This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Developer's Guide

DOM Restrictions

12/14/2025

DOM Restrictions

Contents

Genesys Co-browse allows you to hide sensitive data from agents and restrict control of elements in a co-browse session by providing a map of the elements that should be restricted. For every element in this map, there should be a CSS selector which identifies it and the type of restriction applied.

You can implement two types of restrictions:

- Data masking
- DOM control

> **Important**
>
> - Data masking and DOM controls apply to both Pointer Mode and Write Mode.
> - DOM restrictions do not support images.

For details about DOM restrictions, see Configuring DOM Restrictions.

Data Masking

Private, critical, or sensitive customer-related data can be masked on the agent side with asterisks. This masked data does not leave the customer browser; it is not retrieved by Co-browse Server or the agent browser.

If agents try to change a masked input field, they see a notification that only the customer can access the input field.

Data masking can be applied to any visual HTML element. For input elements, the `value` attribute is masked; for all other elements, the text content is masked.

On the customers' side, if the data masked input field is in focus (for example, a customer selects an input field to enter a credit card number), customers see a notification that the information they type is not visible to agents.

> **Important**
>
> Data masking for all password inputs is enabled in the system and cannot be changed by Configuring DOM Restrictions.

DOM Control

DOM Control allows you to disable web page elements that agents should not be able to use. An example could be some buttons that perform some type of action on your web site, which must be available only for the users, not the agents.

> **Important**
>
> Buttons of type `submit` are disabled for agents by the default Co-browse DOM Restrictions configuration.

Configuring DOM Restrictions

To implement DOM restrictions, you must create an XML file to store your configuration. In this XML file, you configure rules for data masking and DOM control. Sets of rules match to specific pages or groups of pages using regular expressions (regexp).

Creating an XML configuration file

Create an XML configuration with the following structure:

```
<?xml version="1.0" encoding="UTF-8" ?>
<domRestrictions>
    <restrictionsSet>
        <uriTemplate type="regexp" pattern="<URL matching regex>"/>
        <dataMasking>
            <element selector="..."/>
        </dataMasking>
        <domControl>
            <element selector="[type=submit]"/>
        </domControl>
        <dataMasking/>
    </restrictionsSet>
</domRestrictions>
```

For a detailed example, see XML configuration file example.

After you create a DOM restrictions configuration file, you must link the XML configuration file to your Co-browse server.

XML structure description

The XML configuration file contains the following elements:

- `<domRestrictions>`—root tag containing any number of `restrictionsSet` tags.

  ```
  <domRestrictions>
      <restrictionsSet>
          ...
  ```

```
    </restrictionsSet>
    <restrictionsSet>
        ...
    </restrictionsSet>
</domRestrictions>
```

- `<restricitonsSet>`—defines restriction rules applied to a web page or group of pages matching the regular expression in the `pattern` attribute. Restriction sets are cumulative. More than one restriction set can apply to a single webpage.

  ```
  <restrictionsSet>
      <uriTemplate type="regexp" pattern="..."/>
      <dataMasking>
          ...
      </dataMasking>
      <domControl>
          ...
      </domControl>
  </restrictionsSet>
  ```

- `<uriTemplate>`—defines the set of web pages the restriction applies to using a URL matching regex pattern.

  ```
  <uriTemplate type="regexp" pattern="<URL matching regex>"/>
  ```

  `pattern` value:

  - The `pattern` value is a regular expression (regex) that matches the URL of the target page. For more about regular expressions, see http://www.regular-expressions.info/.
  - Some characters have special meaning in regular expression syntax. When using these characters, you must *escape* the characters using a backslash (\). URL characters you must escape: `.:()\/?*|+{}^$[]`.
  - For example, the regex for the URI `http://www.genesys.com/about-genesys/contact-us` can be:

    ```
    http:\/\/www\.genesys\.com\/about-genesys\/contact-us
    ```

    or simply,

    ```
    www\.genesys\.com\/about-genesys\/contact-us
    ```

    > **Tip**
    > You may use online tools like Regexper to validate your regular expressions.

    `pattern` examples:

Regex	Description
`.*`	Matches any page
`login\.html`	Matches all pages that include `login.html` in the URL
`(login|registration)-page\.html`	Matches pages prefixed with `login` or `registration` such as `login-page.html` and `registration-page.html`.

Regex	Description
`genesys\.com\/about-genesys\/contact-us`	Matches pages like `http://www.genesys.com/about-genesys/contact-us` and `https://genesys.com/about-genesys/contact-us`.
`(https\:\/\/)`	Matches all HTTPS pages

- `<dataMasking>`—list of all web elements whose data should be masked.

```
<dataMasking>
    <element selector="..."/>
</dataMasking>
```

- `<domControl>`—list of all web elements that should be restricted from agent control.

```
<domControl>
    <element selector="..."/>
</domControl>
```

- `<element>`tag describing which element(s) to restrict.

```
<element selector="<jQuery specific selector>"/>
```

`element` examples:

Value	Description
`<element selector="#sendRequest"/>`	Elements with id="sendRequest"
`<element selector="[name=login]"/>`	Elements with name="login"
`<element selector="[type=submit]"//>`	Elements with type="submit"
`<element selector=".SendButton"/>`	Elements with "SendButton" class
`<element selector="[href='/about-us/contacts']"/>`	Elements with href="/about-us/contacts"
`<element selector="[href$='.org']"/>`	Elements with href attribute ending with ".org"
`<element selector=":button"/>`	All normal buttons
`<element selector="[type=submit]:not(#uniqueSubmitId)"/>`	All `submit` buttons without id = "uniqueSubmitId"
`<element selector=".Input:not(#InputId2)"/>`	All elements with `.Input` class and without id = "InputId2"

For more information about selecting elements from a webpage see Using browser tools to select an element.

XML structure summary

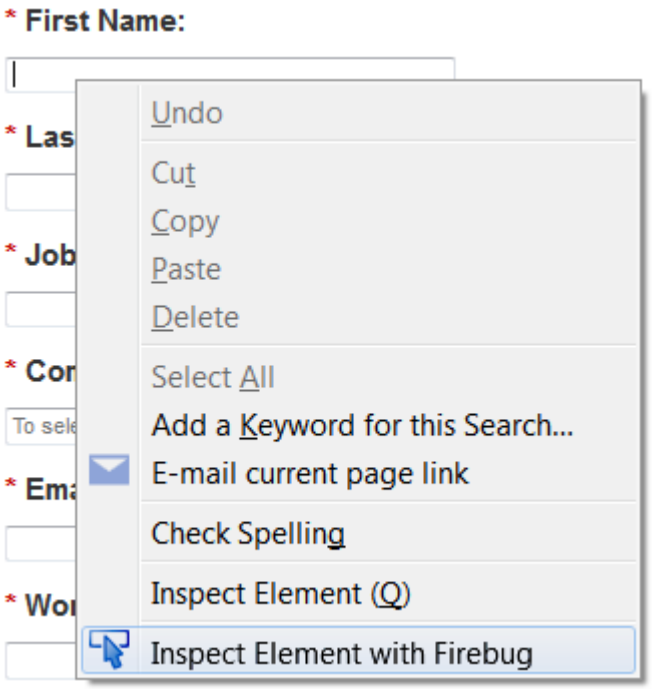
For each matched page or group of pages defined by a `pattern` in a `<uriTemplate>` tag, you can provide data masking and DOM control rules for specific page elements. Elements are represented by `<element>` tags. Each `<element>` contains a `selector` attribute with a jQuery selector describing the path to the element on the page.

Using browser tools to select an element

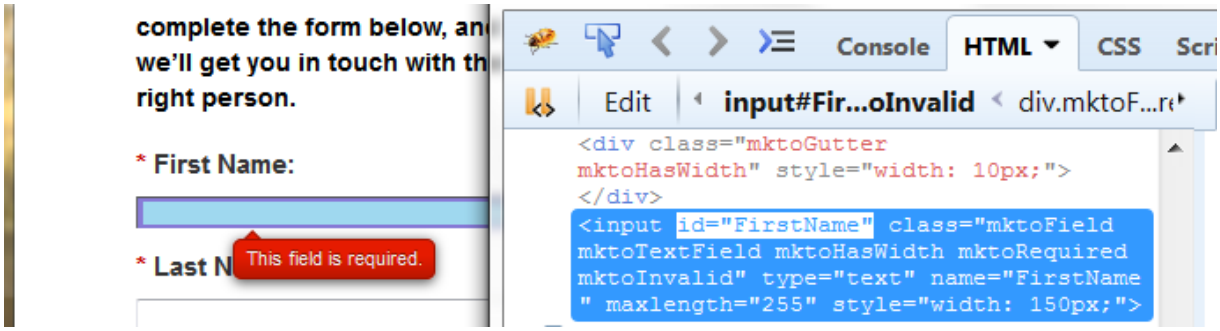
You can use browser tools to help you define an `element selector` in your XML configuration files. The example below uses the Firebug browser tool for Firefox.

Using Firebug to find an element selector

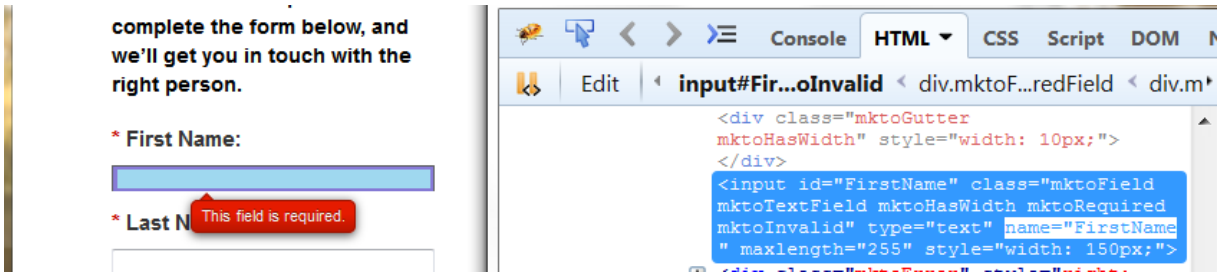
1. Open the webpage containing the element you want to select. Right click the element and click `Inspect Element with Firebug` to open the Firebug tool.



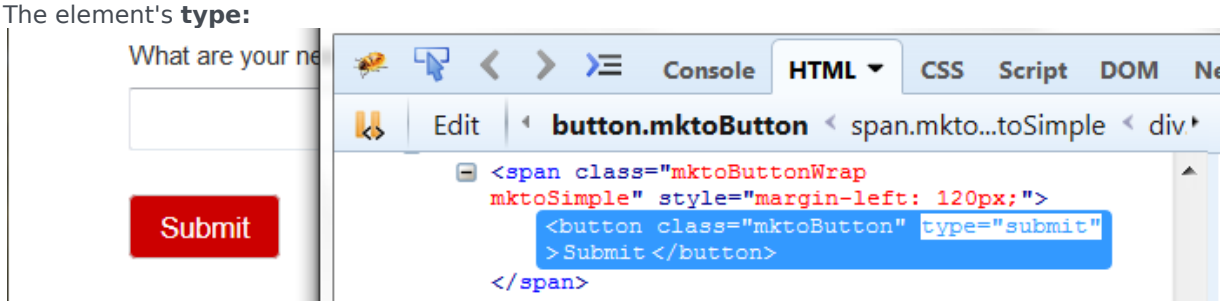
2. Firebug can help you identify the right selector for an element. For example, you could use one of the following as a selector:
   - The element's **id:**



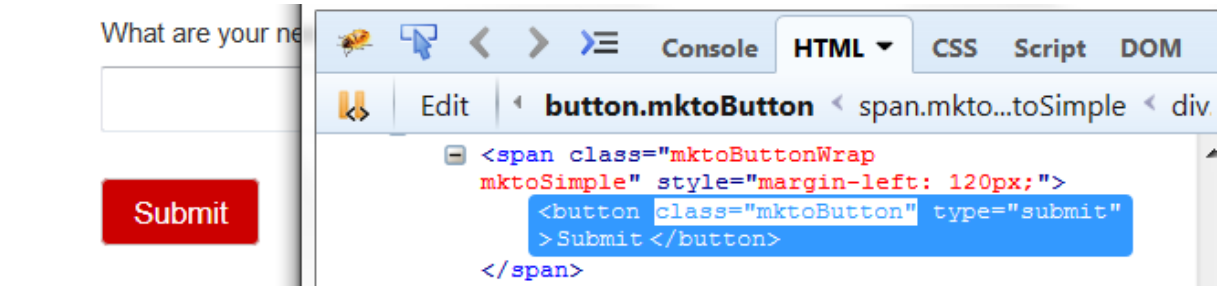
   - The element's **name:**



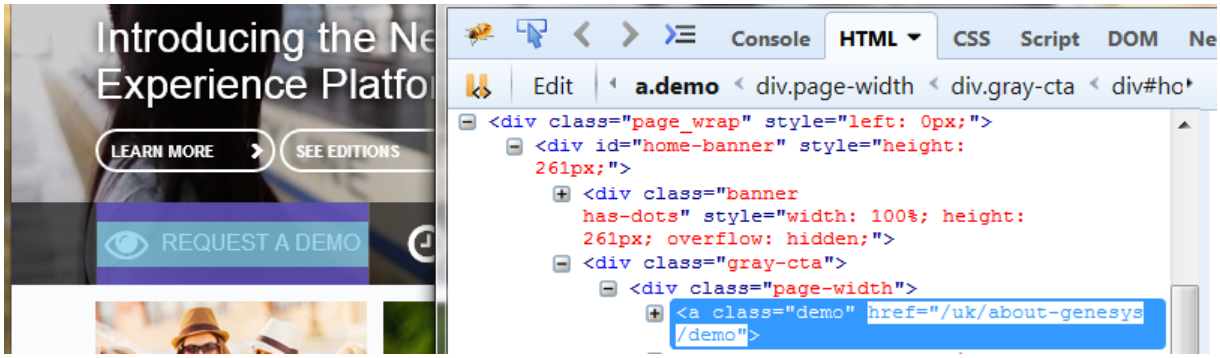
- The element's **type:**



- The element's **class:**



- The element's **href:**



> **Tip**
>
> For a comprehensive list of JQuery selectors, see http://api.jquery.com/category/selectors/.

Default DOM Restrictions Configuration

The default configuration ensures DOM control for all submit buttons is restricted from agents.

> **Important**
>
> To preserve this default behavior, create your custom configuration by extending and not overwriting the default configuration.

```
<?xml version="1.0" encoding="UTF-8" ?>
<domRestrictions>
    <restrictionsSet>
        <uriTemplate type="regexp" pattern=".*"/>
        <domControl>
            <element selector="[type=submit]"/>
        </domControl>
        <dataMasking/>
    </restrictionsSet>
</domRestrictions>
```

> **Important**
>
> Data masking for all password inputs is enabled in the system and cannot be changed using DOM restrictions configuration.

XML configuration file example

The example below provides a sample configuration with comments explaining the purpose of each element.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Use this configuration as a set of examples and common documentation -->
<!--
    domRestrictions contains any number of restrictionsSet elements.
    There must be only one domRestrictions element.
-->
<domRestrictions>
    <!--
        Each set defines restrictions rules matched by URL.

        Starting with including restrictionSet from default configuration,
        so that agents can never click submit buttons on behalf of customers.
    -->
    <restrictionsSet>
        <!-- Pattern ".*" matches any string and therefore any URL -->
        <uriTemplate type="regexp" pattern=".*"/>
        <domControl>
            <!-- All submit buttons (elements with type = "submit" ) will be restricted -->
            <element selector="[type=submit]"/>
        </domControl>
        <dataMasking/>
    </restrictionsSet>
```

```
    <!--
         All domControl and dataMasking rules in this restrictionSet will apply
        to pages that have "page.html" in their URL
    -->
    <restrictionsSet>
        <uriTemplate type="regexp" pattern="page\.html"/>
        <domControl>
            <element selector=":button"/> <!-- All normal buttons -->
            <element selector="#mySubmitButton"/> <!-- Concrete element with id =
"mySubmitButton" -->
            <element selector=".MySubmitButton"/> <!-- All elements that have class
"MySubmitButton" -->
            <element selector="[href='/checkout']"/> <!-- All links that lead to /checkout
page -->
        </domControl>
        <dataMasking>
            <element selector="[name=login]"/> <!-- All elements with name="login" will be
masked -->
            <element selector=".LicenseCode"/> <!-- All elements with class "LicenseCode"
will be masked -->
        </dataMasking>
    </restrictionsSet>
    <restrictionsSet>
        <!-- Range of pages pattern. -->
        <uriTemplate type="regexp" pattern="genesys\.com\/page[1-9]\.html"/>
        <domControl>
            <!--  Element with id = "uniqueSubmitId" will be excluded from restriction -->
            <element selector="[type=submit]:not(#uniqueSubmitId)"/>
        </domControl>
        <dataMasking/>
    </restrictionsSet>
</domRestrictions>
```

Linking the XML configuration file to your Co-browse Server

After you have created your configuration file, set the value of the domRestrictionsURL configuration option from the session section to point to your file. You can configure this value as:

- a URL reachable by Co-browser Server.
- a path to the file pre-fixed with `file`. For example, `domRestrictionsURL=file:C:\restrictions.xml`.

> **Important**
>
> Do not use the /static folder for storing the DOM restrictions XML file.