Procedure

1. In the search field, enter `html`.
2. From the search results, click **HTML Forms Login Policy**.
3. Click **Add** or **New**.
4. Enter the name of the policy.
5. Optional: In the **General** section, enter the required information.
    a. Set the administrative state of the configuration.
    b. In the **Comments** field, enter a descriptive summary.
    c. From the **Source for Form-processing** list, select **Static** to specify the explicit location of the HTML pages or **Custom** to use a custom file that generates the HTML pages and the form. The default support type for HTML forms-based login processing is **Static**.

- For **Static**, select how to retrieve host information for HTTP URL-redirects from the **Redirect URL Type** list.
- For **Custom**, select the location and file name from the **Custom Processing for Form** lists. The example `store:///Form-Generate-HTML.xsl` custom file is available. This support type is not available to the Web Application Firewall.

6. Optional: In the **Security** section, override the default security configuration. By default, **Use TLS for login** is enabled. The defined port must correspond to the TLS port of the service.
    a. If you select **Static** from the **Source for Form-processing** list, specify whether to use TLS or not to retrieve login information.

b. When you use TLS, enter the TLS port for the service.

c. Specify whether users must re-authenticate if their session moves to another DataPower® Gateway in the standby group if the current DataPower Gateway becomes unavailable.

   Whether another DataPower Gateway is available depends on the standby control configuration for the network interface.

   - For use existing authentication, set **Enable Session Migration** to **on**.
   - To require re-authentication, set **Enable Session Migration** to **off**.

d. When you use session migration, specify the shared secret.
   In a standby configuration, each DataPower Gateway in the group must use the same shared secret.
e. Optional: Specify a cookie attribute policy to allow predefined or custom attributes to be included in the forms login cookie.

7. Optional: In the **Client-side URL fragments** section, override the default fragments.
   a. Select the path fragments for the login, logout, and error pages.
   b. For the **Default URL** field, enter the URL of the web page to display after a user successfully logs in if the user went directly to the login page. Otherwise, the user is sent to the secured web page that the user was attempting to contact before the user is redirected to the login form.

8. In the **Location of HTML pages** section, indicate whether the web pages are local or remote.
    - When local, you can accept the default location of the web pages or you can override the default locations.
    - When remote, which in on an application server, the pages use client-side URL fragments as maps to the application server.

9. Optional: In the **Login form properties** section, override the HTML form content for the login page.
10. Optional: In the **Timeouts** section, modify the default values.
11. Click **Apply** to save the changes to the running configuration.
12. Click **Save Configuration** or **Save changes** to save the changes to the persisted configuration.

1. Add custom HTML code to a view:
    a. In a toolkit or process application, create a new view named *getAccountTypes*.
    b. In the Layout page, click the plus sign to add a **Custom HTML** item onto the canvas.
    c. In the properties under **HTML**, select the **Text** option and then provide the custom HTML code. For this example, you can use the following code to define a select view:

```
<select name="AccountType" s
  <option value="Savings">Sa
  <option value="Current">Cu
</select>
```

    d. On the Overview page, select **Can Fire Boundary Event**.

a. Register the Dojo button module and alias that the view will load dynamically.
    i. In the Behavior page, select **AMD dependencies**.
    ii. Click **Add** and then specify the following information:

        In the **Module ID** column, type `dojo/_base/connect`. This declares a dependency on the module that provides event handling for DOM nodes and related functionality.

        In the **Alias** column, type `connect`. This is the alias used in the code to refer to the `connect` module.

b. Under **Event Handlers**, select **load** and then provide a custom script. For this example, you can use the following script:

```
var selectElement = this.con

var onChangeHandle = connect
if(this.context.binding){
var tempBinding = this.conte
tempBinding.set("TSAPP_Accou
}
});
```

Reference

Table 2. Example business objects

Library item	Example name
Business Objects	TSAPP_ValidateDocumentCase Parameters: TSAPP_Zipcode (String) TSAPP_Age (String) TSAPP_AccountStatus (String) TSAPP_CustomerType (String) TSAPP_Name (String) TSAPP_City (String) TSAPP_AccountType (String)