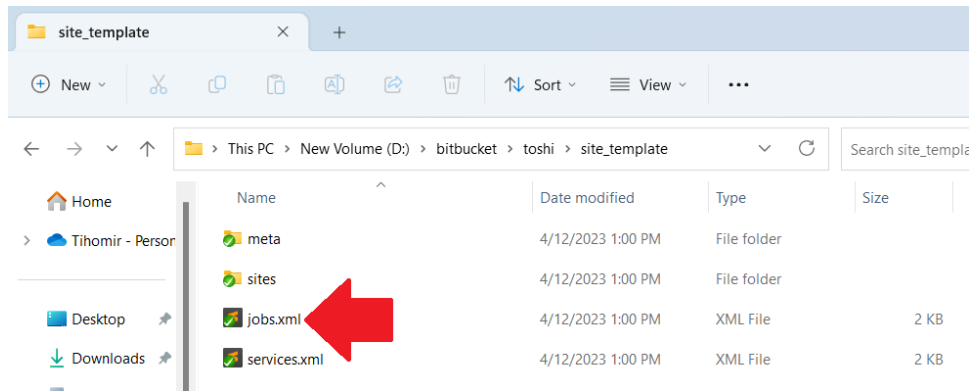
[Back to List](#)

- Click **Apply**.
- Repeat steps 3 to 6 for each site that requires the functionality provided by TOSHI.

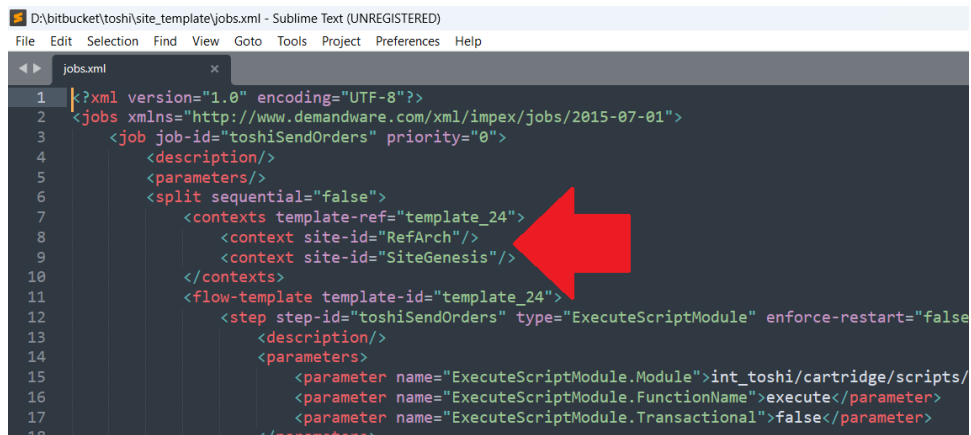
2.3.2. Importing metadata with a single site import

All import files are located in the import folder - *toshi/site_template*. To import the required settings, follow the steps below:

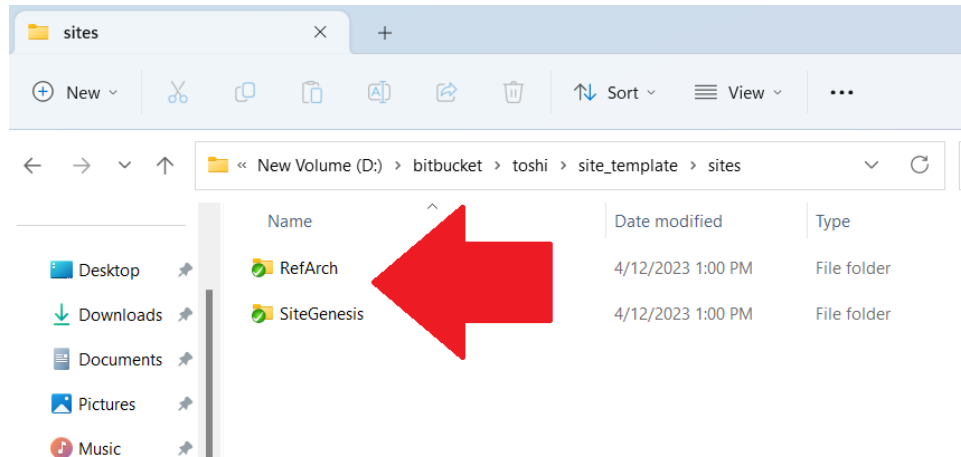
1. Open the file *toshi/site_template/jobs.xml*



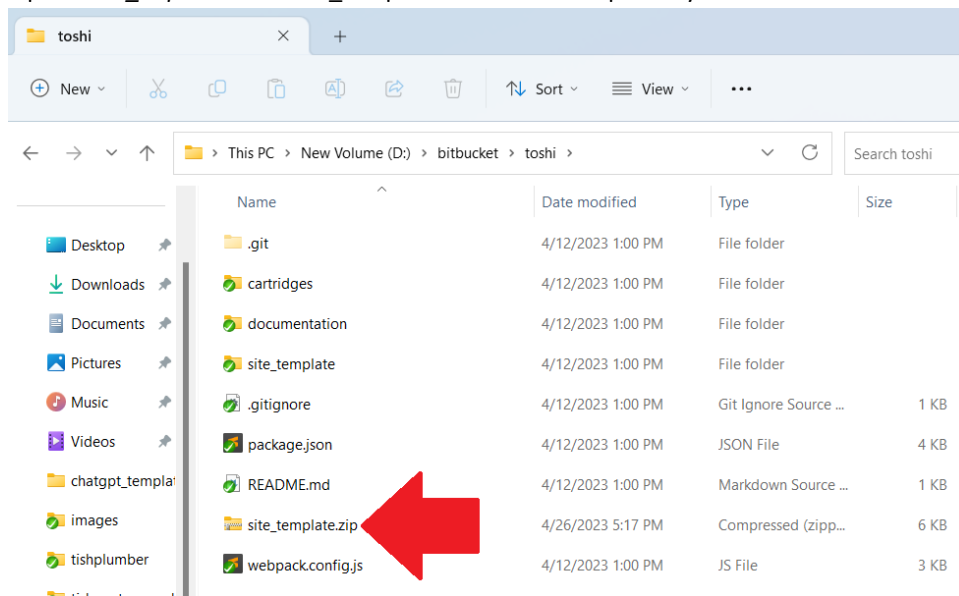
2. Change the site ID references of *RefArch* and *SiteGenesis* to your site IDs. Developers could also do it after the job is already imported.



3. Ensure that the folders for sites in the *site_template/sites* (toshi/site_templates/sites/RefArch and toshi/site_templates/sites/SiteGenesis) have the appropriate site ID of the sites where the XML is imported.
 1. Change the folder name *RefArch* to your first site ID (Apply the same for SiteGenesis path to second site ID)
 2. Then copy and rename this folder to the site IDs for other sites you are planning to use Toshi.

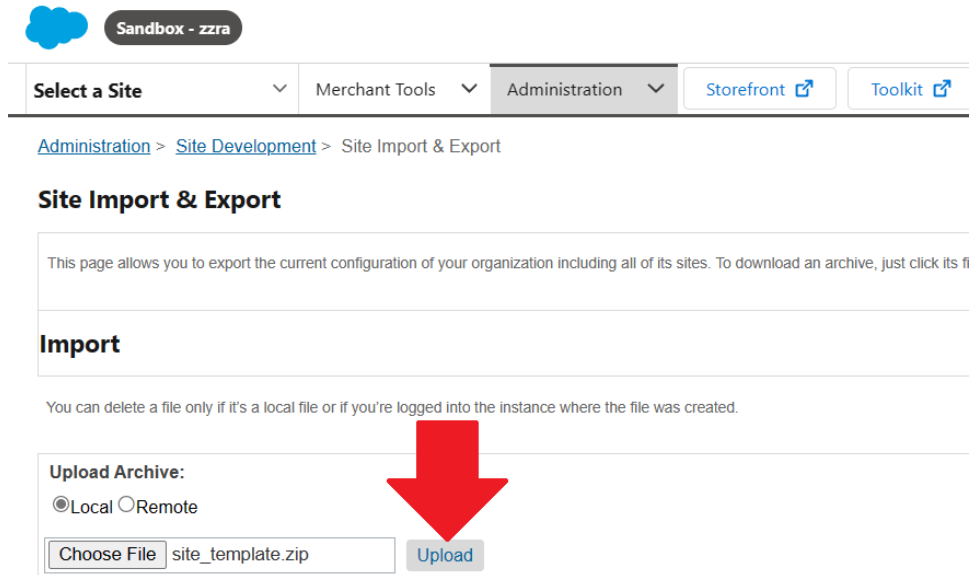


4. Zip the *site_import* folder site_template folder in the repository structure



5. Log in to Business Manager and go to *Administration/Site Development/Site Import & Export*

6. Click the **Upload** button to upload the zipped *site_import* folder



Sandbox - zzra

Select a Site ▾ Merchant Tools ▾ Administration ▾ [Storefront](#) [Toolkit](#)

[Administration](#) > [Site Development](#) > Site Import & Export

Site Import & Export

This page allows you to export the current configuration of your organization including all of its sites. To download an archive, just click its file name.

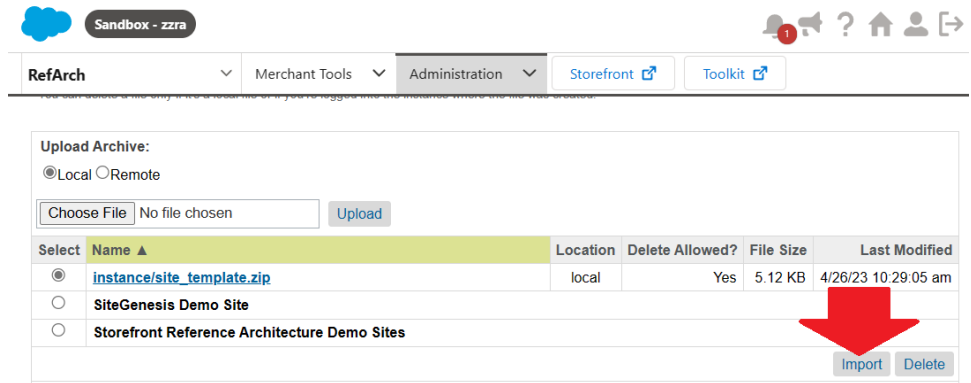
Import

You can delete a file only if it's a local file or if you're logged into the instance where the file was created.

Upload Archive:
☒ Local ☐ Remote

site_template.zip

7. Select the uploaded file and click the **Import** button.



Sandbox - zzra

RefArch ▾ Merchant Tools ▾ Administration ▾ [Storefront](#) [Toolkit](#)

[Administration](#) > [Site Development](#) > Site Import & Export

Site Import & Export

This page allows you to export the current configuration of your organization including all of its sites. To download an archive, just click its file name.

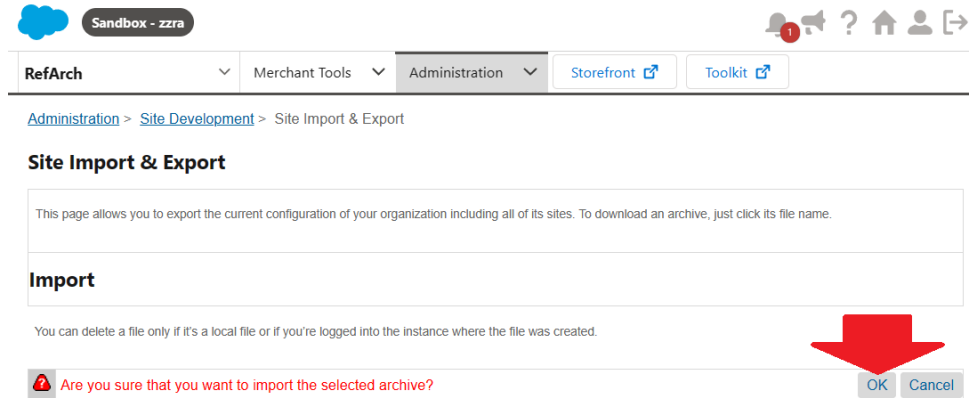
Import

You can delete a file only if it's a local file or if you're logged into the instance where the file was created.

Upload Archive:
☒ Local ☐ Remote

No file chosen

Select	Name ▲	Location	Delete Allowed?	File Size	Last Modified
<input checked="" type="radio"/>	instance/site_template.zip	local	Yes	5.12 KB	4/26/23 10:29:05 am
<input type="radio"/>	SiteGenesis Demo Site				
<input type="radio"/>	Storefront Reference Architecture Demo Sites				



Sandbox - zzra

RefArch ▾ Merchant Tools ▾ Administration ▾ [Storefront](#) [Toolkit](#)

[Administration](#) > [Site Development](#) > Site Import & Export

Site Import & Export

This page allows you to export the current configuration of your organization including all of its sites. To download an archive, just click its file name.

Import

You can delete a file only if it's a local file or if you're logged into the instance where the file was created.

Are you sure that you want to import the selected archive?

After a successful import, all cartridge configurations will be available as per the account data provided for the merchant from team TOSHI.

- Price Books
- Inventory Lists
- Customer Lists
- Global Data
- Assignments

All product images

All price books

All inventory lists

All customer lists

All global data

All assignments

Status

Select All	Process	Start	Duration	Status
<input type="checkbox"/>	Site Import (site_template.zip)	4/26/23 10:32:32 am	00:00:02	Success

[Refresh](#)
[Delete](#)

- Verify the custom attributes for the Order object in Business Manager - go to *Site > Ordering > Order*. Open any order and go to the *'Attributes'* tab. It should look like the screenshot below:

Sandbox - zzra

RefArch

Merchant Tools

Administration

Storefront

Toolkit

[Merchant Tools](#) > [Ordering](#) > [Orders](#) > Order: 00002105(RefArch)

[General](#)
[Attributes](#)
[Payment](#)
[Notes](#)
[History](#)

Attributes for Order '00002105'

On this page you can edit the attributes of the order. Fields with a red asterisk (*) are mandatory. Click [Apply](#) to save changes. Click [Reset](#) to revert your changes.

Toshi

Toshi Checkout ID:

Order sent to Toshi: - None -

Toshi Store ID:

Salesforce Service Cloud

sscid:

2.4. Configuration

2.4.1. Managing custom site preferences

The integration must be configured with the account details provided by team TOSHI, by following the steps below:

- Access the Business Manager of the Salesforce B2C Commerce environment and navigate to:
 - *Site > Merchant Tools > Site Preferences > Custom Preferences*
- Locate the preference group associated with TOSHI:
 - *Site > Merchant Tools > Site Preferences > Custom Preferences > Toshi*
- Click on the Group Name or ID to apply the account data configurations provided by team TOSHI

2.4.2. Site Preferences configuration table

To access the preferences related to Toshi, please navigate to: *Site > Merchant Tools > Site Preferences > Custom Preferences > Toshi.*

Site Preference Name	Description	Sample values
Enable Toshi	The flag manages Toshi integration. If set to True, Toshi will be enabled for the site	True
Toshi Api Key Configuration	JSON holding the config for the api key per country/city	<pre>{ "gb" : { "client" : "0e8ddc700a014d74a9cc968b1ee2b9ad", "server" : "88a37d6059a240029f8c41c75dde868d" }, "us" : { "client" : "5fc8074bd4ca4bab90767ccddde8ce09", "server" : "d5585ef51cf74a7ca21679987a01fa7e" } }</pre>
Disable other shipping methods when Toshi is present	When Toshi is enabled, it you can disable other shipping methods	No
Toshi Notification emails	Comma separated list of emails to send error in case of webservice error	
Basket timeout	Number of minutes customer will be redirected back to shipping page	30
Minimal Amount for Toshi method	JSON string containing currency and amount for example { "USD" : xy.00 }	<pre>{ "USD" : 2, "GBP" : 2 }</pre>
Toshi No Size Value	Value which to be sent to Toshi in case no size variants	N/A
Toshi Modal URL	Modal API url passed in the javascript constructor	https://staging.toshi.co

2.4.3. Store configuration

To configure the store details which will be utilized for the delivery method of TOSHI, please follow these steps:

1. Navigate to the Business Manager of the Salesforce B2C Commerce environment
2. Navigate to *Site > Merchant Tools > Online Marketing*
3. Click on '*Stores*' to access the store settings
4. Update the '*Toshi Store*' option to 'Yes' for the desired shop.

Please note that for this integration it is **recommended** to have one shop per country.

2.4.4. Shipping Method Configuration

Please consider implementing a new shipping method and configuring its Toshi flag to either 'checkout' or 'Try Before You Buy'. It is essential to ensure that only one Toshi delivery method is present at a time, based on the customer groups' configuration.

2.5. Custom Code changes

2.5.1. Controllers

File: app_storefront_controllers/cartridge/controllers/COSummary.js

In the function showConfirmation add at the top

```
var Transaction = require('dw/system/Transaction');
var toshiClient = require('*/cartridge/scripts/toshi/toshiClient');
if (order.custom.sendToToshi && order.custom.sendToToshi.value === 1){
    var orderSentResult = toshiClient.sendOrder(order);
    if (orderSentResult){
        Transaction.wrap(function(){
            order.custom.sendToToshi = 2;
        });
    }
}
```

2.5.2. Scripts

File: app_storefront_controllers/cartridge/scripts/app.js

change function getController

```
return require('~cartridge/controllers/' + controllerName);
```

To:

```
return require('*/cartridge/controllers/' + controllerName);
```

2.5.3. Forms

File: app_storefront_core/cartridge/forms/default/singleshipping.xml

Add following group in shippingAddress group:

```
<!-- email field is contained in separate form group to enable binding to customer profile -->
    <group formid="email">

        <field formid="emailAddress" label="billing.email" type="string" mandatory="true" regexp="^[\\w.%+-]
        ]+@[\\w.-]+\\.([\\w]{2,6})$" binding="email" max-length="50" missing-error="address.email.invalid" range-
        error="address.email.invalid" parse-error="address.email.invalid" value-error="address.email.invalid"/>
```



```
</group>
```

2.5.4. Client JS

File: app_storefront_core/cartridge/js/pages/checkout/shipping.js

Under var shippingMethods; add var toshiTimeSelected = false;

In updateShippingMethods function callback add

```
initToshi();
and after it add function
function initToshi(){
  var $form = $('address');
  var toshiMethods = $("#shipping-method-list").find('[data-istoshi=true]');
  if (toshiMethods.length > 0 ){
    var method = $(toshiMethods[0]);
    var key = method.data('toshikey');
    var mode = method.data('toshi-type');
    const modal = window.toshi.createBookingIntegration({
      api: {
        url: toshiClient.toshiModalUrl,
        key: key
      },
      ui: {
        mode: mode
      }
    });

    // This is fired by the ecommerce integration when the customer attempts to
    // proceed without selecting a timeslot.
    window.showErrorMessage = () => {
      modal.setInlineErrorMessage(
        toshiClient.errorMsg
      );
    };
    window.hideErrorMessage = () => {
      modal.setInlineErrorMessage(undefined);
    };

    modal.setFirstName($form.find('input[name$="_firstName"]').val());
    modal.setLastName($form.find('input[name$="_lastName"]').val());
    modal.setPhone($form.find('input[name$="_phone"]').val());
    $form.find('input[name$="_phone"]').change(function(e){
      modal.setPhone($form.find('input[name$="_phone"]').val());
    });
    modal.setEmail($form.find('input[name$="_emailAddress"]').val());
```

```

modal.setAddress({
  addressLine1: $form.find('input[name$="_address1"]').val(),
  addressLine2: $form.find('input[name$="_address2"]').val(),
  town: $form.find('input[name$="_city"]').val(),
  province: $form.find('select[id$="_country"]').val(),
  postcode: $form.find('input[name$="_postal"]').val(),
  country: $form.find('select[id$="_country"]').val()
});
$form.find('input[name$="_emailAddress"]').on('change', function(){
  modal.setEmail($form.find('input[name$="_emailAddress"]').val());
});

modal.setProducts(window.toshiClient.products);
modal.setBrandCheckoutReference(window.toshiClient.checkoutID);

modal.mount($('.shipping-method-toshi')[0]);
modal.onOrderCreated(function (e){
  toshiTimeSelected = true;
  method.prop("checked", true);
  method.trigger('change');
});
}
}
Before giftMessageBox(), add
$('#dwfrm_singleshipping_shippingAddress').on('submit', function(e){
  var isToshiMethod = $('#shipping-method-list').find("[name$='_shippingMethodID']:checked").data('istoshi');
  if (isToshiMethod && !toshiTimeSelected){
    e.preventDefault();
    showErrorMessage();
  } else {
    hideErrorMessage();
  }
});

And in the end of the file add

window.showErrorMessage = () => {
  modal.setInlineErrorMessage(
    toshiClient.errorMsg
  );
};
window.hideErrorMessage = () => {
  modal.setInlineErrorMessage(undefined);
};

```

2.5.5. ISML changes

File: app_storefront_core/cartridge/templates/default/checkout/cart/cart.isml

add

```
<isif condition="{dw.system.Site.getCurrent().getCustomPreferenceValue('toshiEnabled')}">
    <isinclude url="{URLUtils.https('Toshi-Cart')}" />
</isif>
```

Where you want the content asset for the country if configured to be displayed – the id of the content asset needs to be the format of the content asset id needs to be toshi-cart-{country} – where the country is taken from the customer geolocation and is upper case – for example toshi-cart-GB will display content if the customer is coming from GB.

File: app_storefront_core/cartridge/templates/default/checkout/shipping/singleshipping.isml

After the dynamic address form, add:

```
<inputfield formfield="{pdct.CurrentForms.singleshipping.shippingAddress.email.emailAddress}"
type="input"/>
```

In both Files:

app_storefront_core/cartridge/templates/default/checkout/product/productcontent.isml

app_storefront_core/cartridge/templates/default/checkout/product/producttopcontentPS.isml

Add

```
<isif condition="{dw.system.Site.getCurrent().getCustomPreferenceValue('toshiEnabled')}">
    <isinclude url="{URLUtils.https('Toshi- PDP')}" />
</isif>
```

Where you want the content asset for the country if configured to be displayed – the id of the content asset needs to be the format of the content asset id needs to be toshi-pdp-{country} – where the country is taken from the customer geolocation and is upper case – for example toshi-pdp-GB will display content if the customer is coming from GB.

2.6. Jobs

2.6.1. toshiSendOrders

This job is responsible for interfacing with the Toshi API to process orders and manage payments in the SFCC environment. It includes functions for checking address eligibility, sending orders to Toshi, and retrieving payment and line item details from orders.

The main functionalities of the 'execute' function are as follows:

5. 'checkAddressEligible' function: This function takes an 'address' object as an input and checks if the address is eligible for Toshi service by making a call to the Toshi API. It retrieves the country from the address object, gets the Toshi configuration from the site's custom preferences, and then makes a call to the Toshi API to check the eligibility of the address. If the address is eligible, it sets a session privacy variable with the Toshi client key.
6. 'sendOrder' function: This function sends an order to the Toshi service by creating a request object with various order details such as customer information, billing and shipping addresses, order total, currency, payment status, gift status, and payment provider. It also retrieves the line item details from the order and adds it to the request object. The function then makes a call to the Toshi API with the request object to create an order. If the order creation is successful, it sets the 'orderSent' variable to 'true'.
7. 'getPaymentInstruments' function: This function retrieves the payment instruments associated with an order and returns an array of payment method names.
8. 'getLineItemCntrContent' function: This function retrieves the line item details from a product line item container (e.g., an order or a basket) and returns an array of line item objects containing product details such as SKU, name, price, quantity, and total price.

3. Testing

After successfully installing and integrating the cartridge as per the provided instructions, it is recommended to test the functionality by placing an order on your sandbox environment.

During testing, please ensure that the Toshi payment method is presented and selectable as expected. For testing purposes, you may need to obtain test addresses from Toshi, which can be used to validate if the webservice returns that the address is eligible for Toshi.

4. Release History

Version	Date	Changes
22.1.0	2022-09-20	SFRA support
21.1.0	2021-04-15	Initial release