Oracle® Enterprise Single Sign-on Logon Manager

Best Practices: Template Configuration and Diagnostics for Web Applications

Release 11.1.1.2.0

E20408-01

Oracle Enterprise Single Sign-on Logon Manager Best Practices: Template Configuration and Diagnostics for Web Applications

Release 11.1.1.2.0

E20408-01

Table of Contents

Introduction

About This Guide

This guide describes best practices and recommended procedures for creating and configuring Web application templates. Topics covered include configuring and testing logon and password change forms, as well as diagnosing and resolving most common Web application response issues.

Prerequisites

Readers of this guide must have a working knowledge of the HyperText Markup Language (HTML), Web page structure, and be comfortable reading and deconstructing HTML and CSS code. Readers should also be familiar with deploying and configuring the ESSO-LM Agent and using the ESSO-LM Administrative Console. (Detailed information about each function described in this guide is available in the ESSO-LM Administrative Console help.)

Terms and Abbreviations

The following table describes the terms and abbreviations used throughout this guide:

Term or Acronym	Description
ESSO-LM	Oracle Enterprise Single Sign-On Logon Manager
Agent	ESSO-LM client-side software
Console	ESSO-LM Administrative Console

Accessing ESSO-LM Documentation

We continually strive to keep ESSO-LM documentation accurate and up to date. For the latest version of this and other ESSO-LM documents, visit http://download.oracle.com/docs/cd/E15624_01/index.htm.

Contacting Oracle Support

To contact Oracle Support, visit the Oracle Support site at http://support.oracle.com.

Part 1: Understanding Web Form Detection and Response

This part explains the concepts necessary to understand how and why you should configure application templates to solve specific sign-on scenarios, as explained later in this guide. This part covers the following topics:

- Overview of a Sign-On Event
- Understanding Web Form Detection
- Understanding Form Response

Overview of a Sign-On Event

ESSO-LM can be configured to detect and respond to a wide range of sign-on events, such as logon, password change, and variations of thereof; support is provided for a diverse range of forms, fields, controls, and event flows.

In order to recognize Web forms, ESSO-LM monitors the running Web browser(s) and detects whenever a Web page has completed loading in the browser.

For each completely loaded page, ESSO-LM does the following:

1. **Detects the target Web page and form.** Whenever a new URL is accessed through the browser being monitored, ESSO-LM:
   a. Examines the page URL and compares it to the URLs stored in all available Web application templates.
   b. Loads the first template that matches the detected URL.
   c. Examines the page's Document Object Model (DOM) and identifies each field and control configured in the form definition.
   d. (Optional) Examines the page's source code and performs any additional matching configured in the form definition.
2. **Agent responds to the form and completes the logon.** When detection is complete and a positive match is found, the Agent follows the configuration stored in the template to determine how to interact with the fields and controls in the form. Typically, the Agent does the following:
   a. Retrieves the associated credentials from the user's store (if they exist) and injects them into the appropriate fields. (If the credentials don't exist, the Agent prompts the user to store them.)
   b. Performs the actions necessary to submit the credentials to the application for processing.
   c. (Optional) Detects any follow-up forms, such as password change success or failure messages, and performs the required action.

Understanding Web Form Detection

In order to communicate with the installed Web browser(s), ESSO-LM uses helper objects that hook into the browser(s) and provide ESSO-LM with a means of querying the browser for data, such as the page URL, DOM, and HTML source, as well as inject credentials into the target fields and submit them upon successful detection.

The helper objects run as the following background processes:

- For Microsoft Internet Explorer: `SSOBHO.EXE`
- For Mozilla Firefox: `SSOMOZHO.EXE`

Whenever the Web browser completes loading a page, the respective helper object notifies ESSO-LM of this event and a 500ms grace period timer starts. During this grace period, one of the following happens:

- If the current page is not refreshed or another page does not begin loading, ESSO-LM begins detection.
- If the current page is refreshed, or if another page begins loading, ESSO-LM restarts the timer and waits until the page has finished loading before beginning detection.

> **Note:** If the page is refreshed or another page starts loading after detection has begun, ESSO-LM aborts detection and waits for the page to finish loading, then starts the 500ms grace period timer and waits for it to elapse as described above.

Usually, the browser's status bar, located at the bottom of the browser window, indicates the page has completed loading by displaying an appropriate message, such as "Done" or "Finished."

Once detection begins, ESSO-LM examines the following characteristics of the page for comparison against available templates:

- URL
- Target fields and controls
- (Optional) Any additional matching criteria, such as text, HTML code, and element attributes, which have been configured in the template for the target form.

> **Note:** An *application template* is a set of configuration options that instruct the ESSO-LM Agent how to detect and respond to Web application forms in order to successfully complete the logon.

Keep in mind that in situations listed below you will have to configure your application as a Windows application instead of as a Web application:

- You are using Internet Explorer and the target form is implemented as an ActiveX control
- You are using Mozilla Firefox and the target form is displayed using the browser's built-in authentication dialog

See the guide *Template Configuration and Diagnostics for Windows Applications* for more information.

Supported Form Types

As of the release date of this document, ESSO-LM supports the following types of forms in Web applications:

- Logon
- Password change
- Password change success (a message confirming successful password change)
- Password change failure (a message indicating the new password was rejected)

A single template can contain definitions for the multiple forms that the Web application can display. For most applications, you need to only define the forms to which you want ESSO-LM to respond.

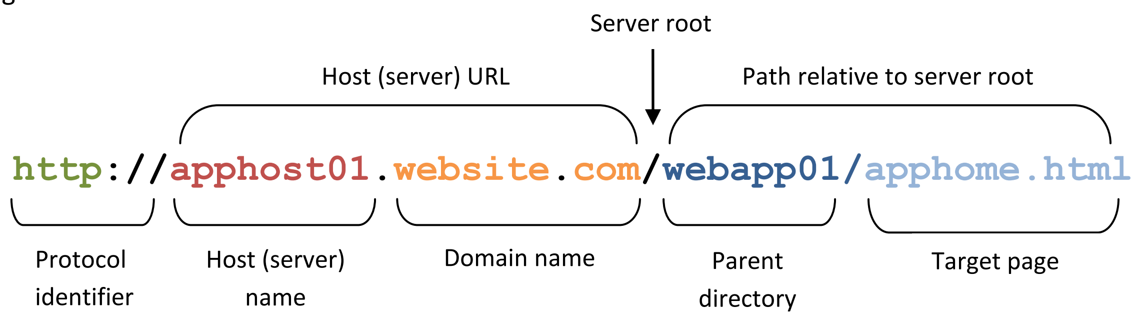
Note: Defining a form comprises providing unique identification criteria, specifying the action to take when the form is detected, and the specific way in which the action (e.g., injecting credentials) should be performed.

ESSO-LM can automatically populate the appropriate fields in a form with credentials retrieved from the user's credential store, as well as operate the form's controls to submit the credentials to the application for processing. Configuration options that instruct the Agent how to interact with the form's fields and controls are stored in the template.

Understanding URL Detection

The first criterion ESSO-LM considers during Web form detection is the Web address, or URL of the target page. This is usually the address you see in the browser's address bar after the page has finished loading, though in certain scenarios you might find that the actual URL you really want to specify in your form definition is the URL of an element of a page, such as the container or form element (these terms are explained in the next section) that actually contains the logon fields and controls.

A page URL is structured as follows:



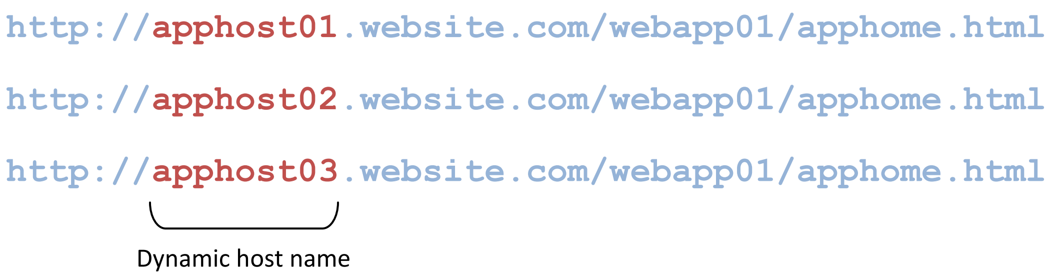
Depending on the way the Web application has been deployed, the URL can be static or partially dynamic. For example, a simple Web application might be deployed on a single Web server that has a static host name and exists as a single HTML page, as in the example shown above. In such case, you can configure the form definition to match against such a URL and enjoy a working template.

Many enterprise Web applications, on the other hand, are deployed on multiple servers that are load-balanced or clustered, and the individual instance URLs are masked behind a single entry URL.

For example, while the URL used to access the Web application might simply be

`http://webapp.website.com/webapp01/apphome.html`

the actual URLs to the individual load-balanced instances might be



`http://apphost01.website.com/webapp01/apphome.html`

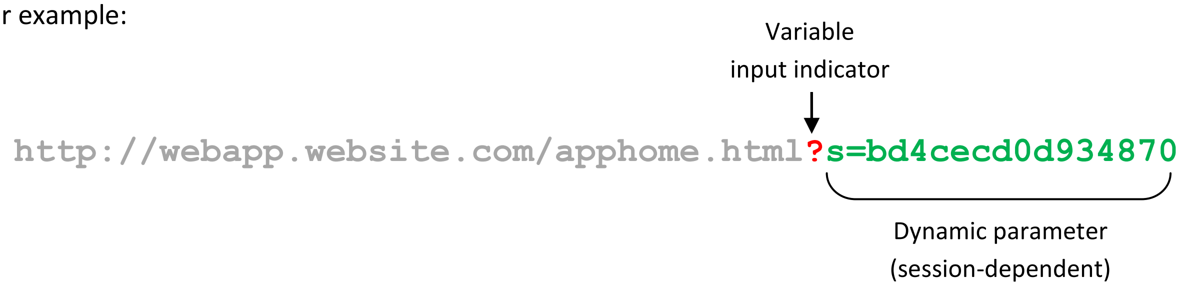
`http://apphost02.website.com/webapp01/apphome.html`

`http://apphost03.website.com/webapp01/apphome.html`

Dynamic host name

Most enterprise Web applications employ active scripting technologies, such as JavaScript or PHP, to pass data back and forth between the browser and the Web server and/or database. Such applications will employ one or more dynamic parameters, or variables, such as unique session IDs, in the page URL.

For example:

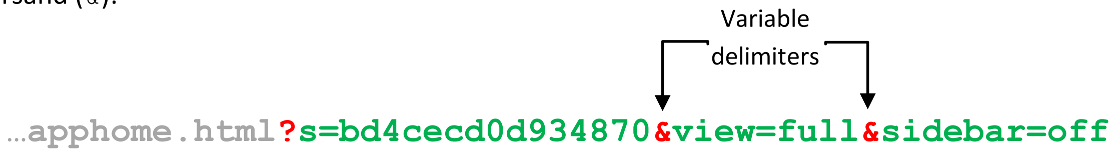


Variable input indicator

`http://webapp.website.com/apphome.html?s=bd4cecd0d934870`

Dynamic parameter (session-dependent)

Some applications stack URL variables by separating them with a variable delimiter symbol, usually an ampersand (&):



Variable delimiters

`…apphome.html?s=bd4cecd0d934870&view=full&sidebar=off`

In most enterprise Web application scenarios, you will find the combination of dynamic host names and dynamic parameters, and will need to account for both in order to successfully configure your form definition. ESSO-LM allows for matching against dynamic URLs via regular expressions, as well as a global Agent setting that determines the URL matching precision used by ESSO-LM for all Web applications.

Understanding URL Matching Precision

URL matching precision determines how many segments of the host URL, counting from right to left, within a page URL are considered during detection.

For example, at the default precision level of `2`, if the host name URL is application URL is

$$\texttt{http://}\overbrace{\underbrace{\texttt{webapp}}_{\text{Host (server) name}}\texttt{.}\underbrace{\texttt{website.com}}_{\text{Domain name}}}^{\text{Host (server) URL}}\texttt{/apphome.php}$$

then both the above host URL as well as the URLs of the individually-hosted instances of the application shown below will be positive matches for the template:

`http://apphost01.website.com/apphome.php`

`http://apphost02.website.com/apphome.php`

This is equivalent to masking the last segment of the host URL either via a wildcard:

`http://*.website.com/apphome.php`

or via regular expression:

`http://.*\.website\.com/apphome\.php`

> **Note:** Certain URLs might include Top Level Domains (TLD) that consist of two segments (e.g., `.co.uk` for England, `.co.au` for Australia, and so on) instead of the typical single-segment TLD (e.g., `.com`, `.net`, and so on). In such cases, you must consider the extra segment in the host URL when configuring URL matching precision and/or Web application templates.

If you increase URL matching precision to 3, then ESSO-LM will match against three consecutive segments of the host URL – in our example, it will match the entire host name exactly as it appears:

$$\texttt{http://}\underbrace{\texttt{webapp}}_{\text{Host (server) name}}\texttt{.}\underbrace{\texttt{website.com}}_{\text{Domain name}}\texttt{/apphome.php}$$

Variations in the server name will thus not be permitted and anything other than the exact host URL configured in the template will be rejected by ESSO-LM during detection. In our example, the URLs of the individual instances of the Web application shown above would not constitute positive matches.

Follow these guidelines when determining the optimal URL matching precision for your environment:

- If the URL matching precision is set too low, ESSO-LM may mistake one intranet application for another and respond with incorrect credentials.
- If URL matching precision is set too high, an application served through a distributed infrastructure with unique host names may be erroneously recognized as separate applications due to the varying host URL.
- Typically, set URL matching precision to 5 (the maximum value). This will ensure that ESSO-LM only responds when the URL of the application requesting logon exactly matches the URL stored in the template. The auto-recognize feature will have limited functionality.
- If you want to get the maximum benefit from ESSO-LM's auto-recognize feature for Web applications, leave URL matching precision at its default value of 2. However, response to intranet applications might be impaired.

ESSO-LM allows you to set the URL matching precision level globally via the Administrative Console. The option is located under **Global Agent Settings → End-User Experience →Response → Web Apps**. For more information on configuring the Agent and publishing your changes to end-user machines, see the *Best Practices* guide *Configuring the ESSO-LM Agent*.

Obtaining the URL of a Page, Element, or Pop-Up

In certain situations, the URL of a page or element within a page might not be readily obtainable, but is necessary to complete the form definition in the template. For example, you might need to obtain the URL for:

- An embedded form that loads from a different URL than the URL of the Web page
- A logon processing script that passes data back and forth between the browser and the Web server and/or database
- An image, or an embedded multimedia object
- A pop-up

In such cases, do one or more of the following to obtain the desired URL:

- Hit F11 to display the browser's address bar, if it is not visible
- Right-click within the object or form and select Properties; the Properties dialog might reveal the object's URL
- Examine the HTML source code of the page after it has completed loading and find the object in question
- Use a third-party tool, such as HTTPWatch or URL Getter to obtain the URL of the element, especially if the URL is partially dynamic. In some cases, viewing the source code of the page might be enough to obtain the complete URL of the object in question.

Understanding Field Detection

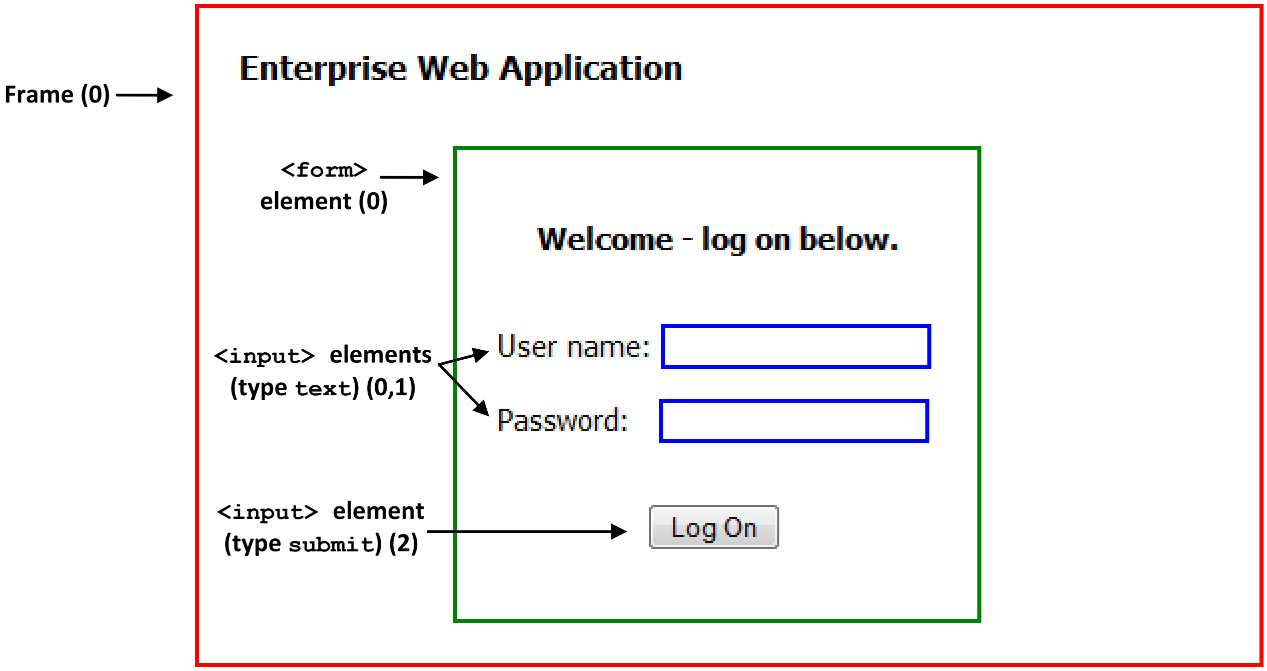
Once ESSO-LM has positively matched the detected URL to a template, it examines the page's Document Object Model (DOM) in order to detect the fields configured in the form definition.

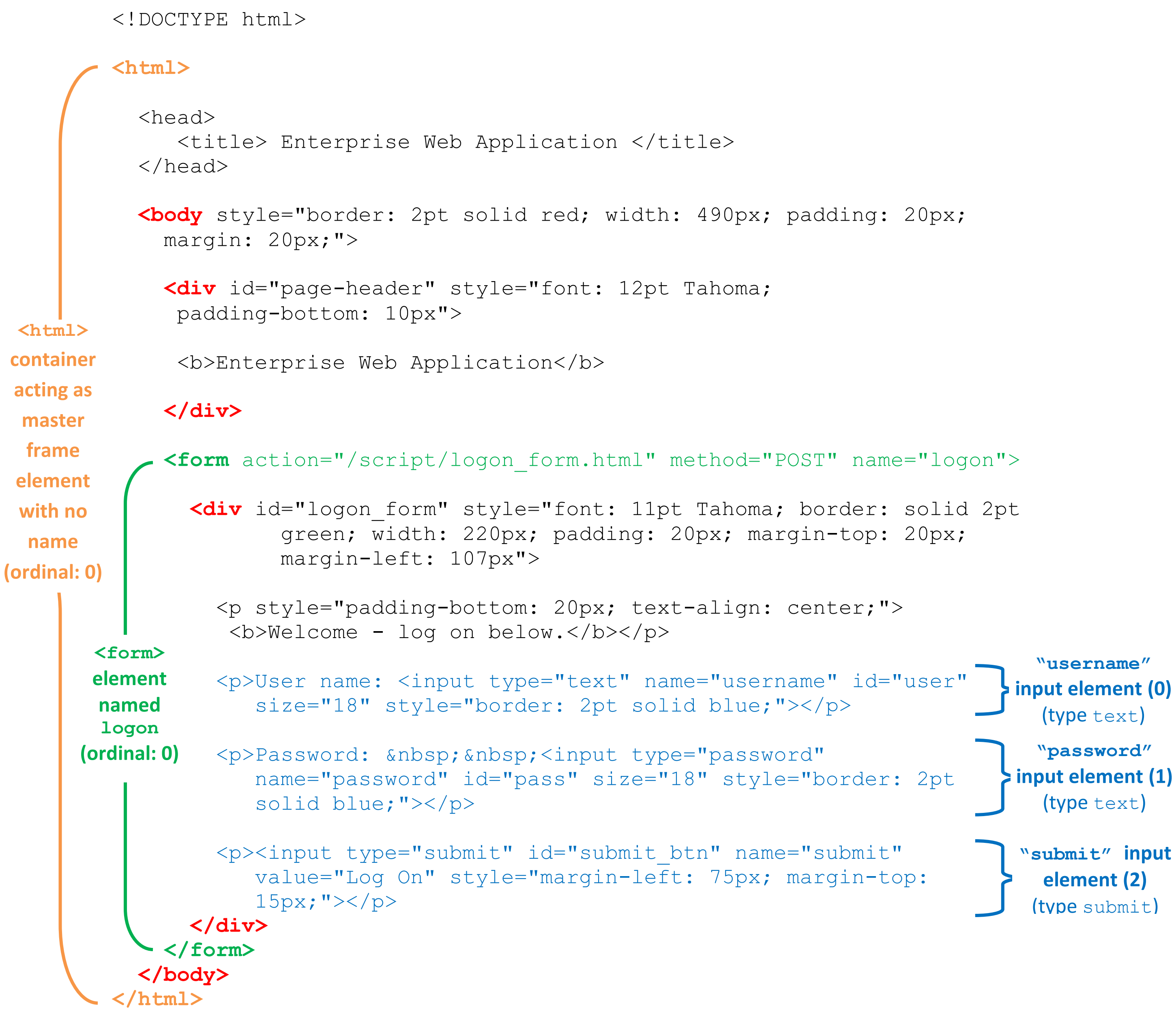
For each field and control, the Agent looks at the name or ordinal of:

- The parent frame element
- The parent form element
- The target field element

In order for a field to be successfully identified, all three values must match the values configured in the form definition in the template. The figure and corresponding code example below illustrate the structure of a simple Web page containing a logon form.

Note: The vast majority of today's Web pages no longer use frames to sub-divide their content. For detection purposes, ESSO-LM treats the `<html>` container as the master frame in the page, identifying it with the ordinal value of 0. In most cases, you will never have to change that value.


Frame (0)
Enterprise Web Application
<form> element (0)
Welcome - log on below.
<input> elements (type text) (0,1)
User name:
Password:
<input> element (type submit) (2)
Log On



```
<!DOCTYPE html>

<html>

  <head>
     <title> Enterprise Web Application </title>
  </head>

  <body style="border: 2pt solid red; width: 490px; padding: 20px;
    margin: 20px;">

    <div id="page-header" style="font: 12pt Tahoma;
     padding-bottom: 10px">

     <b>Enterprise Web Application</b>

    </div>

    <form action="/script/logon_form.html" method="POST" name="logon">

      <div id="logon_form" style="font: 11pt Tahoma; border: solid 2pt
             green; width: 220px; padding: 20px; margin-top: 20px;
             margin-left: 107px">

        <p style="padding-bottom: 20px; text-align: center;">
         <b>Welcome - log on below.</b></p>

        <p>User name: <input type="text" name="username" id="user"
           size="18" style="border: 2pt solid blue;"></p>

        <p>Password: &nbsp;&nbsp;<input type="password"
           name="password" id="pass" size="18" style="border: 2pt
           solid blue;"></p>

        <p><input type="submit" id="submit_btn" name="submit"
           value="Log On" style="margin-left: 75px; margin-top:
           15px;"></p>
      </div>
    </form>
  </body>
</html>
```

<html> container acting as master frame element with no name (ordinal: 0)
<form> element named logon (ordinal: 0)
"username" input element (0) (type text)
"password" input element (1) (type text)
"submit" input element (2) (type submit)

Note the hierarchical, nested structure of the code which corresponds to the parent-child relationship of the elements of the rendered page.

> **Note:** Each pair of opening and closing tags in the above example is an *element* - a piece of code that constitutes a functional entity within the page, for example a frame, a form, or a field.

When you configure a form definition in a template manually, you can identify the frame, form, and field elements to ESSO-LM by:

- Value of the element's `name` attribute, or, if not specified,
- Ordinal, an index number assigned by ESSO-LM to elements of each supported type according to the order in which they appear in the page's DOM.

Understanding HTML Elements

As mentioned in the previous section, an *element* in HTML is a piece of code that constitutes a functional entity within the page, for example a frame, a form, or a field. An element can be a single line of code - a single tag such as `<input>`, or consist of an opening and closing tag, such as `<div>…</div>` and everything in between those tags.

For example, a typical `<input>` element looks as follows:

```
<input type="text" name="username" id="user" size="18"
   style="border: 2pt solid blue;">
```

Note the code after the element declaration (`<input`) – these are the element's *attributes.* Each attribute has a name and value in the form:

```
attribute_name="attribute_value"
```

The straight double-quotes (inch-marks) are required for the attribute value to be recognized.

Understanding the Frame Element

A frame is a legacy container that allows the sectioning of a Web page by defining a frameset and specifying the size of each frame and the HTML document that it will display. The vast majority of Web sites today no longer employ this technique and use logical containers defined in the XHTML standard, such as `<div>` and `<table>` in tandem with CSS formatting to achieve sectioning. Therefore, in most scenarios, you will notice that ESSO-LM treats the `<html>` container as the master (and only) frame in the page, identifying it as frame 0.

When configuring your fields and detection matching rules, you will not have to change the frame identifier, unless your Web application is a legacy application and specifically requires this - for example, it uses a logon form that exists as a standalone HTML document and is displayed in a frame in the parent Web page. You can identify such a page by examining its DOM or source and looking for the frameset definition, denoted by the `<frameset>` tag. An example of a frameset definition looks as follows:

```
<FRAMESET cols="20%, 20%, 60%">
      <FRAME src="frame1.html">
      <FRAME src="frame2.html">
      <FRAME src="frame3.html">
</FRAMESET>
```

A frame can be identified by the `<frame>` tag:

```
<FRAME src="contents_of_frame3.html">
```

Understanding the Form Element

A `<form>` element defines an HTML form, which is a logical entity containing one or more *input fields* (described in the next section).

A <form> element definition references a script or another resource that accepts and processes the values entered into those fields and submitted through the browser. For example:

```
<form action="/script/logon_form.html" method="POST"
  name="logon">
```

In our example, the `logon_form.html` page handles the user authentication; when the user submits the form, the `logon_form.html` receives the values entered into the form's input fields and executes the appropriate authentication logic.

Understanding the Input Elements Supported by ESSO-LM

Typically, an HTML form field is defined as an input element of a specific class, denoted by the tag and `type` attribute used to define the element. When configuring a form definition, ESSO-LM recognizes the following element types:

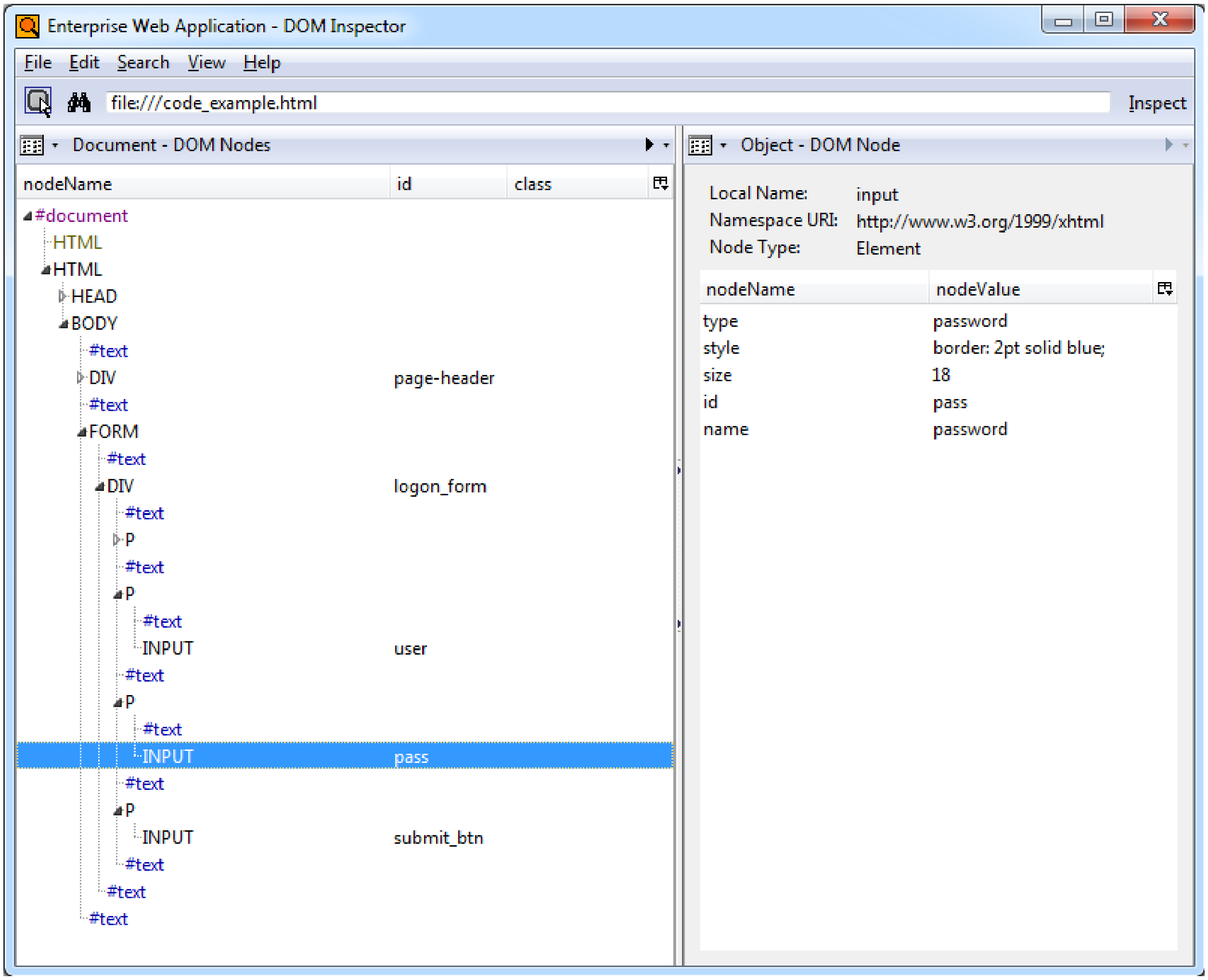
ESSO-LM Field Type	Input Element Types Accepted by the ESSO-LM Field Type
• **User name/ID** • **Password** • **Third Field** • **Fourth Field**	• **Text** – a text field identified by the `<input>` tag of `type="text"`. Text entered into this field is unmasked. • **Password** – a text field identified by the `<input>` tag of `type="password"`. Text entered into this field is masked. • **Select-one** – a list field identified by the `<select>` tag that permits the selection of a single list item as its value. • **Select-multiple** – a list field identified by the `<select multiple … >` tag that permits the selection of one or more list items as its value.
• **Submit**	• **Submit** – a button control identified by the <input> tag of `type="submit"`. Executes the code specified in the `action` attribute of the `<form>` tag (usually, calls an HTML document or script that contains the application's logon logic.) • **Button** - a button control identified by the <input> tag of `type="button"`. Functionally identical to the `submit` type `<input>` element, except allows a greater degree of styling. • **Image** –a hyperlinked image identified by the <input> tag of `type="image"` that executes the referenced link when clicked. • **Anchor** – the most common manifestation of a text hyperlink (by default, appears as blue underlined text in a page). Identified by the `<A HREF="…">` tag.

Recommended Page Inspection Tools

When provisioning Web applications, Oracle highly recommends obtaining a page inspection tool that will display the page's DOM (and optionally, source code) in an easily navigable and searchable manner, allowing for much easier and faster understanding of the page's structure.

DOM Inspector for Mozilla Firefox

An example of such a tool is the DOM Inspector add-on for Mozilla Firefox. The DOM of our example page would appear in DOM Inspector as follows:


Enterprise Web Application - DOM Inspector
File
Edit
Search
View
Help
file:///code_example.html
Inspect
Document - DOM Nodes
Object - DOM Node

```
nodeName                          id              class
◢#document
  HTML
 ◢HTML
   ▷HEAD
   ◢BODY
      #text
     ▷DIV                          page-header
      #text
     ◢FORM
        #text
       ◢DIV                        logon_form
          #text
         ▷P
          #text
         ◢P
            #text
            INPUT                  user
          #text
         ◢P
            #text
            INPUT                  pass
          #text
         ◢P
            INPUT                  submit_btn
          #text
        #text
      #text
```

Local Name: input
Namespace URI: http://www.w3.org/1999/xhtml
Node Type: Element

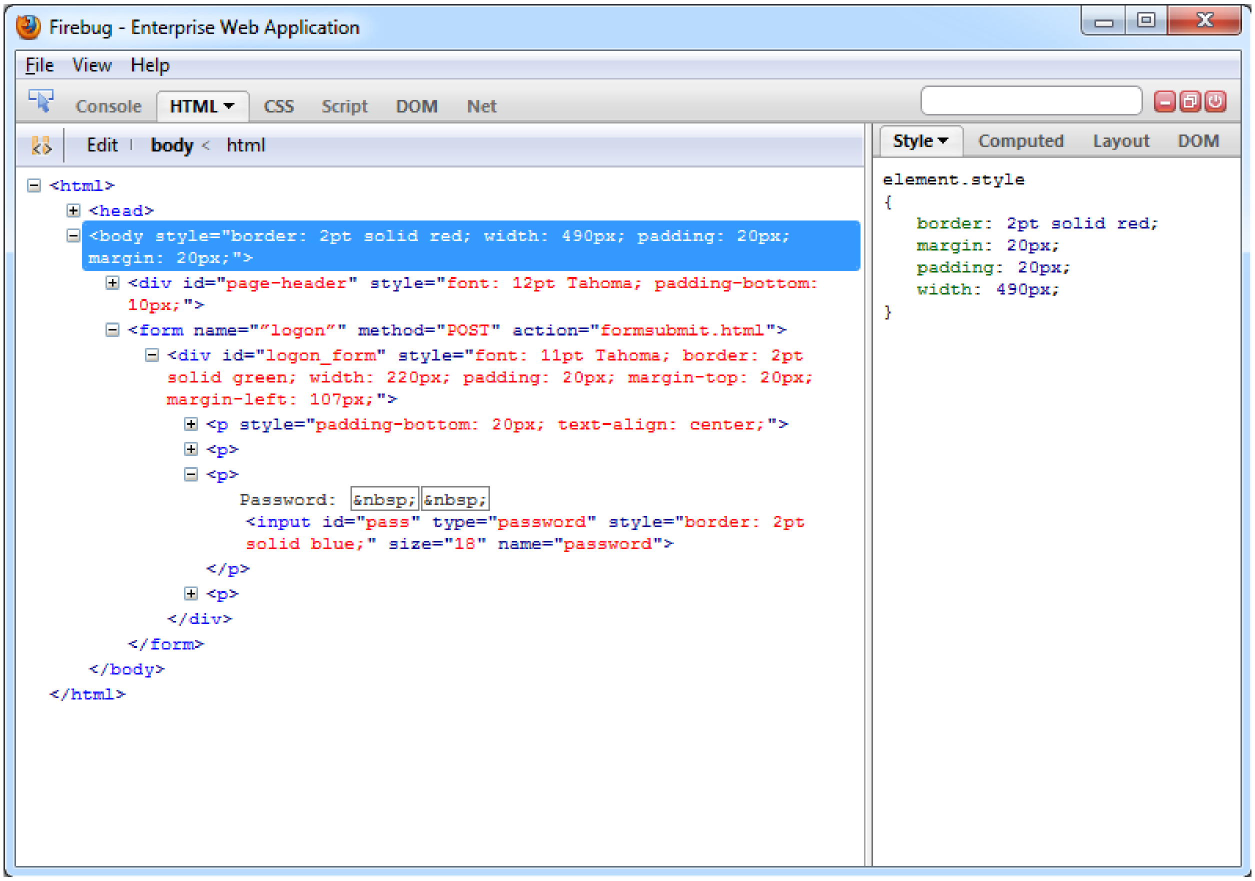
nodeName	nodeValue
type	password
style	border: 2pt solid blue;
size	18
id	pass
name	password

The hierarchical structure of the page is displayed in a navigable tree in the left pane and clicking an element highlights it in the browser window and displays the element attributes and their values in the right pane, making it easy to identify and understand target page elements without sifting through the page's source code. This tool is especially useful for large, complex pages.

The DOM Inspector is available from the Mozilla Firefox Add-Ons site at http://addons.mozilla.org.

Firebug for Mozilla Firefox

A more advanced and flexible tool similar to DOM Inspector is the Firebug add-on for Mozilla Firefox. Firebug sports all of the features of DOM inspector and adds full source code display in the DOM tree including visual highlighting of each code element on mouse-over. Users can also modify the code and see the changes in the browser window. While Firebug is aimed mainly at Web developers, it can prove itself invaluable during analysis of complex Web pages.


Firebug - Enterprise Web Application
File View Help
Console HTML ▾ CSS Script DOM Net
Edit body < html
Style ▾ Computed Layout DOM

```
⊟ <html>
    ⊞ <head>
    ⊟ <body style="border: 2pt solid red; width: 490px; padding: 20px;
      margin: 20px;">
        ⊞ <div id="page-header" style="font: 12pt Tahoma; padding-bottom:
          10px;">
        ⊟ <form name=""logon"" method="POST" action="formsubmit.html">
            ⊟ <div id="logon_form" style="font: 11pt Tahoma; border: 2pt
              solid green; width: 220px; padding: 20px; margin-top: 20px;
              margin-left: 107px;">
                ⊞ <p style="padding-bottom: 20px; text-align: center;">
                ⊞ <p>
                ⊟ <p>
                      Password: &nbsp;&nbsp;
                      <input id="pass" type="password" style="border: 2pt
                      solid blue;" size="18" name="password">
                  </p>
                ⊞ <p>
              </div>
          </form>
      </body>
  </html>
```

```
element.style
{
    border: 2pt solid red;
    margin: 20px;
    padding: 20px;
    width: 490px;
}
```

Firebug is available from the Mozilla Firefox Add-Ons site at http://addons.mozilla.org.

Understanding Detection Matching

If the URL and the elements discussed above are not enough to uniquely identify a logon form within a page, you may specify additional detection criteria to further narrow down the detection scope via *matching*. The term “matching” refers to the comparison of the values of certain parameters, such as element names or attribute values, to the values stored in the application’s template.

For example, if a Web application is down for maintenance, attempting logon may result in the logon form being redisplayed with an error message indicating the reason of the logon failure. To prevent ESSO-LM from repeatedly attempting logon in such a scenario, you would define two logon forms in the template and use text matching in one of the definitions to match against the error message text and instruct ESSO-LM to ignore the logon form displaying the error.

> **Note:** This section explains the principles behind the matching mechanism necessary to successfully configure matching for a Web application. For actual instructions in configuring matching via the Console, see Configuring Additional Field Detection Criteria.

ESSO-LM allows you to match against:

- Text displayed on the page
- HTML code within the page’s source code
- Names and values of attributes of page elements

To illustrate the above matching types, we will use the example from pp. 15-16. The example displays a header, “Enterprise Web Application,” which is implemented as a `<div>` container to allow for easy inline-styling of that specific piece of text:

```
<div id="page-header" style="font: 12pt Tahoma; padding-bottom: 10px">

        <b>Enterprise Web Application</b>

</div>
```

Detection Matching vs. Offset Matching

ESSO-LM offers two kinds of matching, detection, and offset. In almost all scenarios, you will always use detection matching, and never offset matching. The latter is useful for legacy pages using framesets in which frames are not identified by a name attribute. In such case, ESSO-LM will assign ordinals to the frames in the order the frames appear in the page’s DOM. To match on elements within a specific frame in such a frameset, you would specify its ordinal as the value for the **Offset Start** parameter in the **Matching** tab of the form definition dialog.

Matching Against Text on the Page

The most commonly used matching mode, you can match against the header text itself, regardless of its formatting. For example, to match against "Enterprise Web Application" by configuring the matching rule as follows:

- **Tag:** `div`
- **Criteria:** Text
- **Value:** `Enterprise Web Application`
- **Match whole value:** Yes
- **Use regular expression:** No

In this configuration, the following will be matched:

```
<div id="page-header" style="font: 12pt Tahoma; padding-bottom: 10px">

        <b>Enterprise Web Application</b>

</div>
```

Matching Against HTML Code in the Page's Source

The HTML matching mode allows you to match against any piece of code within the page's source code. Most common use of this mode is to match against rendered text including its markup, but can also be used for matching a specific piece of code that is unique to the page.

> **Note:** Due to the nature of the HTML specification, ESSO-LM cannot match on a `<span>` tag.

For example, to match against "**Enterprise Web Application**" but not "Enterprise Web Application", you would set up your matching rule as follows:

- **Tag:** `div`
- **Criteria:** HTML
- **Value:** `<b>Enterprise Web Application></b>`
- **Match whole value:** Yes
- **Use regular expression:** No

This configuration would result with the following match:

```
<div id="page-header" style="font: 12pt Tahoma; padding-bottom: 10px">

        <b>Enterprise Web Application</b>

</div>
```

> **Note:** The **Match Whole Value** option forces ESSO-LM to match against the complete string entered in the **Value** field, and is enabled by default. To allow for partial matches on the page, (e.g., `<b>Enterprise`), disable this option.

To match against the entire opening tag of the above `<div>` container, you would specify it as the criterion value and specifying its parent element as the tag:

- **Tag:** `body`
- **Criteria:** HTML
- **Value:** `<div id="page-header"…`
- **Match whole value:** Yes
- **Use regular expression:** No

This would result in the following match:

```
<div id="page-header" style="font: 12pt Tahoma; padding-bottom: 10px">

        <b>Enterprise Web Application</b>

</div>
```

Matching Against Names and Values of Element Attributes

You can also match against a specific attribute or its value within an element. For example, you can match against the value of the id attribute of the `<div>` element:

- **Tag:** `div`
- **Criteria:** Attribute
- **Attribute name:** `id`
- **Value:** `page-header`
- **Match whole value:** Yes
- **Use regular expression:** No

This would result in the following match:

```
<div id="page-header" style="font: 12pt Tahoma; padding-bottom: 10px">

        <b>Enterprise Web Application</b>

</div>
```

Leaving the **Value** field blank would, however, result in a match on the attribute name:

```
<div id="page-header" style="font: 12pt Tahoma; padding-bottom: 10px">

        <b>Enterprise Web Application</b>

</div>
```

Constructing Complex Rule Chains Using Logic Operators

ESSO-LM allows you to chain individual matching rules using Boolean logic operators to create complex rule chains that only result in a positive match in a very specific set of circumstances.

For example, you may want to create a match rule chain for a password change form that displays a success message when the password has successfully been changed, while still displaying the password change fields and controls. In such case you would want ESSO-LM to ignore the form once the success message is displayed so that ESSO-LM does not enter a password change loop, while retaining proper response to the standard password change form.

To accomplish this, you would create the following rules:

- A **NOT** match rule for the success message text, which will cause ESSO-LM to ignore the form, and
- An **AND** match rule for any text on the form that will result in a positive match and cause ESSO-LM to respond when the success message is not present.

Based on our example, you would configure the above rules as follows:

Rule 1

- **Tag:** `html`
- **Criteria:** Text
- **Value:** `Logon error`
- **Match whole value:** Yes
- **Use regular expression:** No
- **Operation:** NOT

Rule 2

- **Tag:** `html`
- **Criteria:** Text
- **Value:** `Enterprise Web Application`
- **Match whole value:** Yes
- **Use regular expression:** No
- **Operation:** AND

Note: When defining a **NOT** match rule, you must always follow it with an **AND** rule that will result in a positive match. Otherwise, ESSO-LM will not respond to the form.

Other common matching rule chaining examples are shown below. To get a positive match on:

- **Rule 1 AND Rule 2** – configure both rules with the AND operator, in that order.
- **Rule 1 OR Rule 2** – configure Rule 1 with the AND operator and Rule 2 with the OR operator, in that order.
- **Rule 1 AND Rule 2 but NOT Rule 3** – same as above. Place the NOT rule first in the chain, followed by the two AND rules.
- **Rule 1 OR Rule 2 but NOT Rule 3** – configure Rule 1 as an AND rule, Rule 2 as an OR rule, and Rule 3 as a NOT rule, then place the NOT rule first in the chain, followed by the AND rule and the OR rule.

Using Wildcards to Configure Detection Criteria

Wildcards, such as `?` (single character) and `*` (any combination of characters, excluding space), are the most commonly used methods of accounting for dynamic detection criteria. Use the **Wildcard** option when configuring a criterion in a form definition and follow the guidelines below:

- `Je???e` will match both `Jeanne` and `Jessie`. It will not, however, match `Jeanine`, because neither the length of the string nor character order match.
- `Je*e` will match against all words that begin with `Je` and end with `e`; all three examples above are matches in this case.
- `webapp*.company.com` will match against all URLs that contain a host name that starts with `webapp`. This technique is useful for matching against the URLs of applications that are load-balanced and masked via a common URL and load from a different physical host each time they are accessed, causing the URL to vary.

Using Regular Expressions to Configure Detection Criteria

You can also use regular expressions supported by the Microsoft .NET Framework to formulate the text pattern that ESSO-LM should recognize as a match for the criteria discussed earlier in this guide. (Use the **Regular expression** option when configuring a criterion in a form definition.) For your reference, the most commonly used operators and meta-characters are listed below:

<table>
<tr><th>Expression</th><th>Description</th></tr>
<tr><td>[ ]</td><td>Indicates a character class that matches any character inside the brackets.
Example: [abc] matches "a," "b," and "c."</td></tr>
<tr><td>( )</td><td>Indicates a character grouping operator.
Example: (\d+,)*\d+ matches a list of numbers separated by commas, such as "1" or "1,23,456".</td></tr>
<tr><td>{ }</td><td>Indicates a match class.</td></tr>
<tr><td>|</td><td>Separates two expressions, exactly one of which matches.
Example: T|the matches "The" or "the".</td></tr>
<tr><td>.</td><td>Matches any single character.</td></tr>
<tr><td>^</td><td>If ^ occurs at the start of a character class, it negates the character class. A negated character class matches any character *except* those inside the brackets.
Example: [^abc] matches all characters except "a", "b", and "c".
If ^ is at the beginning of the regular expression, it matches the beginning of the input.
Example: ^[abc] will only match input that begins with "a," "b," or "c".</td></tr>
<tr><td>$</td><td>At the end of a regular expression, $ matches the end of the input.
Example: [0-9]$ matches a digit at the end of the input.</td></tr>
<tr><td>-</td><td>In a character class, a hyphen indicates a range of characters.
Example: [0-9] matches any of the digits "0" through "9."</td></tr>
<tr><td>!</td><td>Negates the expression that follows.</td></tr>
<tr><td>?</td><td>Indicates that the preceding expression is optional: it matches once or not at all.
Example: [0-9][0-9]? matches "2" and "12".</td></tr>
</table>

<table>
<tr><th>Expression</th><th>Description</th></tr>
<tr><td>+</td><td>Indicates that the preceding expression matches one or more times.

Example: [0-9]+ matches "1," "13," "666," and so on.</td></tr>
<tr><td>*</td><td>Indicates that the preceding expression matches zero or more times.</td></tr>
<tr><td>??,+?,*?</td><td>“Non-greedy” versions of ?, +, and *. These match as little as possible, unlike the greedy versions that match as much as possible.

Example: Given the input <abc><def>, <.*?> matches <abc> while <.*> matches <abc><def>.</td></tr>
<tr><td>\</td><td>Escape character that forces the next character to be insterpreted literally.

Example: [0-9]+ matches one or more digits, but [0-9]\+ matches a digit followed by a plus character).

If \ is followed by a number *n*, it matches the *n*th match group (starting from 0).

Example: <{.*?}>.*?</\0> matches <head>Contents</head>.

The **** is also used for abbreviations, the most common of which are shown below:

<table>
<tr><th>Abbreviation</th><th>Description</th></tr>
<tr><td>\a</td><td>Any alphanumeric character. Substitutes for: a-z, A-Z, 0-9</td></tr>
<tr><td>\b</td><td>White (blank) space. Substitutes for : \\t</td></tr>
<tr><td>\c</td><td>Any alphabetic character. Substitutes for: a-z, A-Z</td></tr>
<tr><td></td><td>Any decimal digit. Substitutes for: 0-9</td></tr>
<tr><td>\d \h</td><td>Any hexadecimal digit. Substitutes for: 0-9, a-f, A-F</td></tr>
<tr><td>\n</td><td>New line. Substitutes for: \r|\r?\n</td></tr>
<tr><td>\q</td><td>A quoted string.

Substitutes for: \"[^\"]*\"|\'[^\']*\'</td></tr>
<tr><td>\z</td><td>An integer. Substitutes for: 0-9+</td></tr>
</table>
</td></tr>
</table>

Understanding Form Response

Once ESSO-LM has successfully identified the Web form and its input fields, it uses the appropriate helper object to programmatically inject the appropriate credentials into the form fields and engage the submit control. Each helper object utilizes the respective browser’s API to accomplish the above tasks.

Part 2: Configuring and Testing Forms

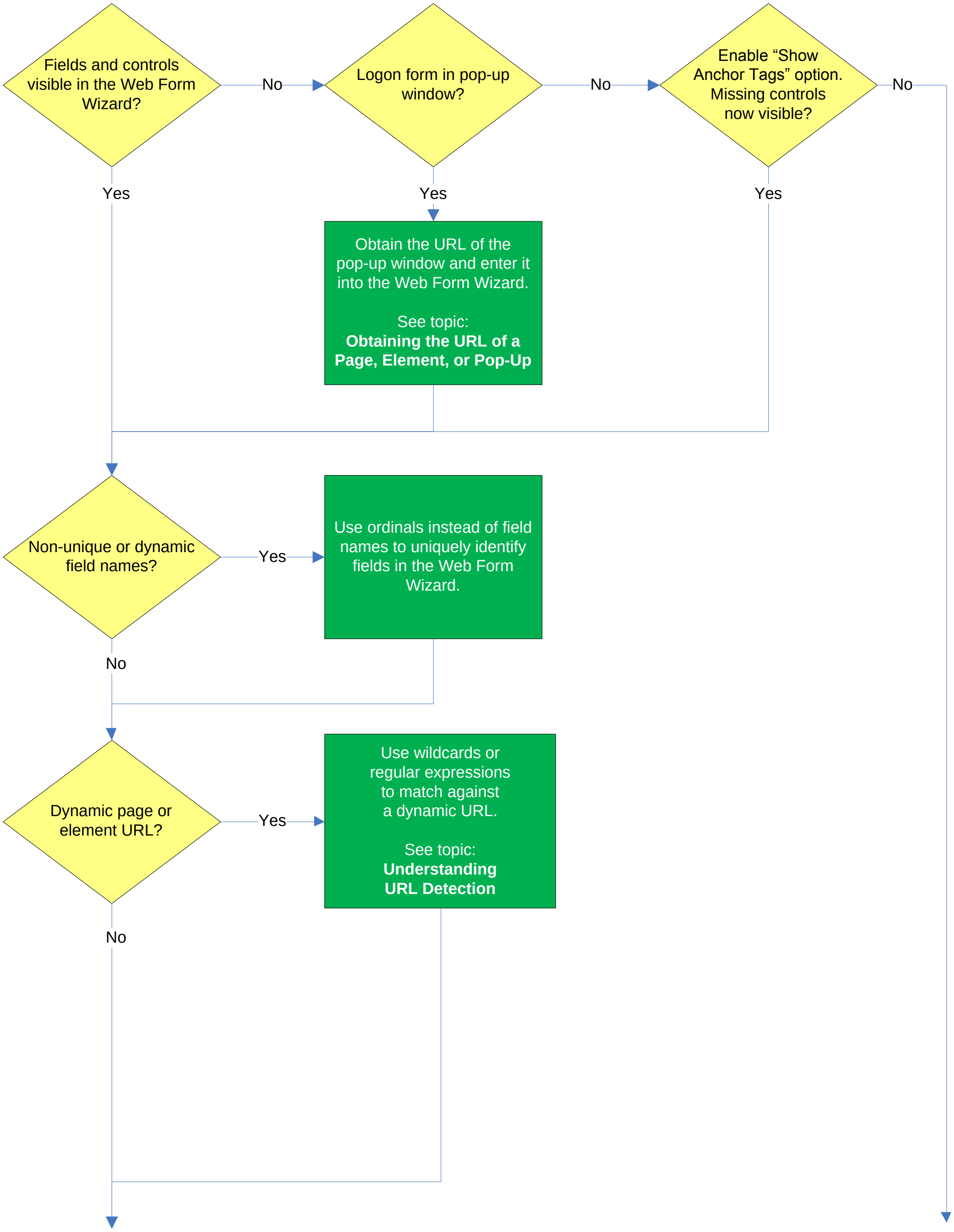
This part describes the recommended best-practice procedures for configuring and testing Windows application forms. It covers the following topics:

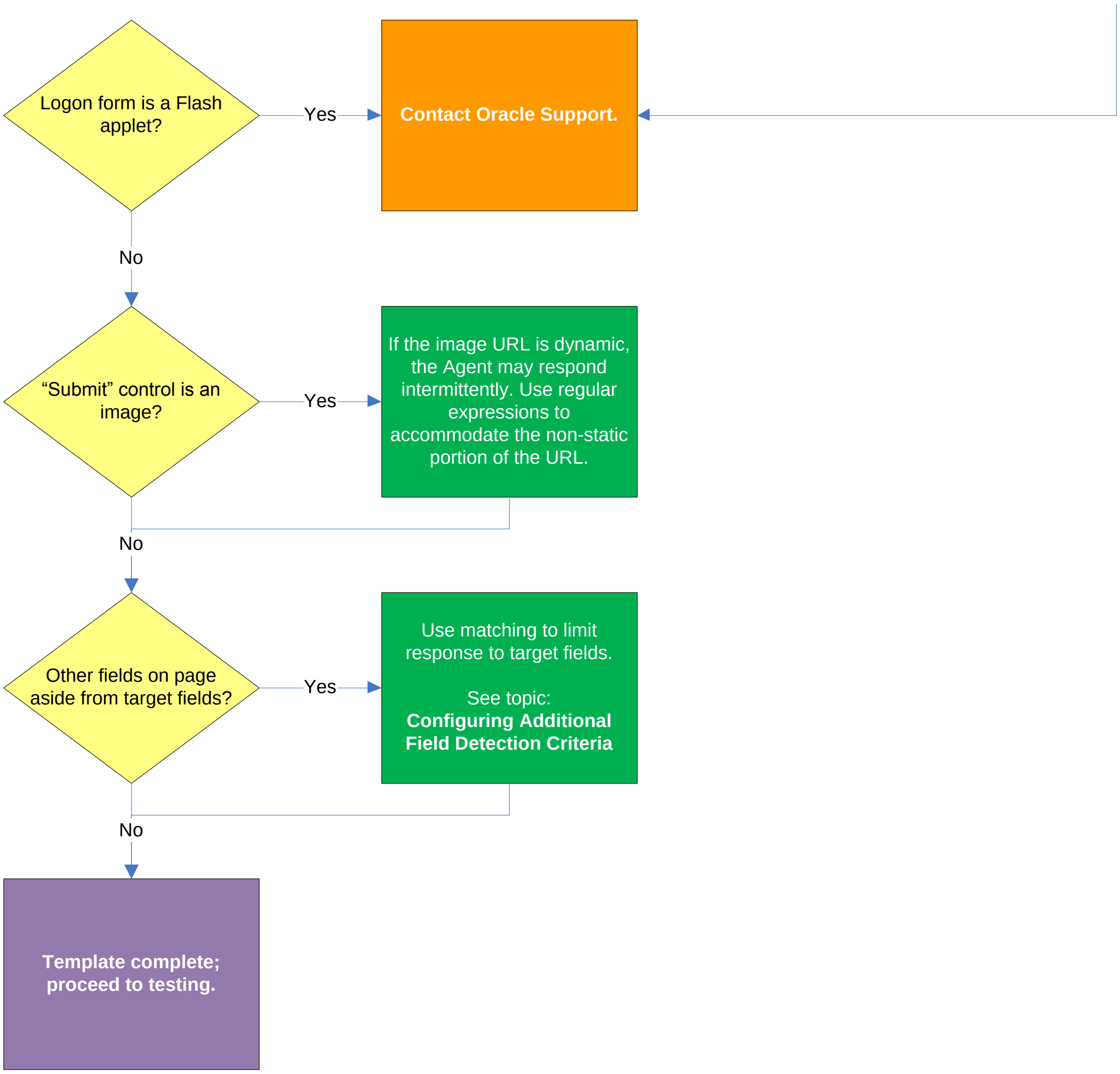
- Determining the Optimal Configuration for a Form
- Configuring a Form
- Testing the Configuration of a Form
- Publishing a Template to the Repository

Determining the Optimal Configuration for a Form

When configuring a form, use the information in this section to determine its optimal configuration based on the requirements and features of the target application.

Determining the Optimal Configuration for a Logon Form


Fields and controls visible in the Web Form Wizard?
No
Logon form in pop-up window?
No
Enable “Show Anchor Tags” option. Missing controls now visible?
No
Yes
Yes
Yes
Obtain the URL of the pop-up window and enter it into the Web Form Wizard.
See topic:
Obtaining the URL of a Page, Element, or Pop-Up
Non-unique or dynamic field names?
Yes
Use ordinals instead of field names to uniquely identify fields in the Web Form Wizard.
No
Dynamic page or element URL?
Yes
Use wildcards or regular expressions to match against a dynamic URL.
See topic:
Understanding URL Detection
No


Logon form is a Flash applet?
Yes
Contact Oracle Support.
No
“Submit” control is an image?
Yes
If the image URL is dynamic, the Agent may respond intermittently. Use regular expressions to accommodate the non-static portion of the URL.
No
Other fields on page aside from target fields?
Yes
Use matching to limit response to target fields.
See topic: Configuring Additional Field Detection Criteria
No
Template complete; proceed to testing.

Target fields and controls appear in the Web Form Wizard?

In most situations, the Web Form Wizard successfully detects and displays the credential fields and controls present in the target form. If some or all fields or controls are missing, do one of the following:

- If no fields or controls are visible in the Form Wizard, the logon form has a separate URL or is in a pop-up. Obtain this URL as described in Obtaining the URL of a Page, Element, or Pop-Up and supply it to the wizard instead of the page's main URL.
- If the submit controls are not recognized, enable the "Show Anchor Tags" option to reveal controls implemented as anchor (i.e., `<A HREF=`) tags.
- If fields are still not visible, see Contacting Oracle Support to obtain further assistance.

Non-unique or dynamic field names?

If the names of the target field elements are non-unique or dynamic, use ordinals instead of element names to identify the fields to ESSO-LM. Examine the page's DOM and/or source code to determine whether this is the case. For more information, see Understanding Field Detection.

Dynamic page or element URL?

If the URL of the page or form element is dynamic, you will need to use wildcards or regular expressions to limit Agent response to the desired URL or range of URLs. For more information, see Understanding URL Detection.

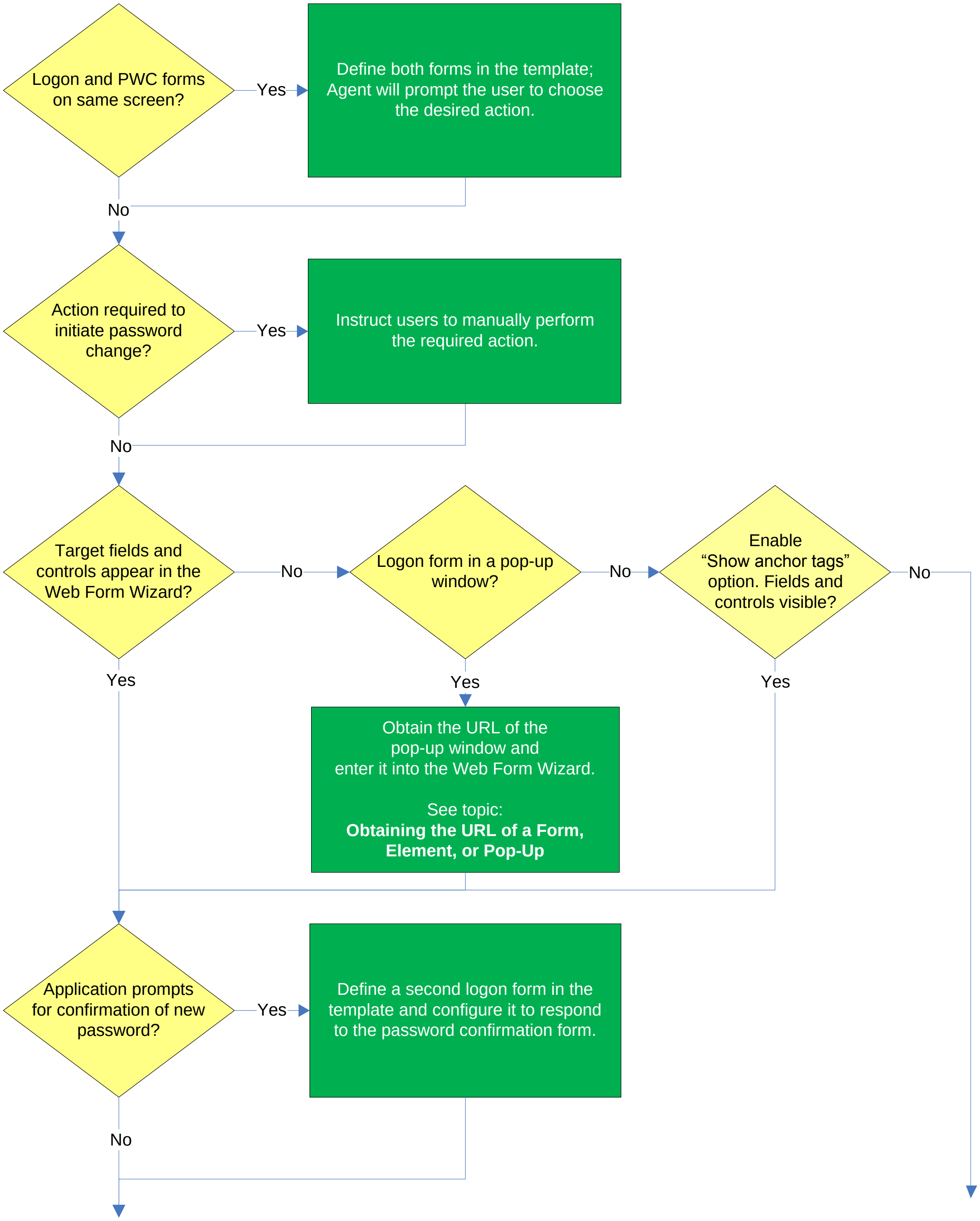
"Submit" control is an image?

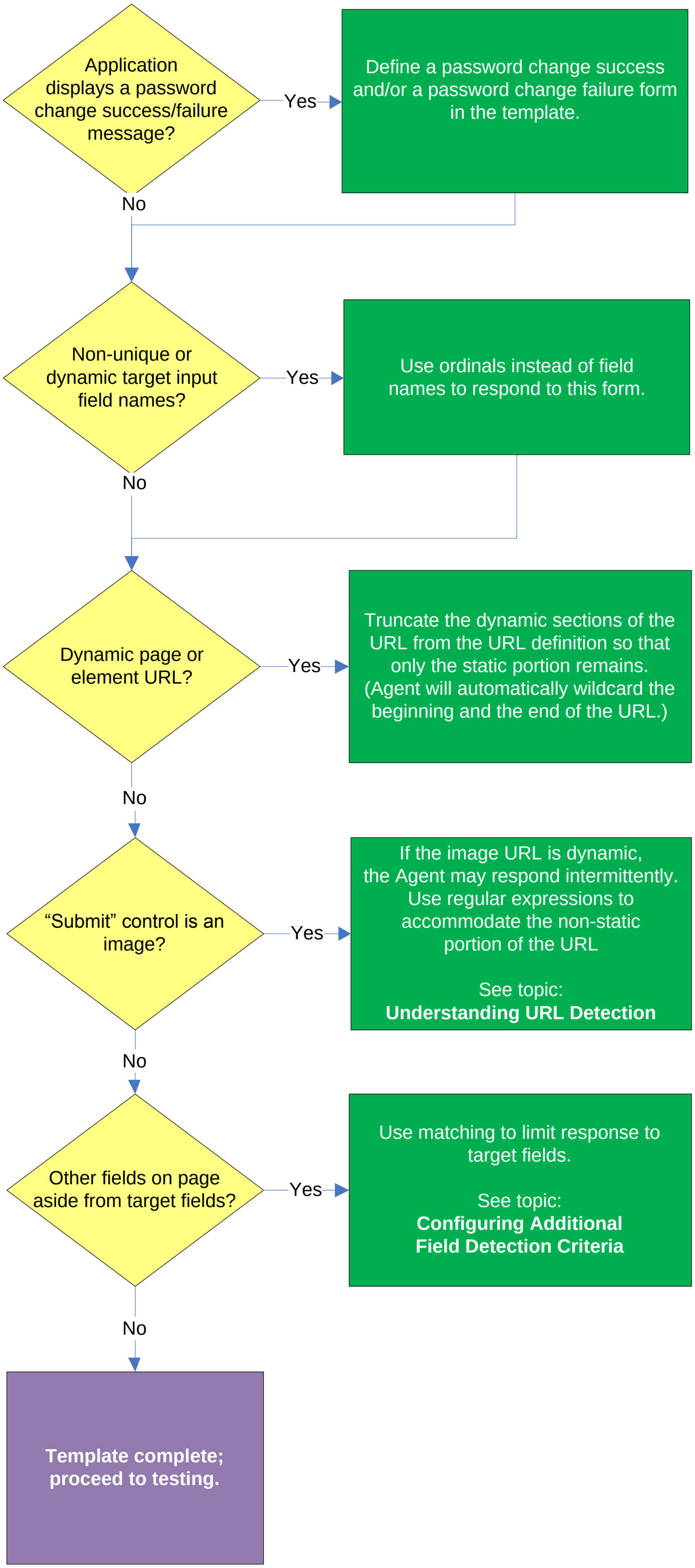
If the "submit" control is an image whose URL is dynamic (i.e., the image is hosted in a distributed environment and the host name portion of its URL is not static), you must use wildcards or regular expressions to specify the desired range of URLs as valid for the control; otherwise, ESSO-LM will stop detecting the control as soon as the control URL deviates from the original static value used for configuring the template. For more information, see Understanding URL Detection.

Other fields on the page besides target fields?

If there are other input fields and controls on the page aside from the target fields, use matching to limit Agent response to the desired fields. For more information, see Configuring Additional Field Detection Criteria.

Determining the Optimal Configuration for a Password Change Form


Logon and PWC forms on same screen?
Yes
Define both forms in the template; Agent will prompt the user to choose the desired action.
No
Action required to initiate password change?
Yes
Instruct users to manually perform the required action.
No
Target fields and controls appear in the Web Form Wizard?
No
Logon form in a pop-up window?
No
Enable "Show anchor tags" option. Fields and controls visible?
No
Yes
Yes
Obtain the URL of the pop-up window and enter it into the Web Form Wizard.
See topic:
Obtaining the URL of a Form, Element, or Pop-Up
Yes
Application prompts for confirmation of new password?
Yes
Define a second logon form in the template and configure it to respond to the password confirmation form.
No



Application displays a password change success/failure message?
Yes
Define a password change success and/or a password change failure form in the template.
No
Non-unique or dynamic target input field names?
Yes
Use ordinals instead of field names to respond to this form.
No
Dynamic page or element URL?
Yes
Truncate the dynamic sections of the URL from the URL definition so that only the static portion remains. (Agent will automatically wildcard the beginning and the end of the URL.)
No
"Submit" control is an image?
Yes
If the image URL is dynamic, the Agent may respond intermittently. Use regular expressions to accommodate the non-static portion of the URL
See topic:
Understanding URL Detection
No
Other fields on page aside from target fields?
Yes
Use matching to limit response to target fields.
See topic:
Configuring Additional Field Detection Criteria
No
Template complete; proceed to testing.

Logon and password change forms on the same screen?

Some applications might present password change forms that also contain logon-related fields or controls, such as a user name field or a **Change Password** button, which invokes the application's password change (PWC) form. If the **Auto Submit** feature is enabled and the Agent responds to such application, the user is logged in automatically without being given the option to change the password.

In order to allow the user to select the desired course of action, define the logon and password change forms in the template**.** The Agent will prompt the user for the desired course of action (logon or password change) when it responds to an application with consolidated logon and password change forms.

> **Note:** You also have the option to configure a grace period for the "action chooser" feature, during which the Agent will automatically assume that logon is the preferred action and log the user on without prompting to choose the desired action. This option, named **Action Chooser Grace Period**, is available in the **Miscellaneous** tab in the application template.

Action required to initiate password change?

If the form requires an action to initiate password change, such as selecting a checkbox, due to the nature of Web applications, ESSO-LM may not be able to complete that action programmatically; Oracle recommends instructing users to perform the required action manually, then allowing ESSO-LM to complete the password change.

Target fields and controls appear in the Web Form Wizard?

In most situations, the Web Form Wizard successfully detects and displays the credential fields and controls present in the target form. If some or all fields or controls are missing, do one of the following:

- If no fields or controls are visible in the Form Wizard, the logon form has a separate URL or is in a pop-up. Obtain this URL as described in Obtaining the URL of a Page, Element, or Pop-Up and supply it to the wizard instead of the page's main URL.
- If the submit controls are not recognized, enable the **Show Anchor Tags** option to reveal controls implemented as anchor (i.e., `<A HREF=`) tags.
- If fields are still not visible, see Contacting Oracle Support to obtain further assistance.

Application requires confirmation of new password?

Some applications prompt the user to confirm the new password when performing a password change. If the target application displays such a form, define a second logon form in the template and configure it to respond to the password confirmation form.

Application displays a password change success and/or failure dialogs?

Some application display a message indicating whether password change was a success or a failure. In such cases, define a password change success or a password change failure form in the application template.

Non-unique or dynamic field names?

If the names of the target field elements are non-unique or dynamic, use ordinals instead of element names to identify the fields to ESSO-LM. Examine the page's DOM and/or source code to determine whether this is the case. For more information, see Understanding Field Detection.

Dynamic page or element URL?

If the URL of the page or form element is dynamic, you will need to use wildcards or regular expressions to limit Agent response to the desired URL or range of URLs. For more information, see Understanding URL Detection.

"Submit" control is an image?

If the "submit" control is an image whose URL is dynamic (i.e., the image is hosted in a distributed environment and the host name portion of its URL is not static), you must use wildcards or regular expressions to specify the desired range of URLs as valid for the control; otherwise, ESSO-LM will stop detecting the control as soon as the control URL deviates from the original static value used for configuring the template. For more information, see Understanding URL Detection.

Other fields on the page besides target fields?

If there are other input fields and controls on the page aside from the target fields, use matching to limit response to the desired fields. For more information, see Configuring Additional Field Detection Criteria.

Configuring a Form

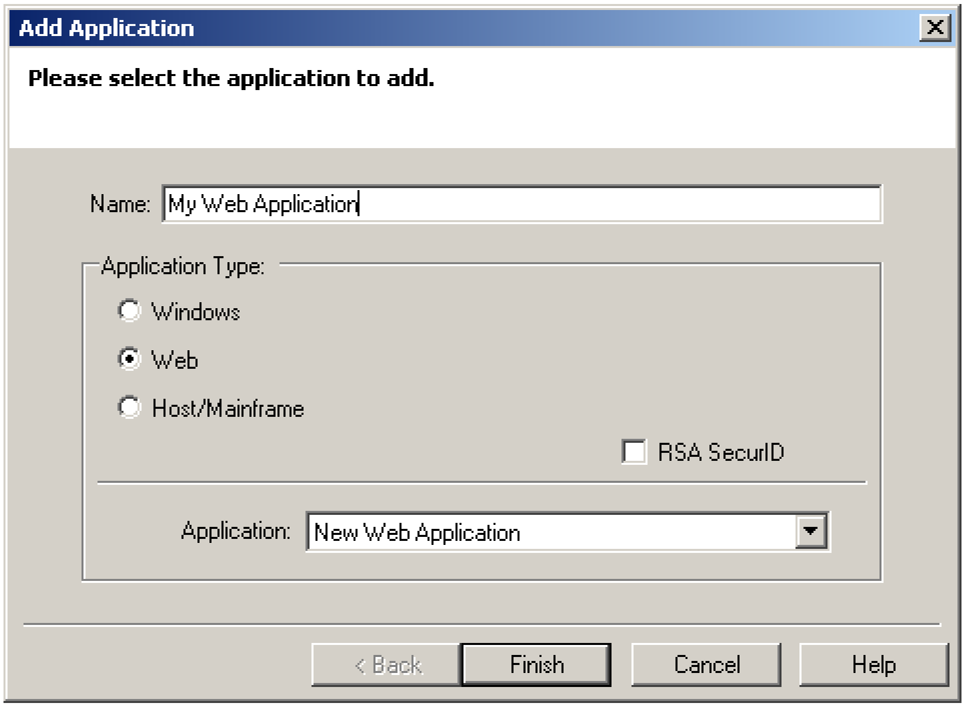
The procedures in this section use concepts and terminology explained earlier in this guide. When performing the procedures in this section, refer to Determining the Optimal Configuration for a Form to make configuration decisions that best suit the target application.

Basic Configuration

To complete a basic configuration of a form, do the following:

1. Open the ESSO-LM Administrative Console. By default, the shortcut is located in **Start → Programs → Oracle → ESSO-LM Console.**
2. In the left-hand tree, select the **Applications** node and do one of the following:
   - If you want to create a new template and define its first form:
     i. Click **Add** in the right-hand pane.
     ii. In the **New Application** dialog, enter a descriptive name for the template and click **Finish**. The new template appears in the list of stored templates.

Caution: If two or more application templates are named such that the name of one of the templates occurs in the beginning of the name of another template, the Agent will erroneously use the template with the shortest name to respond to all of the affected applications. To avoid this behavior, ensure that your template names do not begin with the same string of text.



Please select the application to add.

Name: My Web Application

Application Type:

○ Windows

● Web

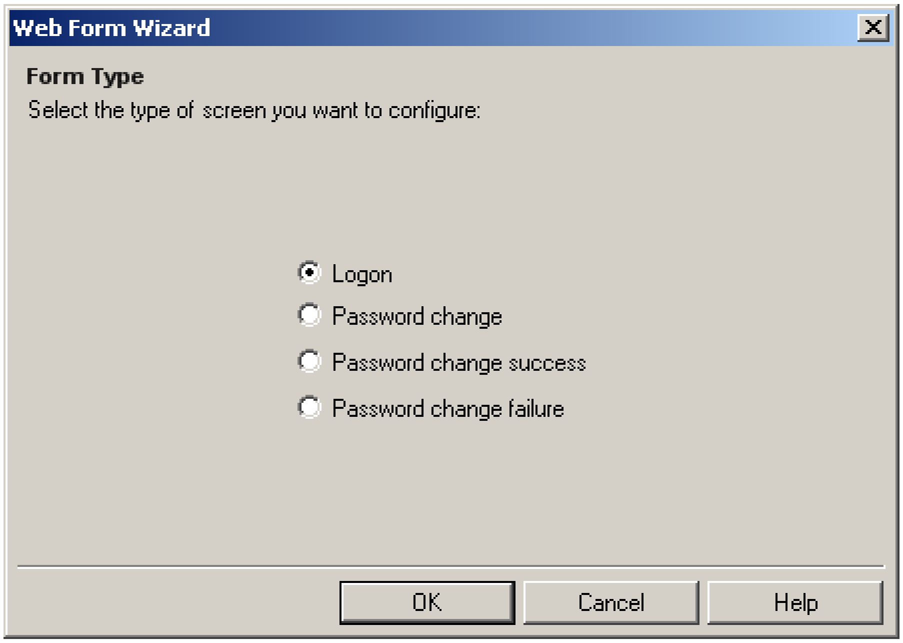
○ Host/Mainframe

□ RSA SecurID

Application: New Web Application

- If you want to add a form definition to an existing template, do the following:
    - i. Expand the **Applications** node and select the desired template.
      The template appears in the right-hand pane.
    - ii. Click **Add** at the bottom of the pane.

3. In the Web Form Wizard that appears, configure the fields and controls that you want the Agent to interact with when responding to the target form:
    - a. In the “Form Type” screen, select the desired form type and click **Next**.

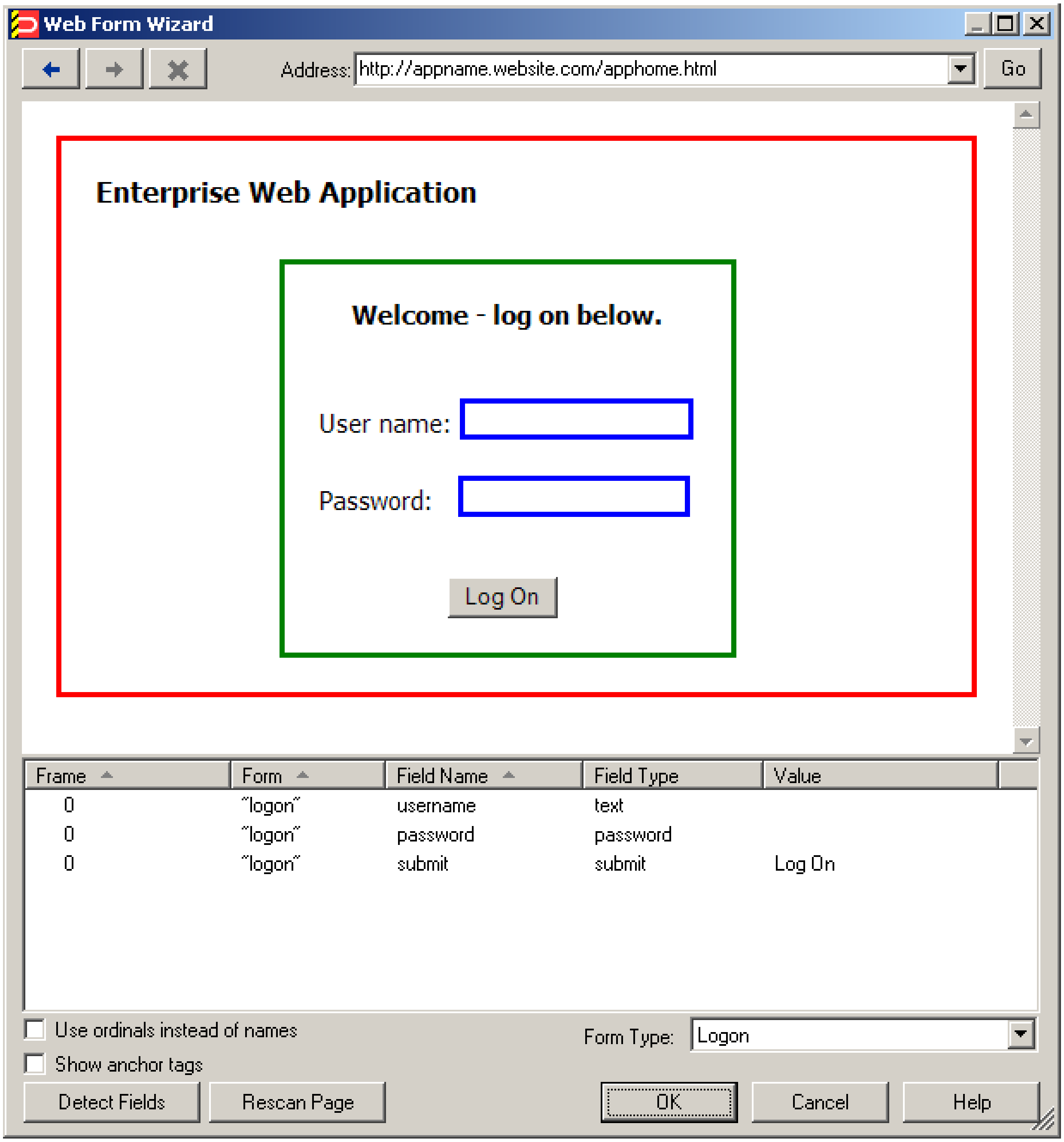


Form Type

Select the type of screen you want to configure:

● Logon
○ Password change
○ Password change success
○ Password change failure

b. In the next screen, enter the URL of the target page, form, or element and wait for it to complete loading in the preview pane.


Web Form Wizard
Address: http://appname.website.com/apphome.html
Go
Enterprise Web Application
Welcome - log on below.
User name:
Password:
Log On

Frame	Form	Field Name	Field Type	Value
0	"logon"	username	text	
0	"logon"	password	password	
0	"logon"	submit	submit	Log On

□ Use ordinals instead of names

□ Show anchor tags

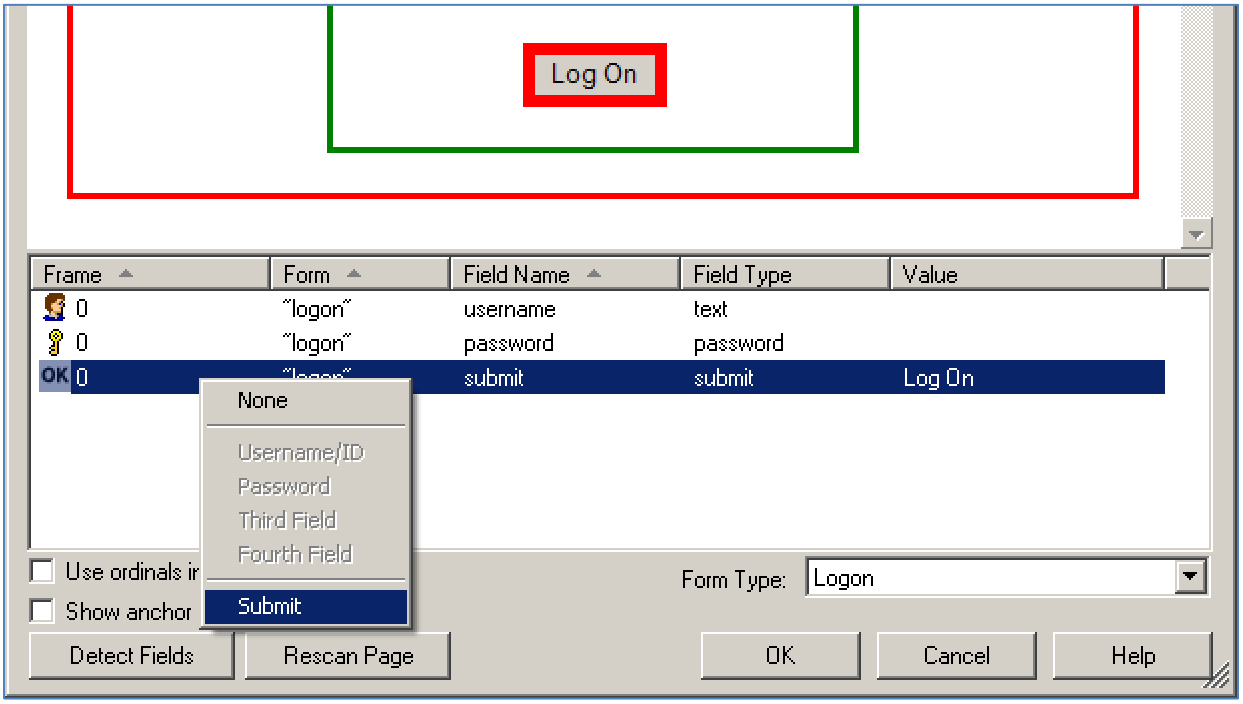
Form Type: Logon

Note: If you see two or more fields that share the same name, the Agent might respond to the form erroneously when configuration is complete. In such cases enable the Use ordinals instead of names option to uniquely identify the input fields to the Agent via ordinals (consecutive index numbers). For more information, see Understanding Web Form Detection.

c. In the field list at the bottom of the screen, select and configure the target fields and controls from among the objects in the list as follows:

Note: If one or more fields or controls are missing from the list, you may have to configure them manually. In such cases, complete the remainder of the Wizard as-is, then follow the instructions in Manually Configuring a Field or Control for each affected input field.

i. Right-click each desired field or control and select its function from the context menu. The selected field will be highlighted in the page preview.



Frame	Form	Field Name	Field Type	Value
0	"logon"	username	text	
0	"logon"	password	password	
OK 0	"logon"	submit	submit	Log On

☐ Use ordinals ir

☐ Show anchor

Form Type: Logon

Caution: While the **Detect Fields** feature is accurate the majority of the time, Oracle recommends always configuring fields and controls manually for guaranteed accuracy.

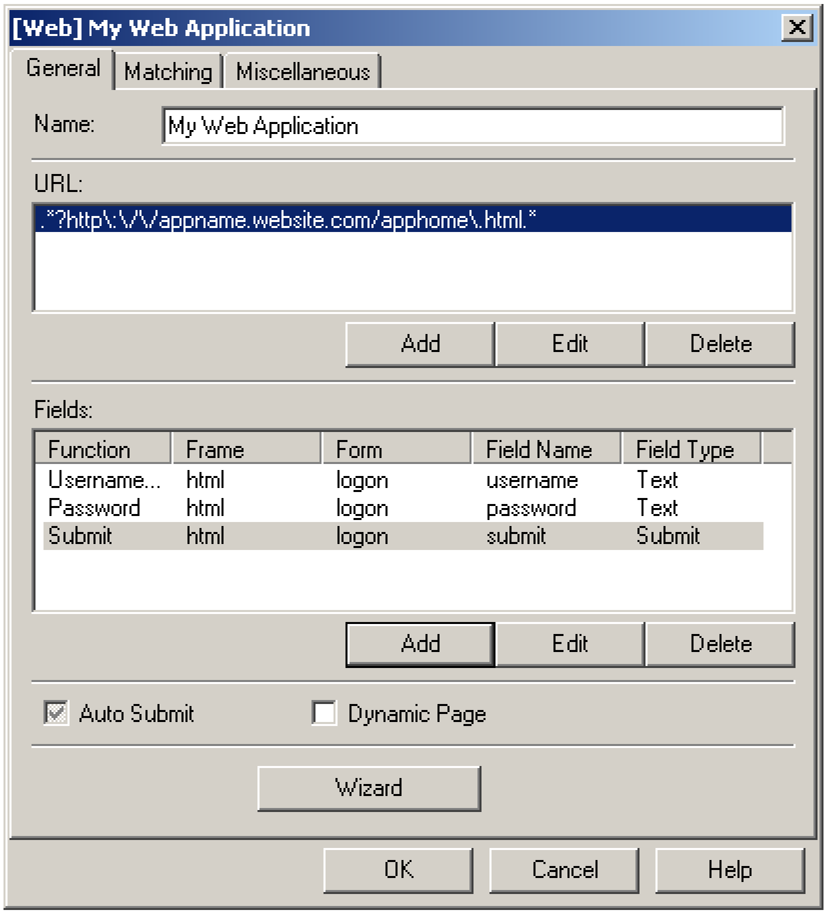
Note: If a "submit" button (usually labeled **OK**, **Logon**, etc.) is not visible in the list, ESSO-LM will still send a "submit" action to the application after injecting credentials. This action is equivalent to the user manually pressing the **Enter** key.

ii. If the submit control is missing, it might be implemented as an anchor (i.e., `<A HREF=`) tag. In such cases, enable the **Show anchor tags** option to expose such a control.

iii. When you have configured all of the visible input fields, click **OK**.

d. The Console displays the properties dialog for the Web form definition you have just configured. Do one of the following:

- If you have successfully completed all of the target input fields using the Web Form Wizard and require no further configuration, click OK in the dialog to dismiss it, then publish your changes to the repository as described in Publishing a Template to the Repository, if applicable.
- If you were unable to configure one or more target fields using the Web Form Wizard, continue to step 3 in Manually Configuring a Field or Control.
- If you need to modify existing or specify additional URL detection criteria, continue to Configuring URL Detection Criteria.
- If you need to configure additional field detection criteria via the **Matching** tab, continue to Configuring Additional Field Detection Criteria.



Name: My Web Application

URL:

`.*?http\:\/\/appname.website.com/apphome\.html.*`

Fields:

Function	Frame	Form	Field Name	Field Type
Username...	html	logon	username	Text
Password	html	logon	password	Text
Submit	html	logon	submit	Submit

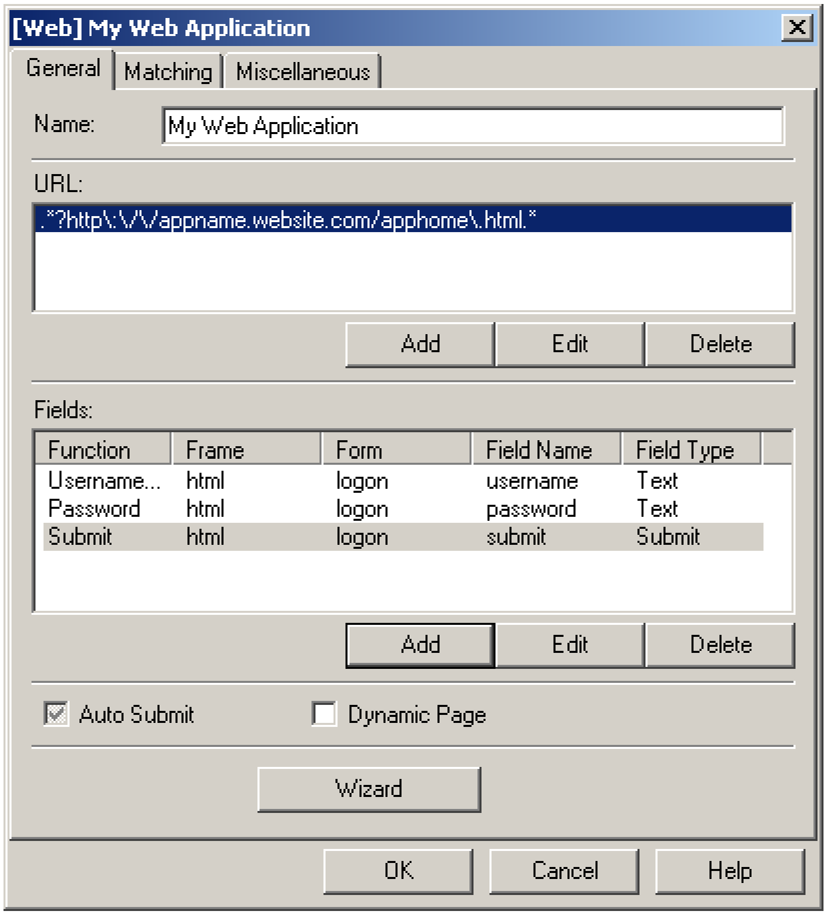
☑ Auto Submit ☐ Dynamic Page

Note: If the contents of the page are dynamic, i.e., change after the page has completed loading, select the **Dynamic Page** option to enable active page polling. This will allow the Agent to monitor the page for changes.

Manually Configuring a Field or Control

If you were unable to configure an input field using the Web Form Wizard, follow the procedure below to configure it manually:

1. In the Console, expand the **Applications** node and select the desired template.
2. In the **General** tab, select the desired form definition and click **Edit**. The form properties dialog appears.



Name: My Web Application

URL: .*?http\:\/\/appname.website.com/apphome\.html.*

Fields:

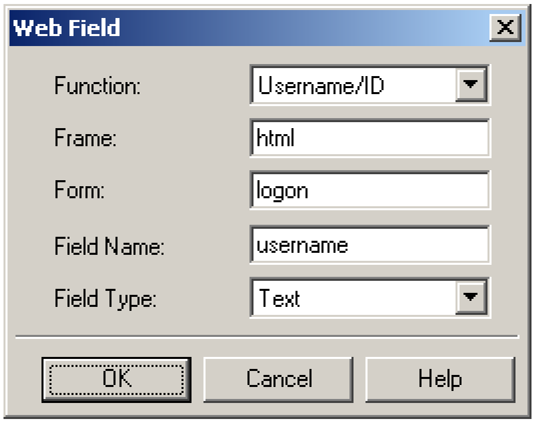
Function	Frame	Form	Field Name	Field Type
Username...	html	logon	username	Text
Password	html	logon	password	Text
Submit	html	logon	submit	Submit

☑ Auto Submit ☐ Dynamic Page

3. In the **Fields** section of the **General** tab of the form properties dialog, do the following:
    a. To configure a new field, click **Add**; to modify the configuration of an existing field, select the field in the list and click **Edit**.
    b. In the field properties dialog that appears, do the following:

> **Note:** See Understanding Field Detection to understand the values you need to specify here.

        i. In the **Function** drop-down list, select the field's purpose. The available values are **Username/ID**, **Password**, **Third Field**, and **Fourth Field**.
        ii. In the **Frame** field, enter the name or ordinal of the field's parent frame element.
        iii. In the **Form** field, enter the name or ordinal of the field's parent form element.
        iv. In the **Field Name** field, enter the name or ordinal of the target input field.
        v. In the **Field Type** drop-down list, select the type of the target input field.



Web Field

Function: Username/ID

Frame: html

Form: logon

Field Name: username

Field Type: Text

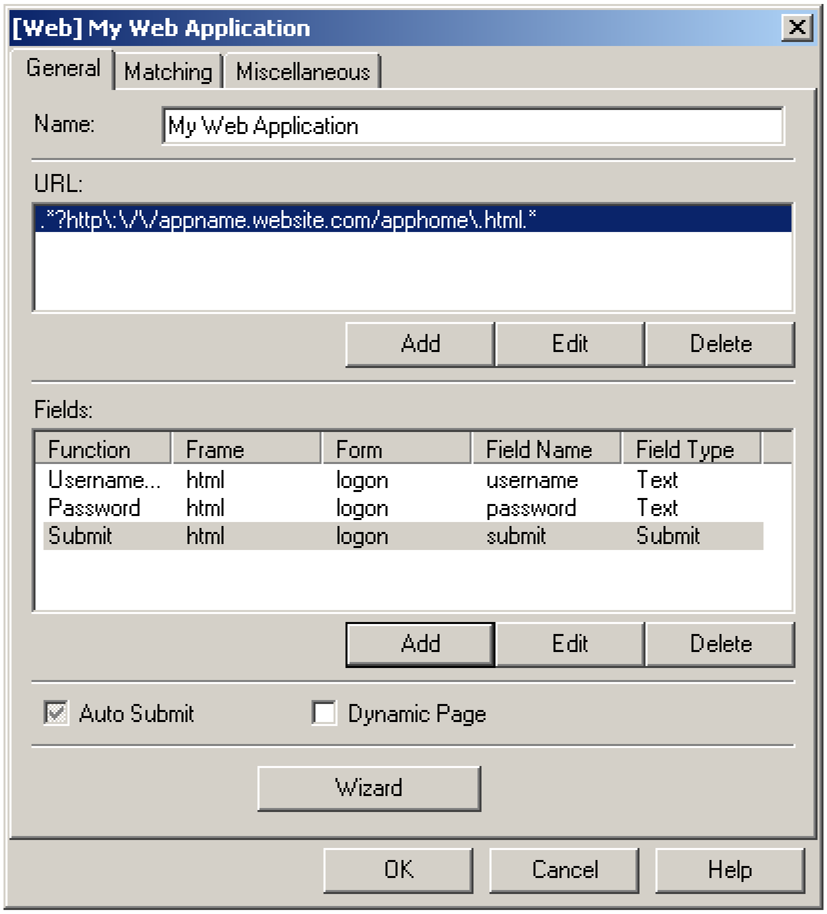
        vi. Click **OK** to save your field definition to the template.
    e. Repeat steps **i – vi** for each additional field you want to configure.
4. Click **OK** in the form properties dialog to dismiss it.
5. Publish your changes to the repository, as described in Publishing a Template to the Repository, if applicable.

Configuring URL Detection Criteria

If you need to modify the default URL detection criteria configured by the Web Form Wizard or specify additional criteria, do the following:

If you were unable to configure an input field using the Web Form Wizard, follow the procedure below to configure it manually:

1. In the Console, expand the **Applications** node and select the desired template.
2. In the **General** tab, select the desired form definition and click **Edit**. The form properties dialog appears.



Name: My Web Application

URL:

.*?http\:\/\/appname.website.com/apphome\.html.*

Fields:

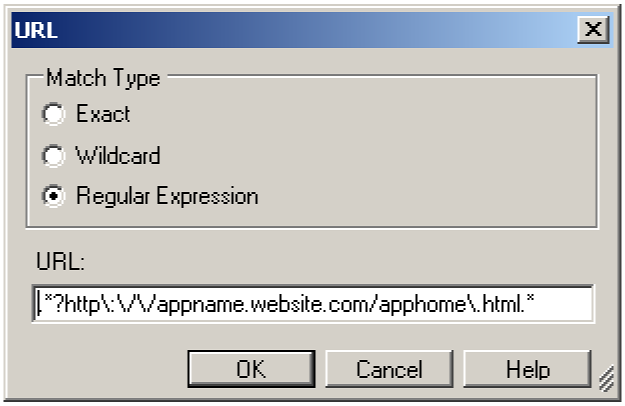
Function	Frame	Form	Field Name	Field Type
Username...	html	logon	username	Text
Password	html	logon	password	Text
Submit	html	logon	submit	Submit

☑ Auto Submit ☐ Dynamic Page

3. In the **URL** section of the **General** tab in the form properties dialog, do the following:
    a. To configure a new URL definition, click **Add**; to modify the configuration of an URL definition, select the definition in the list and click **Edit**.
    b. In the URL definition properties dialog that appears, do the following:

> **Note:** See Understanding URL Detection to understand the values you need to specify here.

i. Select the desired match type.
ii. Enter the desired match criterion.
iii. Click **OK** to store your URL definition.



Match Type
○ Exact
○ Wildcard
● Regular Expression

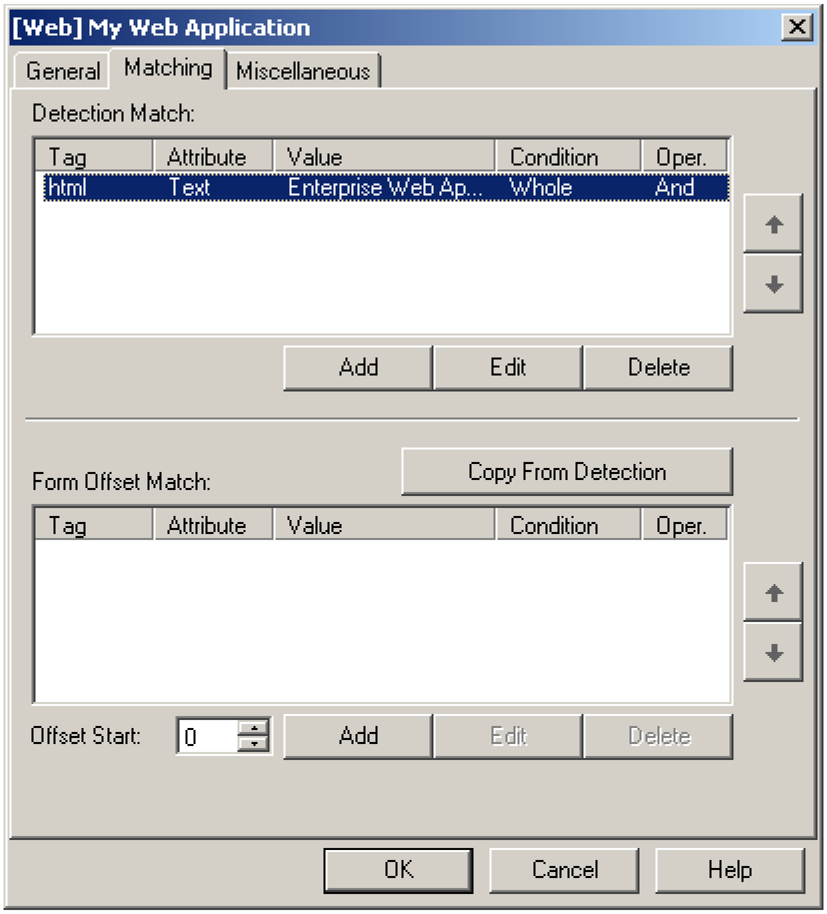
URL: `.*?http\:\/\/appname.website.com/apphome\.html.*`

    c. Repeat steps **i–iii** to configure any additional URL definitions.
    d. Click **OK** in the form properties dialog to dismiss it.
4. Publish your changes to the repository, as described in Publishing a Template to the Repository, if applicable.

Configuring Additional Field Detection Criteria

If you need to configure additional field detection criteria in either standard or offset matching mode, do the following:

1. In the Console, expand the **Applications** node and select the desired template.
2. In the **General** tab, select the desired form definition and click **Edit**. The form properties dialog appears.
3. In the dialog, select the **Matching** tab.



Detection Match:

Tag	Attribute	Value	Condition	Oper.
html	Text	Enterprise Web Ap...	Whole	And

Form Offset Match:

Tag	Attribute	Value	Condition	Oper.

Offset Start: 0

4. Complete the steps below in either the **Detection Match** or the **Form Offset Match** sections, depending on the desired matching type (see Detection Matching vs. Offset Matching for more information).

Note: When performing offset matching, you must specify the **Offset Start** value.

   a. To configure a new matching rule, click **Add**; to modify an existing rule, select the rule in the list and click **Edit**.

b. In the “Match Criteria” dialog that appears, do the following:

> **Note:** See Understanding Detection Matching to understand the values you need to specify here.

i. In the **Tag** field, specify the name of the parent element.

ii. (Optional) If there are multiple instances of the parent element within the page, determine its ordinal, select the **Match tag instance** box and enter the ordinal in the value field.

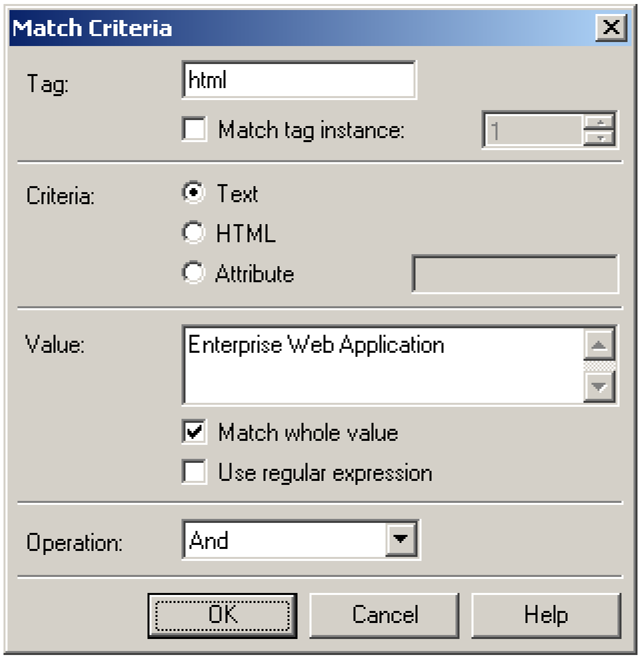
iii. In the **Criteria** field, select the desired matching mode.

iv. In the Value field, enter the target criterion value.

v. For most scenarios, leave the Match whole value option enabled; this will cause ESSO-LM to match strictly against the criterion value. If you disable this option, ESSO-LM will also consider partial matches against the criterion value as positive.

vi. If you want to use regular expressions as part of the criterion value, enable the Use regular expression option.

vii. (Optional) If you are creating a matching rule chain to form a complex matching rule, select the desired Boolean operator in the **Operation** field to indicate how this rule will chain with the preceding rule.


Match Criteria

Tag: html

☐ Match tag instance: 1

Criteria: ● Text ○ HTML ○ Attribute

Value: Enterprise Web Application

☑ Match whole value

☐ Use regular expression

Operation: And

viii. Click **OK** to save your matching rule.

c. Repeat steps **i-viii** for each additional matching rule.

d. Click **OK** in the form properties dialog to dismiss it.

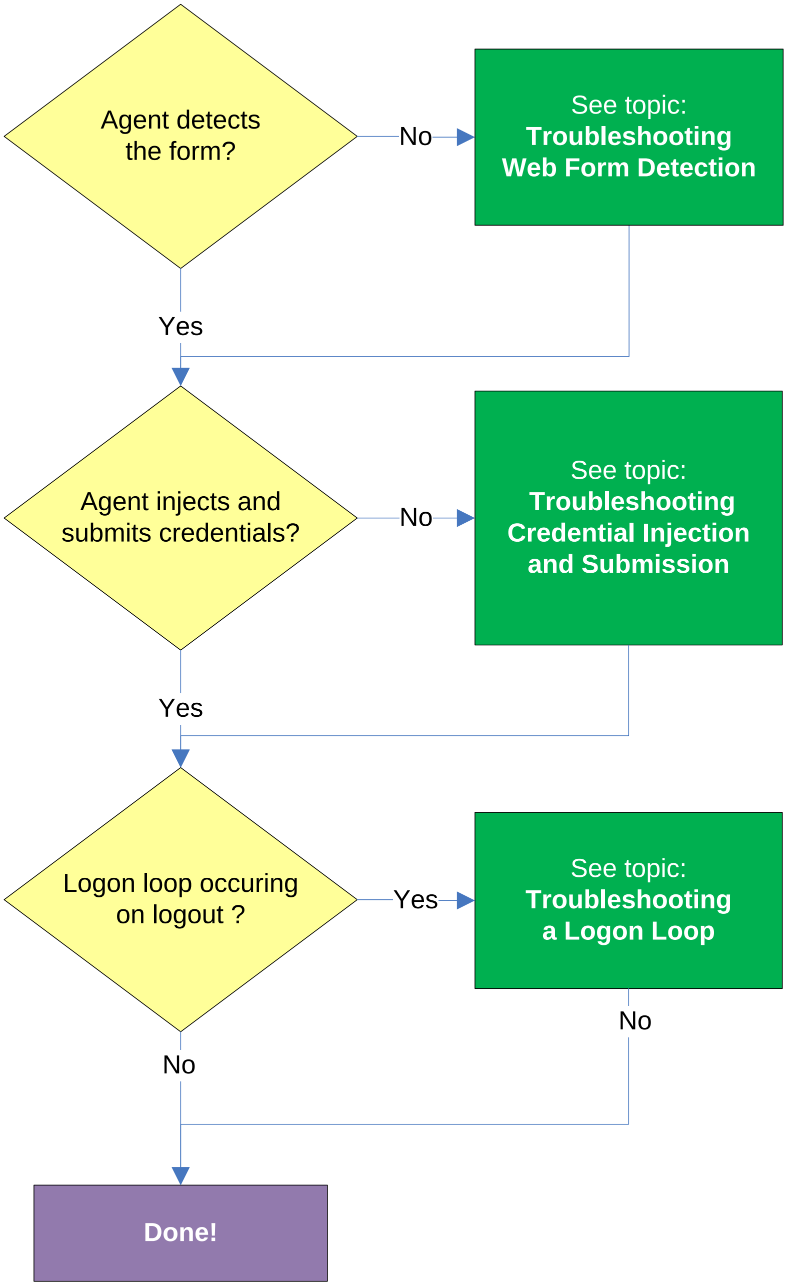
5. Publish your changes to the repository, as described in Publishing a Template to the Repository, if applicable.

Testing the Configuration of a Form

To test the configuration of a form, do the following:

1. Launch the Agent.
2. Access the target application and invoke the desired form.
3. Use the appropriate flowchart to test the form configuration.

Testing the Configuration of a Logon Form


Agent detects the form?
No
See topic: Troubleshooting Web Form Detection
Yes
Agent injects and submits credentials?
No
See topic: Troubleshooting Credential Injection and Submission
Yes
Logon loop occuring on logout ?
Yes
See topic: Troubleshooting a Logon Loop
No
No
Done!

Agent detects form?

Once the Agent has been provided with the template, it will automatically respond to the target form, unless the automatic response feature has been explicitly disabled. If the Agent fails to respond to the form, see Troubleshooting Web Form Detection.

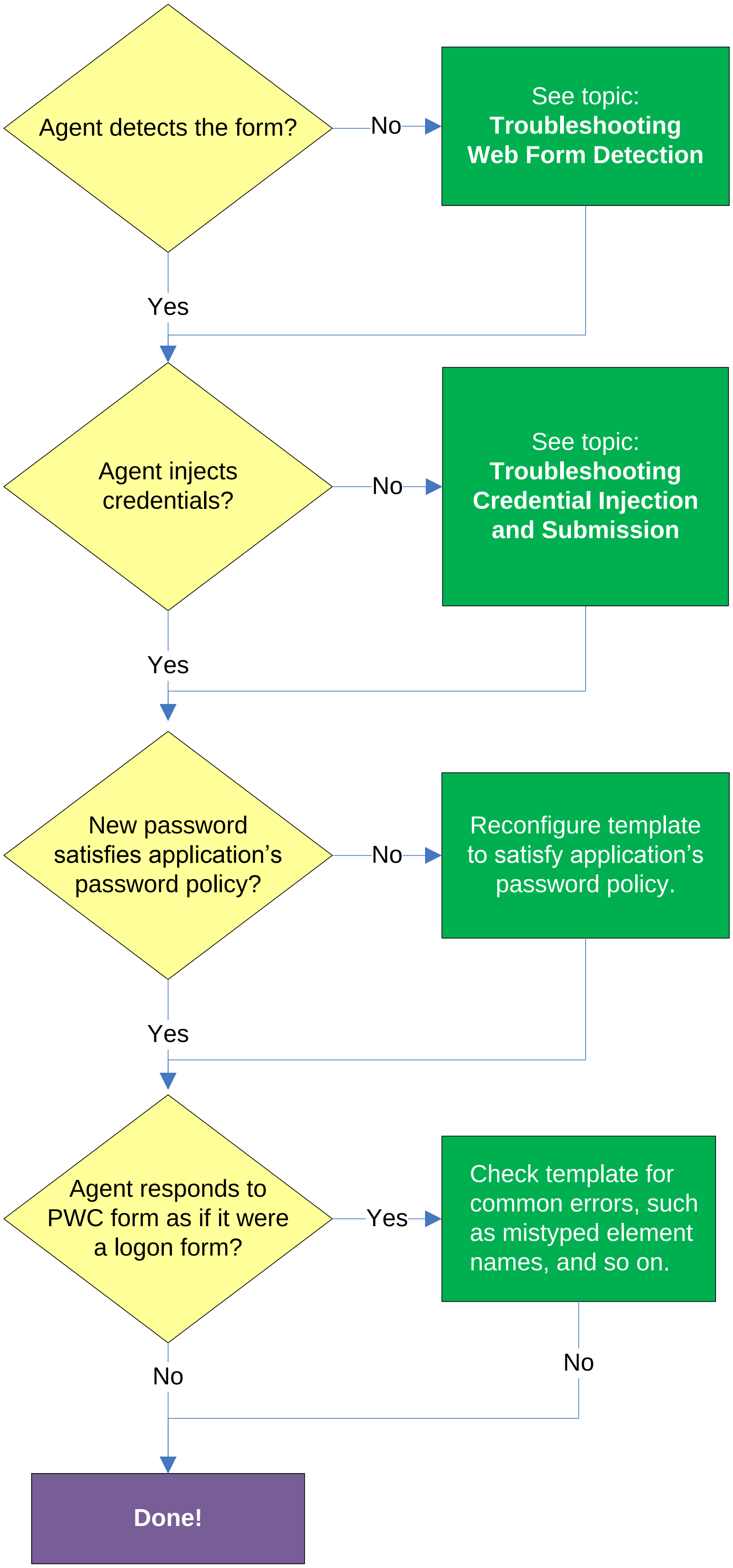
Agent injects credentials?

If credentials have been stored for the target application in the user's store, the Agent will inject them into the appropriate fields upon successful application detection. The Agent will also automatically submit the credentials unless the "Auto-Submit" feature has been explicitly disabled. If credential injection fails, see Troubleshooting Credential Injection and Submission.

Logon loop occurring on logout?

Some applications display their logon screen upon logout, which causes the Agent to enter a logon loop and effectively prevents the user from logging out of the application unless the Agent is shut down.
If this happens, see Troubleshooting a Logon Loop.

Testing the Configuration of a Password Change Form


Agent detects the form?
No
See topic: Troubleshooting Web Form Detection
Yes
Agent injects credentials?
No
See topic: Troubleshooting Credential Injection and Submission
Yes
New password satisfies application's password policy?
No
Reconfigure template to satisfy application's password policy.
Yes
Agent responds to PWC form as if it were a logon form?
Yes
Check template for common errors, such as mistyped element names, and so on.
No
No
Done!

Agent detects the form?

Once the Agent has been provided with the template, it will automatically respond to the target form, unless the automatic response feature has been explicitly disabled. If the Agent fails to respond to the application, see Troubleshooting Web Form Detection.

Agent injects and submits credentials?

When the Agent detects the password change, it injects credentials into the appropriate fields and submits them to the application, unless the Auto Submit feature has been explicitly disabled. If credential injection is erratic or does not occur at all, see Troubleshooting Credential Injection and Submission.

New password satisfies application's password policy?

If the new password generated by ESSO-LM does not satisfy the application's own password policy, password change will be unsuccessful. If you determine this to be the case, compare the password generation policy currently deployed to the Agent with the password policy of the target application and correct any inconsistencies that may cause password change failure.

Agent responds to password change form as if it were a logon?

If the Agent responds to the password change form as if it were a logon form (i.e., Agent injects and submits the user's currently stored credentials), check for the following;

- Configuration mistakes in the template, such as incorrect element or form type, erroneous input field definitions, and so on.
- Check whether the password change form has a dynamic URL and update the template accordingly if it does.
- If you are using detection matching, check whether you are using the correct matching type and examine your matching strings for errors.

Publishing a Template to the Repository

Once you have successfully tested your application template, you can distribute it to end-user machines by publishing it to the selected target container within your repository, either in a directory-style hierarchy (default), or as a flat configuration file.

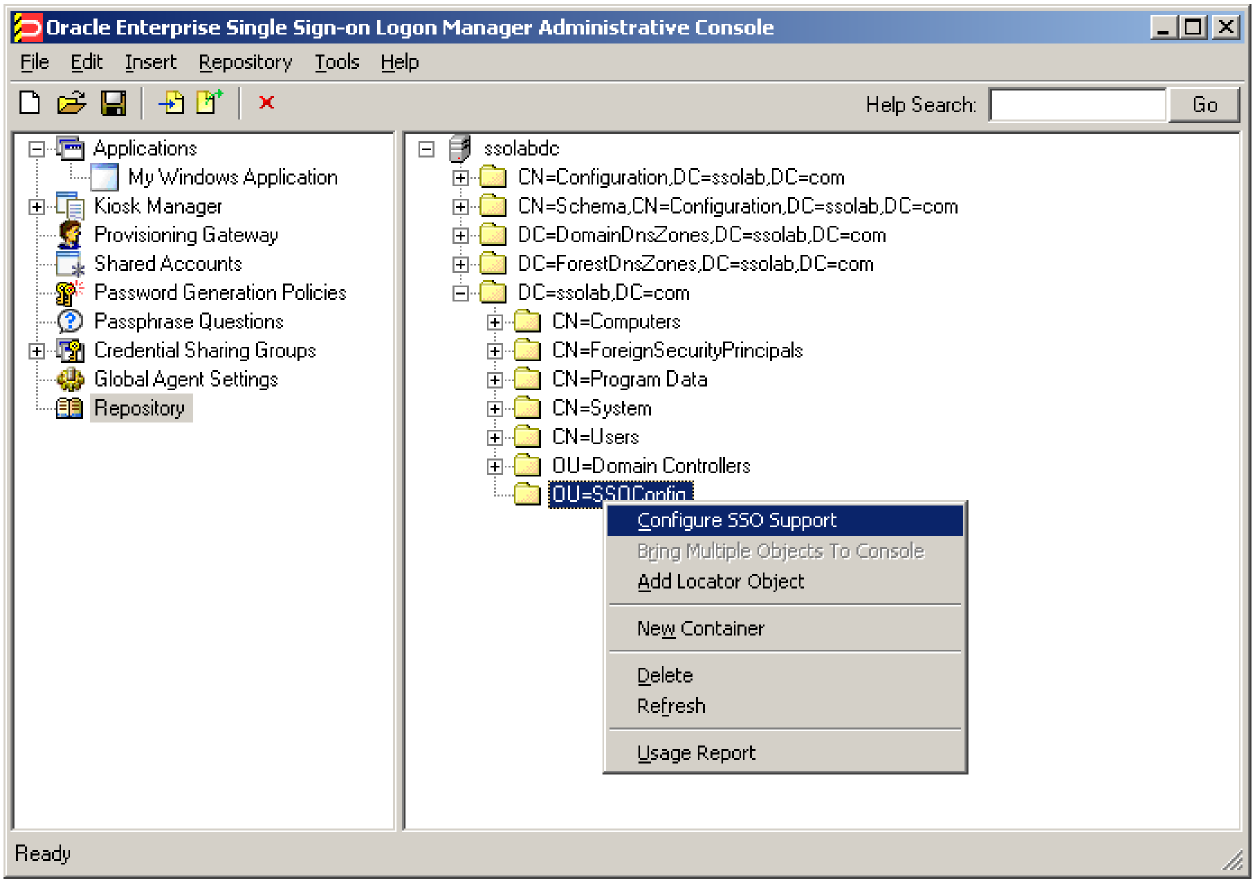
> **Note:** For more information on deploying ESSO-LM with a repository and best practices for structuring the repository tree, see the *ESSO-LM Best Practices* guide for your platform.

Publishing a Template with ESSO-LM Versions Prior to 11.1.1.2.0

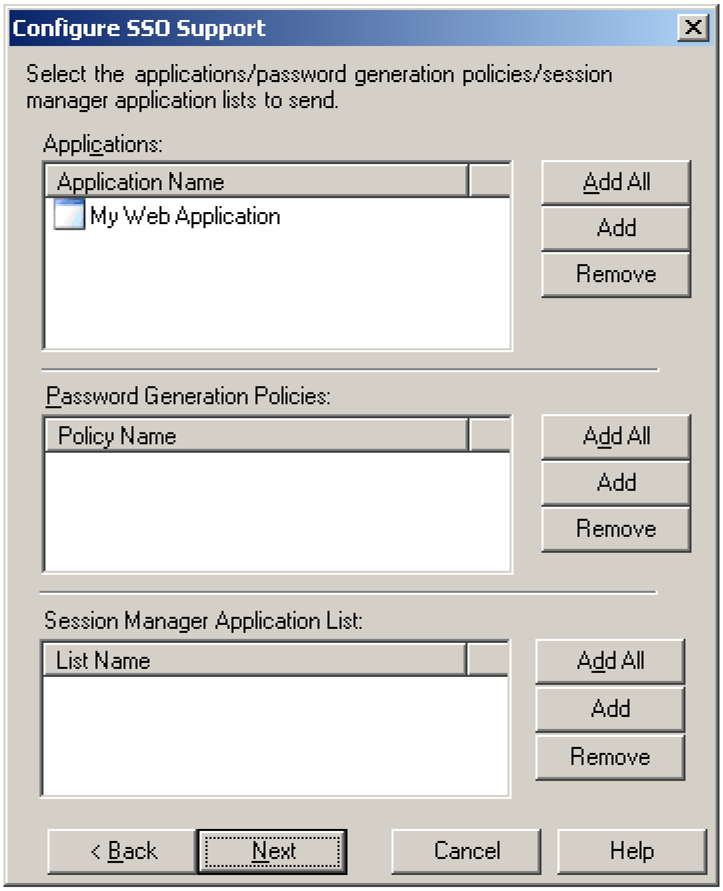
> **Note:** Before performing this procedure, make sure you are familiar with the structure and configuration of your repository.

To select and publish the desired templates and other configuration objects to the repository:

1. Launch the ESSO-LM Administrative Console.
2. In the left-hand pane, right-click the **Repository** node and select **Connect To…** from the context menu.
3. Fill in the required connection information and click **OK** to establish the connection.
   The directory tree appears in the right-hand pane.
4. In the tree, navigate to and select the desired target container.
5. Right-click the target container and select **Configure SSO Support** from the context menu.



6. In the “Configure SSO Support” dialog that appears, click **Administrative Console**.
7. In the screen that appears, select **Advanced mode** and click **Next**.
8. In the screen that appears, specify the objects you want to publish to the selected container.



Configure SSO Support

Select the applications/password generation policies/session manager application lists to send.

Applications:

Application Name
My Web Application

Password Generation Policies:

Policy Name

Session Manager Application List:

List Name

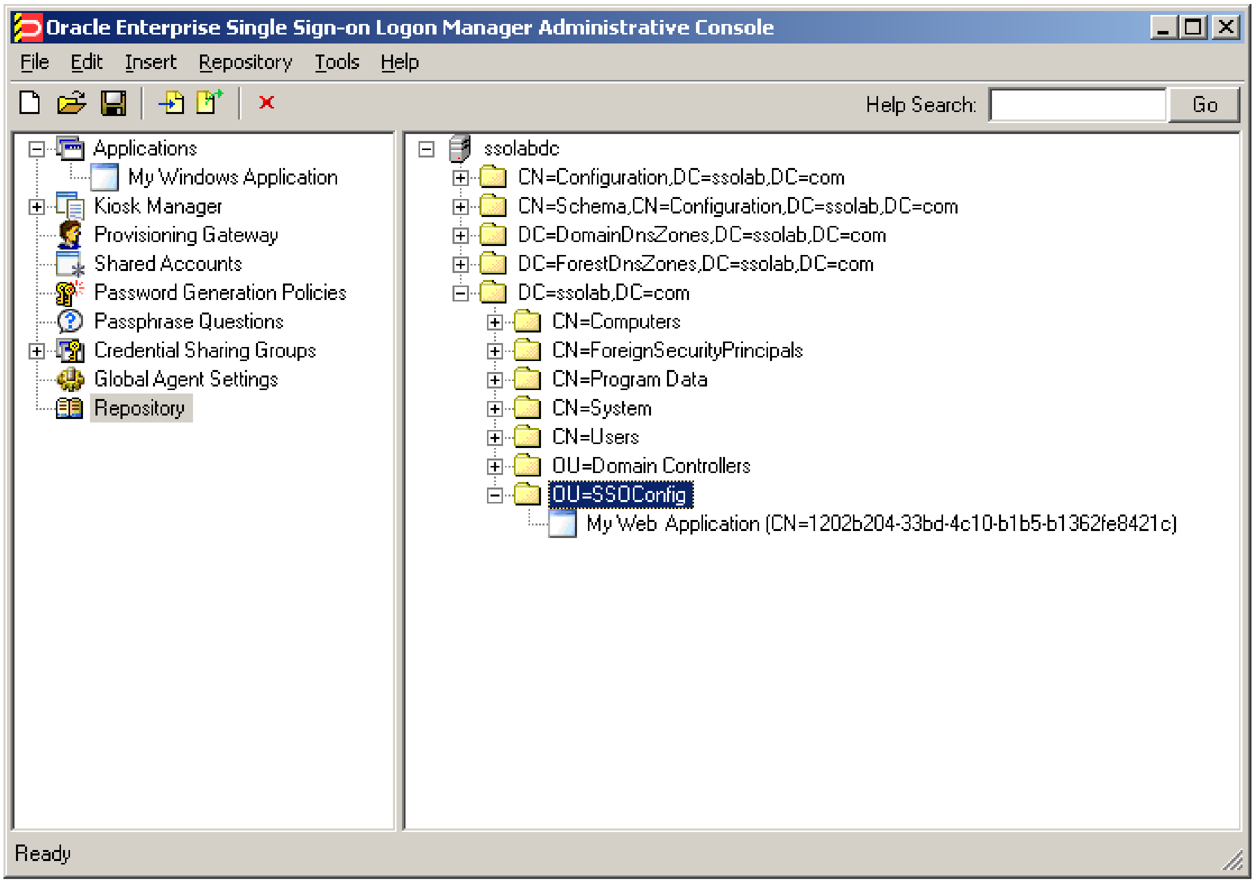
- To add select objects in a given category, do the following:
  i. Click **Add**.
  ii. In the dialog that appears, select the desired objects.

  Tip: Use **Ctrl-click** to select multiple specific objects, or **Shift-click** to select the first and last objects in a desired range.

  iii. Click **OK**.
- To add all objects in a category that currently exist in the Console, click **Add All** in that category’s section of the dialog.
- To remove an object from a list of objects to be published, click **Remove** in that category’s section of the dialog.

When you have made your selections, click **Next**.

9. In the summary screen, review your choices. If you want to make changes, click **Back**; otherwise, click **Finish** to publish the selected objects to the chosen target container. The published objects appear under the target container in the right-hand pane of the Console.


Oracle Enterprise Single Sign-on Logon Manager Administrative Console
File
Edit
Insert
Repository
Tools
Help
Help Search:
Go
Ready

- Applications
  - My Windows Application
- Kiosk Manager
- Provisioning Gateway
- Shared Accounts
- Password Generation Policies
- Passphrase Questions
- Credential Sharing Groups
- Global Agent Settings
- Repository

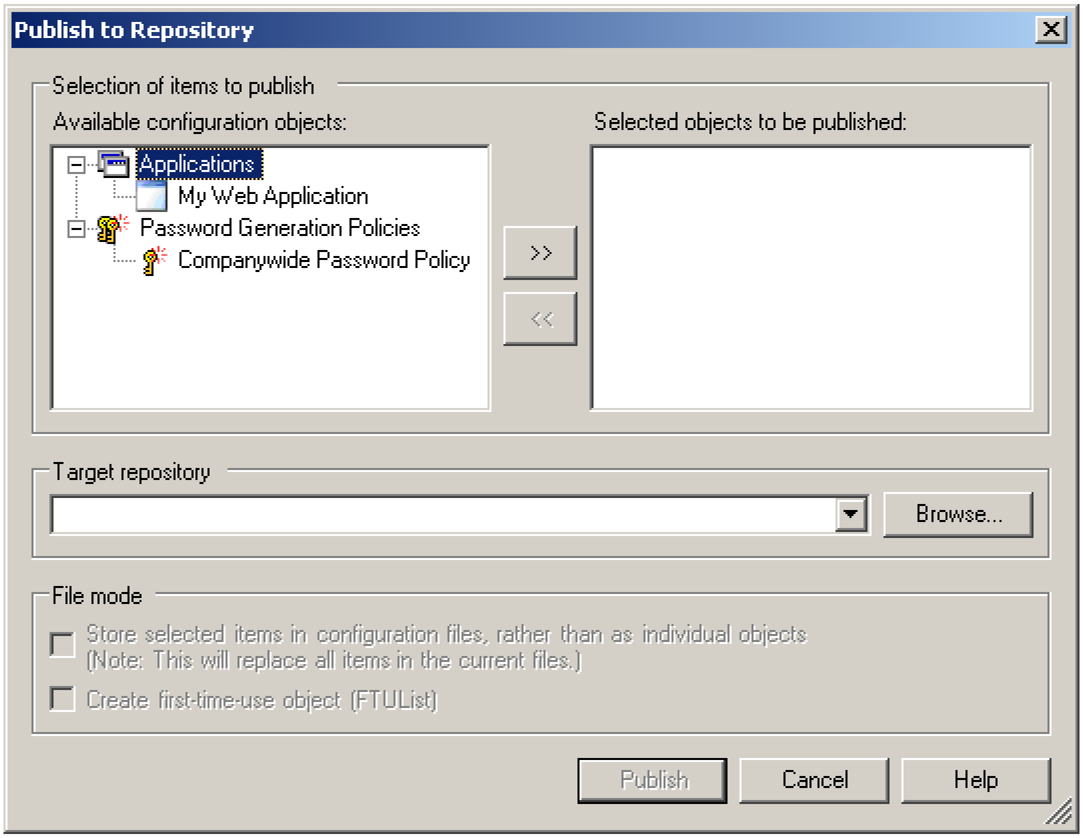
- ssolabdc
  - CN=Configuration,DC=ssolab,DC=com
  - CN=Schema,CN=Configuration,DC=ssolab,DC=com
  - DC=DomainDnsZones,DC=ssolab,DC=com
  - DC=ForestDnsZones,DC=ssolab,DC=com
  - DC=ssolab,DC=com
    - CN=Computers
    - CN=ForeignSecurityPrincipals
    - CN=Program Data
    - CN=System
    - CN=Users
    - OU=Domain Controllers
    - OU=SSOConfig
      - My Web Application (CN=1202b204-33bd-4c10-b1b5-b1362fe8421c)

Publishing a Template with ESSO-LM Version 11.1.1.2.0 and Above

Note: Before performing this procedure, make sure you are familiar with the structure and configuration of your repository.

To select and publish the desired templates and other configuration objects to the repository:

1. Launch the ESSO-LM Administrative Console.
2. Right-click the **Applications** node and select **Publish...** from the context menu.
   The “Publish to Repository” dialog appears.


Publish to Repository
Selection of items to publish
Available configuration objects:
Applications
My Web Application
Password Generation Policies
Companywide Password Policy
>>
<<
Selected objects to be published:
Target repository
Browse...
File mode
Publish
Cancel
Help

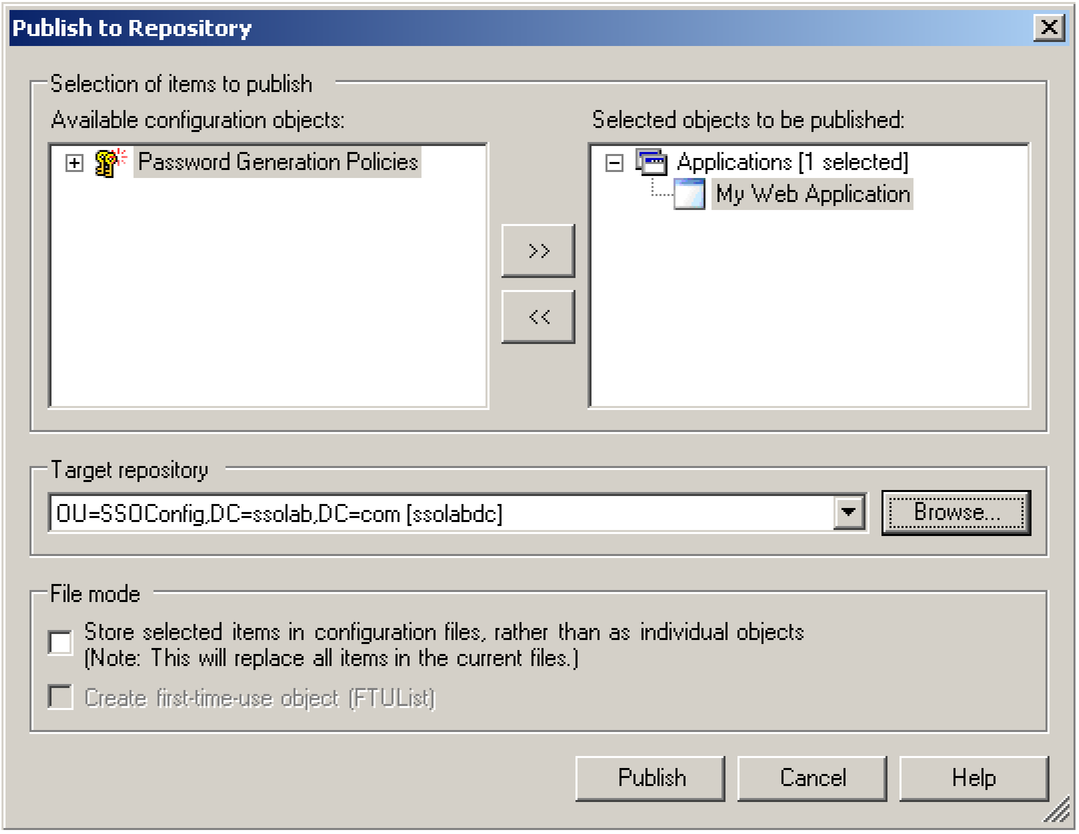
☐ Store selected items in configuration files, rather than as individual objects (Note: This will replace all items in the current files.)

☐ Create first-time-use object (FTUList)

3. In the **Available configuration objects** list, navigate to and select the desired objects.

Note: Only categories for which objects have been configured will appear in this list. For example, if no password generation policies exist, the corresponding category will not appear in this list.

4. Click **>>** to move the selected objects to the **Selected objects to be published** list. (To remove an object from this list and not publish it, select the object and click **<<**.)


Publish to Repository
Selection of items to publish
Available configuration objects:
Password Generation Policies
Selected objects to be published:
Applications [1 selected]
My Web Application
>>
<<
Target repository
Browse...
File mode
Publish
Cancel
Help

Target repository: OU=SSOConfig,DC=ssolab,DC=com [ssolabdc]

□ Store selected items in configuration files, rather than as individual objects (Note: This will replace all items in the current files.)

□ Create first-time-use object (FTUList)

5. Select the target container to which you want to publish the selected objects by doing one of the following:

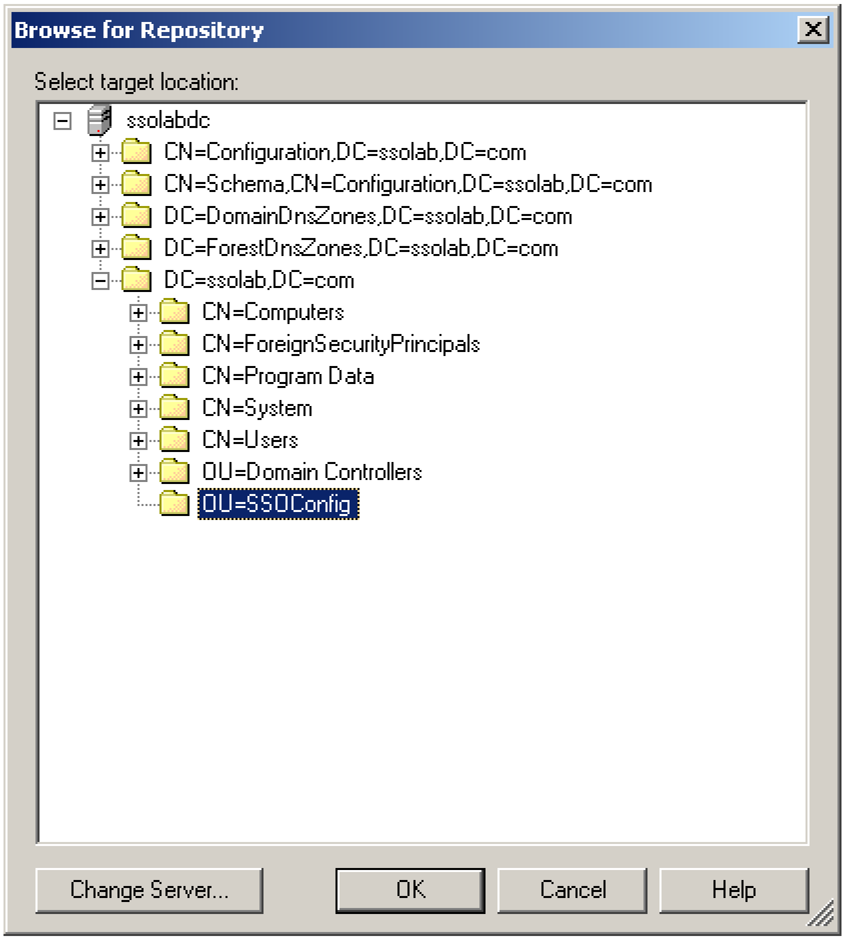
   ○ If you have previously published to the desired container, select it from the **Target Repository** drop-down list.

   ○ If you have not previously published to the desired container, or if the target container path does not appear in the **Target Repository** drop-down list, you must use the Browse feature to find and select the target container:

      i. Click **Browse** to browse the directory tree.

      Note: If you are not already connected to the directory, the Console will prompt you to provide the required connection information.

ii. In the “Browse for Repository” dialog that appears, navigate to and select the target container.



Browse for Repository

Select target location:

```
ssolabdc
  CN=Configuration,DC=ssolab,DC=com
  CN=Schema,CN=Configuration,DC=ssolab,DC=com
  DC=DomainDnsZones,DC=ssolab,DC=com
  DC=ForestDnsZones,DC=ssolab,DC=com
  DC=ssolab,DC=com
    CN=Computers
    CN=ForeignSecurityPrincipals
    CN=Program Data
    CN=System
    CN=Users
    OU=Domain Controllers
    OU=SSOConfig
```

Note: If you want to create a new container, right-click the desired parent container, select **New Container** from the context menu, enter the desired name for the new container, and click **OK** to complete the process.

6. (Optional) If your environment calls for storing configuration objects in flat-format, select the check box **Store selected items in configuration files, rather than as individual objects.**

Note: Selecting this option will overwrite all items stored in existing configuration files, if present in the target container.

7. (Optional) If you want to create the first-time use object (`FTUList`), select the corresponding check box.

Note: This option only becomes active if you choose to store your configuration objects in flat format in step 6.

8. Click **Publish**. The Console publishes the selected objects to the target repository.

> **Caution:** Do not attempt to dismiss the dialog or close the Console until the publishing process completes. The dialog will disappear automatically when the objects have been published.

For more information on the publishing process, see the ESSO-LM Administrative Console help.

Part 3: Troubleshooting Detection and Response Issues

This part describes diagnosis and resolution steps for the most common issues that may cause the Agent to erratically detect and/or respond to Web forms. It covers the following topics:

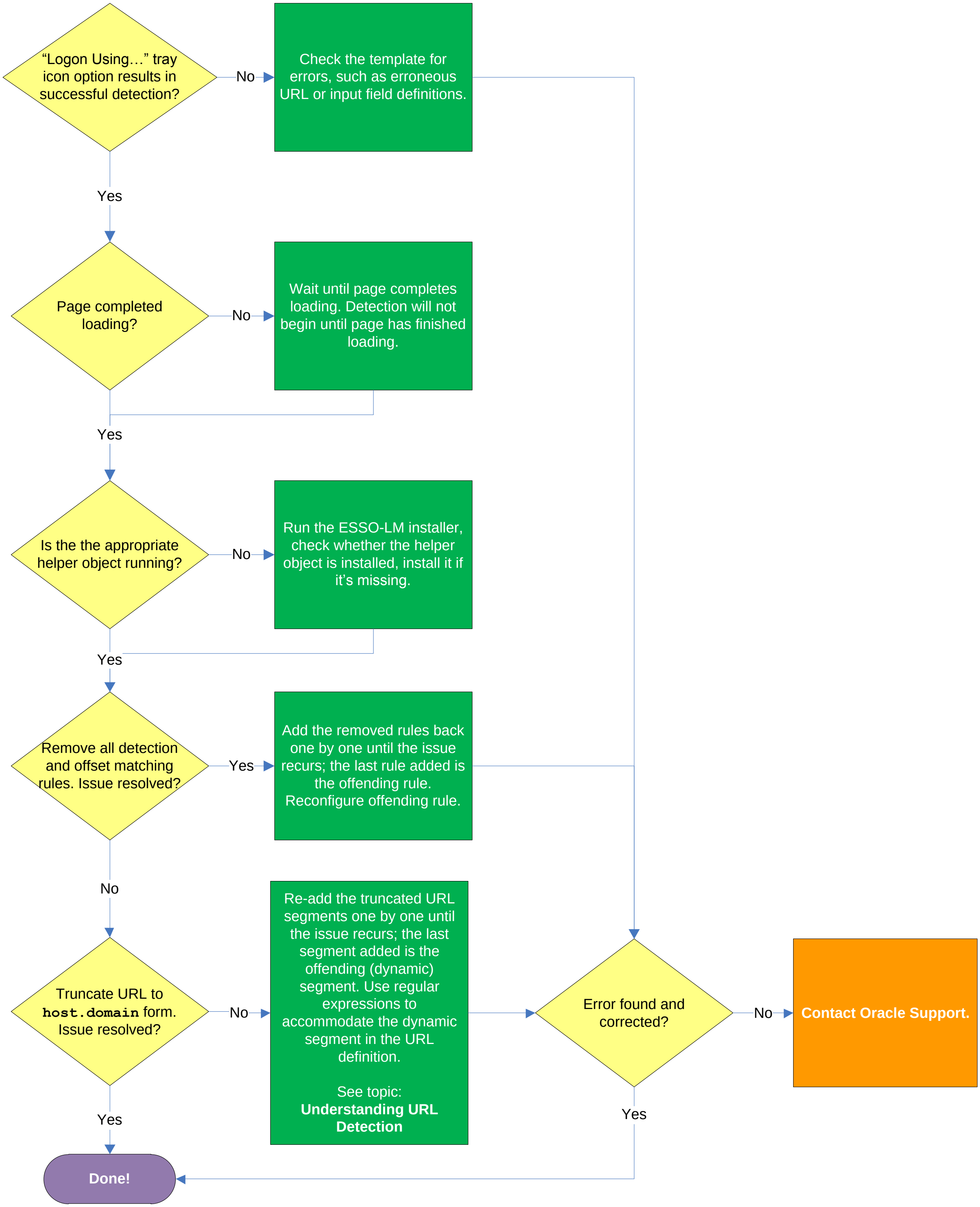
- Troubleshooting Web Form Detection
- Troubleshooting Credential Injection and Submission
- Troubleshooting Detection Matching
- Troubleshooting a Logon Loop
- Troubleshooting Java Application Issues

Tip: If the steps in this section do not resolve your issue, you can troubleshoot further by tracing and logging the activity of ESSO-LM and submitting the logged information to Oracle Support for analysis. For this purpose, Oracle provides the Trace Controller utility, available from Oracle Support. For information on how to use the utility, see the *How-To* guide *Using the Trace Controller Utility*.

If the steps in this section do not resolve the issue, see Contacting Oracle Support to obtain further assistance.

Troubleshooting Web Form Detection

Use the steps below to diagnose erratic form detection.


"Logon Using…" tray icon option results in successful detection?
No
Check the template for errors, such as erroneous URL or input field definitions.
Yes
Page completed loading?
No
Wait until page completes loading. Detection will not begin until page has finished loading.
Yes
Is the the appropriate helper object running?
No
Run the ESSO-LM installer, check whether the helper object is installed, install it if it's missing.
Yes
Remove all detection and offset matching rules. Issue resolved?
Yes
Add the removed rules back one by one until the issue recurs; the last rule added is the offending rule. Reconfigure offending rule.
No
Truncate URL to host.domain form. Issue resolved?
No
Re-add the truncated URL segments one by one until the issue recurs; the last segment added is the offending (dynamic) segment. Use regular expressions to accommodate the dynamic segment in the URL definition.
See topic: Understanding URL Detection
Error found and corrected?
No
Contact Oracle Support.
Yes
Yes
Done!

“Logon Using ESSO-LM” tray icon option results in successful detection?

If the Agent does not detect the form, but invoking the **Logon Using ESSO-LM** option from the Agent’s system tray icon results in successful detection, check the template for errors such as incorrect or imprecise URL and input field definitions, matching rules, and so on.

Page completed loading?

The Agent will not begin detection until the browser indicates that the page has loaded completely. If the page is loading slowly, you must allow it to load completely before deciding whether detection is occurring properly. Usually, the browser’s status bar, located at the bottom of the browser window, indicates the page has completed loading by displaying an appropriate message, such as “Done” or “Finished.”

Is the appropriate helper object running?

In order to communicate with the installed Web browser(s), the Agent uses helper objects that hook into the browser(s) and provide ESSO-LM with a means of sending and receiving data to/from the browser. If the helper objects are not running in the background, ESSO-LM will be unable to communicate with the browser(s). For more information, see Understanding Web Form Detection.

Removing all detection and offset matching rules results in successful detection?

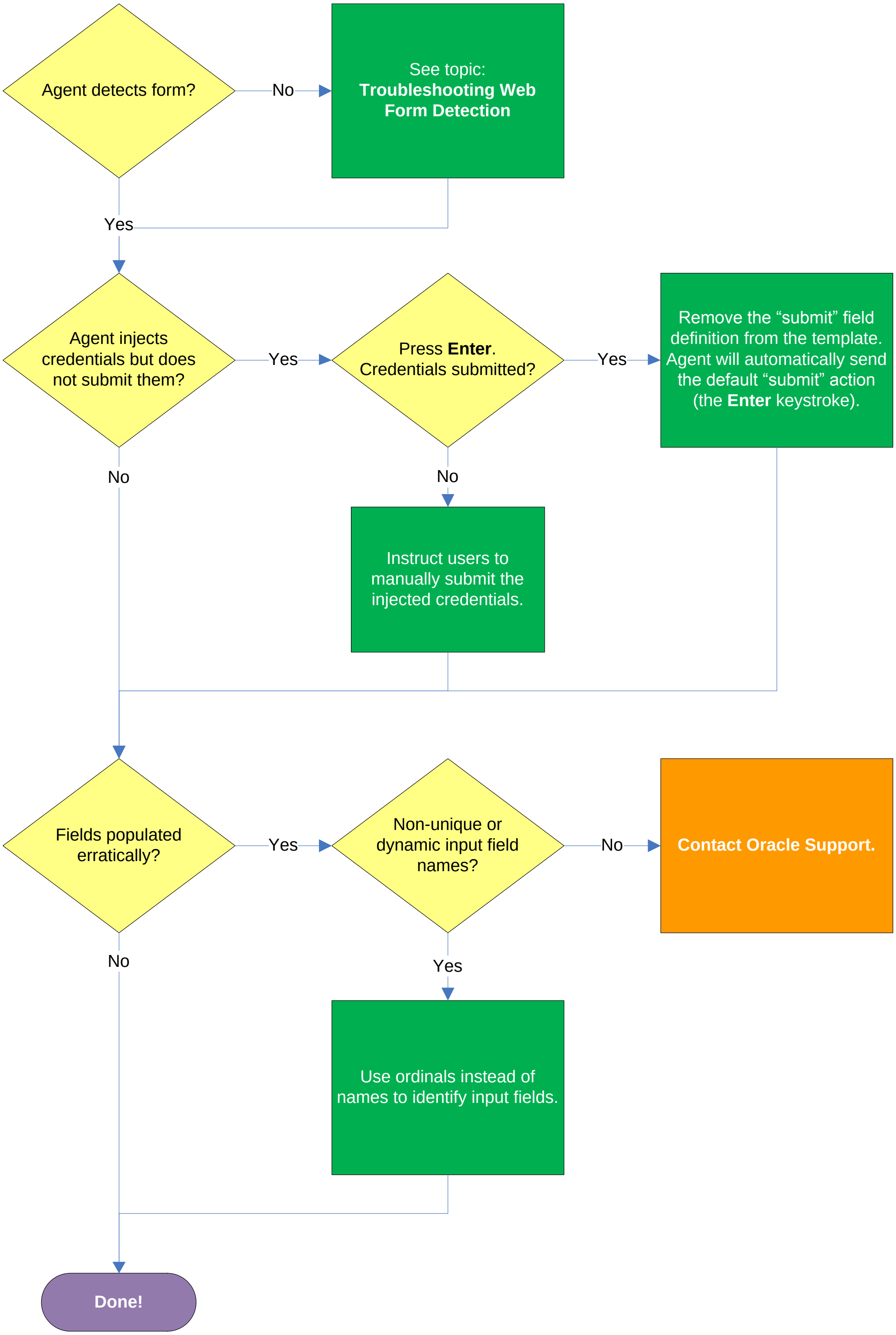
You might have configured one or more matching rules in a way which prevents successful detection of the target Web form. Remove all matching rules and re-add them one by one while testing detection until you identify the offending rule.

Truncating URL definition to `host.domain` form results in successful detection?

The URL might contain a dynamic segment which is preventing the Agent from detecting the form consistently. Simplify the URL definition to `host.domain` form and test the template – if detection is successful, rebuild your URL definition one segment at a time until the offending segment is identified, then use regular expressions to accommodate the dynamic segment. For more information, see Understanding URL Detection.

If the above steps do not resolve the issue, see Contacting Oracle Support to obtain further assistance.

Troubleshooting Credential Injection and Submission


Agent detects form?
No
See topic:
Troubleshooting Web Form Detection
Yes
Agent injects credentials but does not submit them?
Yes
Press **Enter**. Credentials submitted?
Yes
Remove the “submit” field definition from the template. Agent will automatically send the default “submit” action (the **Enter** keystroke).
No
No
Instruct users to manually submit the injected credentials.
Fields populated erratically?
Yes
Non-unique or dynamic input field names?
No
Contact Oracle Support.
No
Yes
Use ordinals instead of names to identify input fields.
Done!

Agent detects the form?

If the Agent does not detect the form at all and does not attempt response, see Understanding Web Form Detection to identify possible problems in template configuration that might be hampering detection.

Agent injects credentials but does not submit them?

If the Agent successfully detects the fields and injects the correct credentials, but is unable to automatically submit them to the application for processing, press Enter to see if the credentials are submitted, then do one of the following:

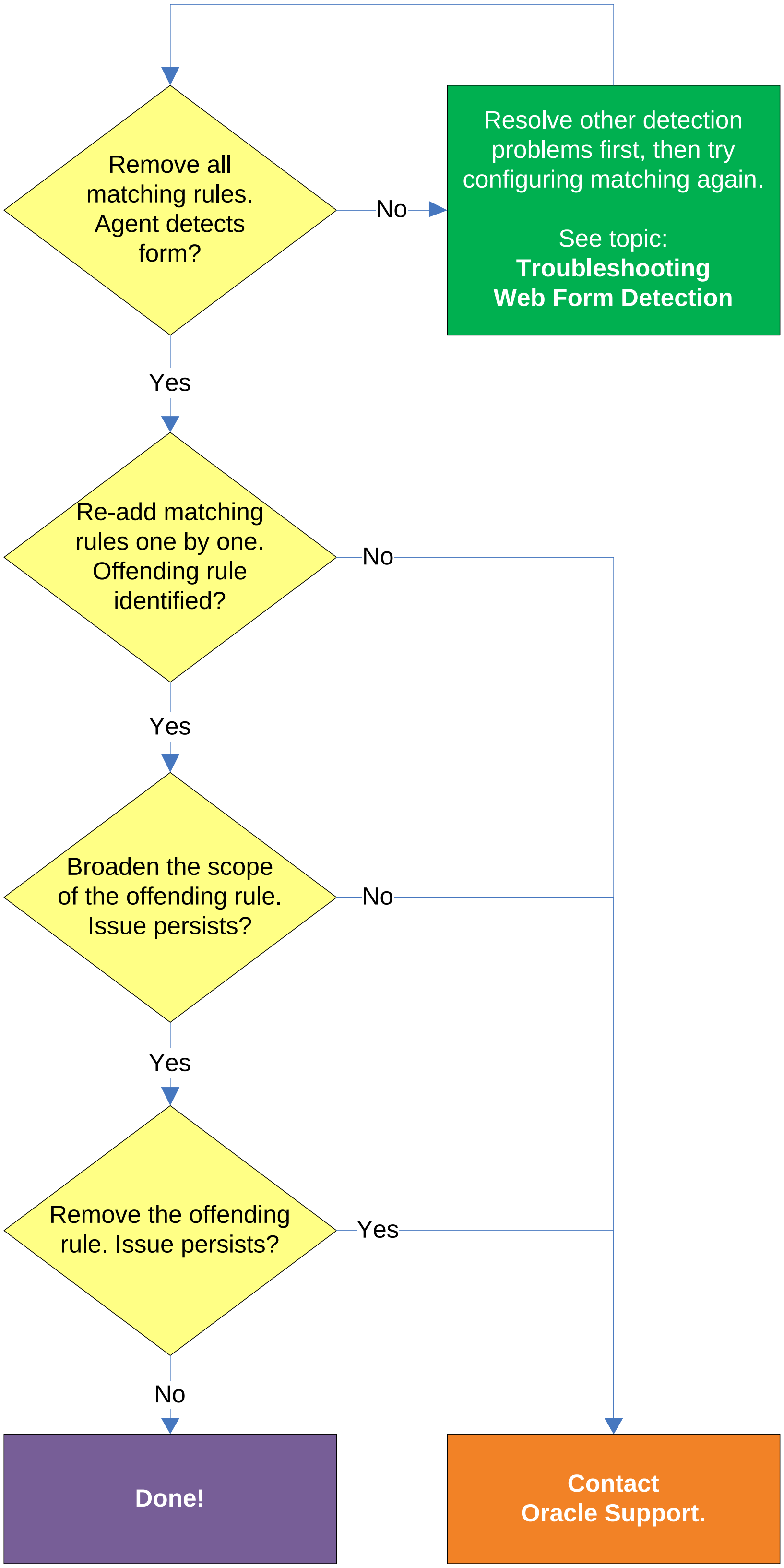
- If pressing **Enter** submits the credentials, remove the “submit” input field definition from the template. When the “submit” field definition is absent, the Agent will send its default submit action – the **Enter** keystroke.
- If pressing **Enter** does not submit the credentials, instruct the users to submit them manually.

Fields populated erratically?

If the Agent populates the fields erratically, i.e., inserts wrong, truncated, garbled, or blank values, one or more of the target element names might be non-unique or dynamic. In such case, use ordinals instead of element names to uniquely identify the elements to ESSO-LM.

If the above steps do not resolve the issue, see Contacting Oracle Support to obtain further assistance.

Troubleshooting Detection Matching


Remove all matching rules. Agent detects form?
No
Resolve other detection problems first, then try configuring matching again.
See topic: **Troubleshooting Web Form Detection**
Yes
Re-add matching rules one by one. Offending rule identified?
No
Yes
Broaden the scope of the offending rule. Issue persists?
No
Yes
Remove the offending rule. Issue persists?
Yes
No
Done!
Contact Oracle Support.

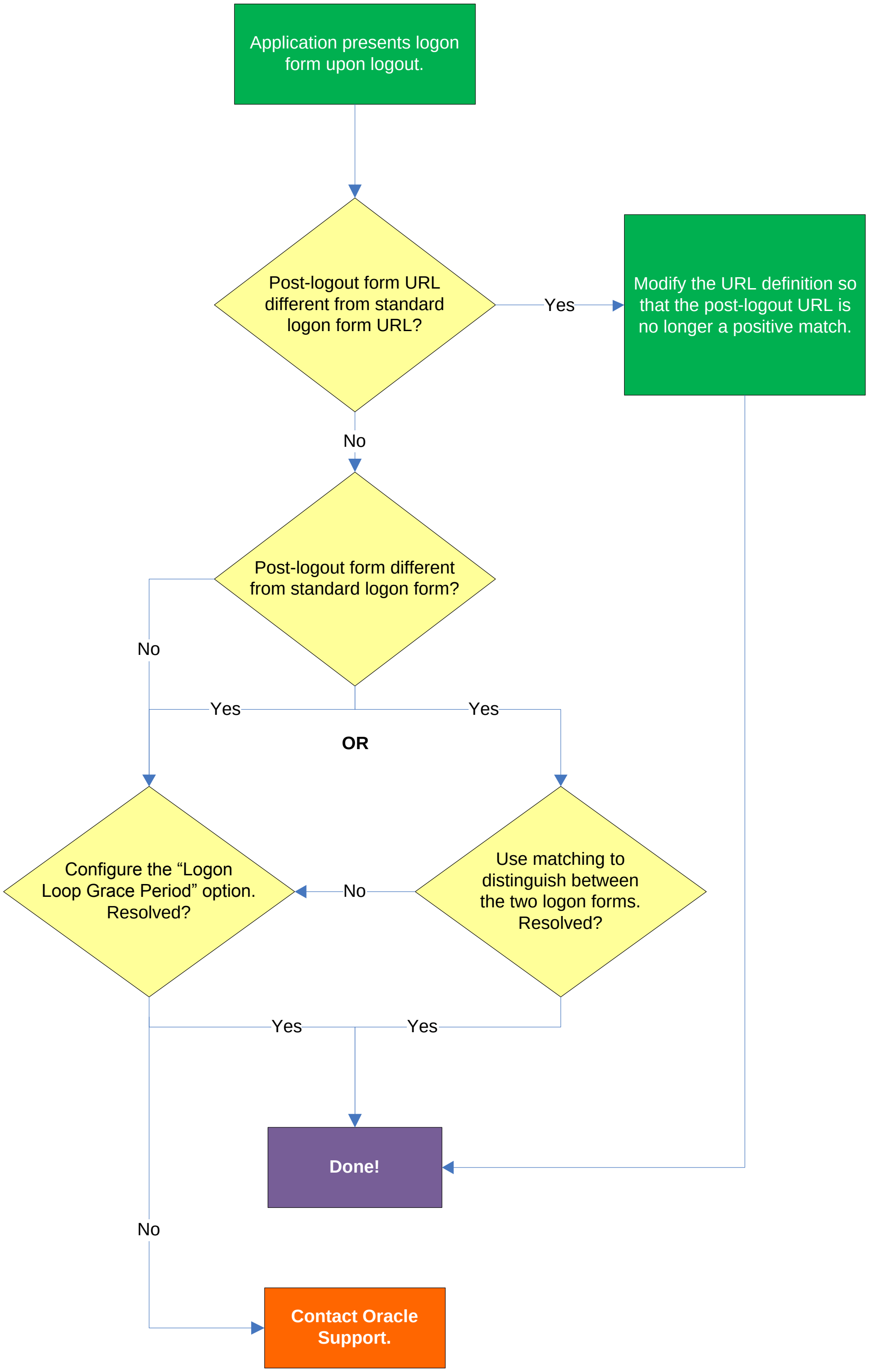
Removing all matching rules restores form detection?

If removing all matching rules from the form definition in the template restores proper form detection, re-add the rules one by one until the offending rule is identified. Then, either broaden the scope of the offending rule (for example, specify an HTML container that's "one-above" in the site's DOM hierarchy than the currently specified one), or remove the offending rule completely and see if detection is restored. If disabling matching completely does not restore detection, the problem lies elsewhere. See Troubleshooting Web Form Detection to resolve the problem.

If the above steps do not resolve the issue, see Contacting Oracle Support to obtain further assistance.

Troubleshooting a Logon Loop

Some applications display their logon form upon logout, which causes ESSO-LM to recognize the logon form and automatically log you back on to the application. This creates an endless “logon loop” preventing you from logging out of the application. To prevent this loop from occurring, the administrator may choose to enable the logon grace period feature which forbids ESSO-LM from logging on to an application within set time period since the last logon.


Application presents logon form upon logout.
Post-logout form URL different from standard logon form URL?
Yes
Modify the URL definition so that the post-logout URL is no longer a positive match.
No
Post-logout form different from standard logon form?
No
Yes
Yes
OR
Configure the “Logon Loop Grace Period” option. Resolved?
No
Use matching to distinguish between the two logon forms. Resolved?
Yes
Yes
Done!
No
Contact Oracle Support.

Post-logout form URL from standard logon form URL?

If the URL of the logon form presented upon logout is different from the URL of the standard logon form, modify the URL definition in the template so that the URL of the post-logon form is no longer a positive match. See Configuring URL Detection Criteria for instructions.

Post-logout form different from standard logon form?

If the logon form presented upon logout is sufficiently different from the application's standard logon form, use matching to uniquely distinguish between the two logon forms. See Configuring Additional Field Detection Criteria for instructions.

Configuring the "Logon Loop Grace Period" option resolves logon loop?

If the post-logout form cannot be uniquely distinguished from the standard logon form, configure a grace period that will prevent the Agent from automatically logging on to the same application if the specified grace period has not fully elapsed.

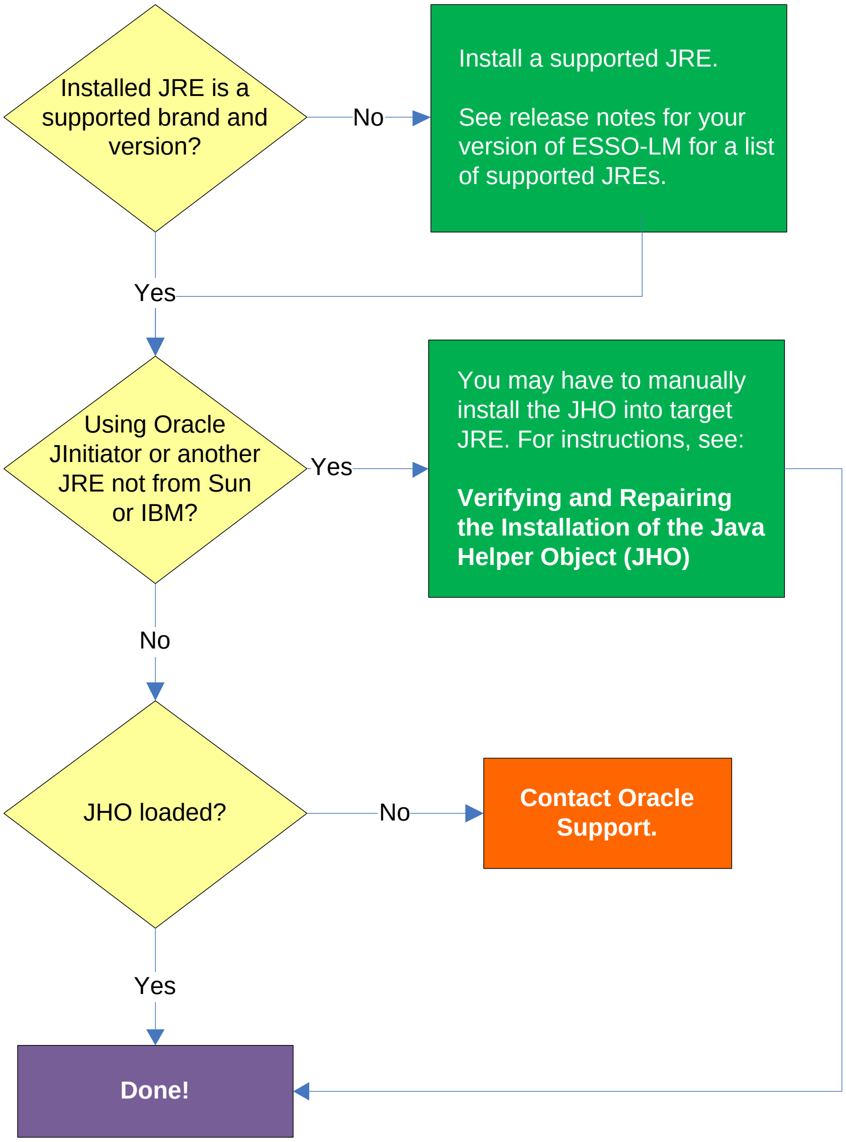
To configure the logon loop grace period timer, do the following:

1. In the ESSO-LM Administrative Console, open the desired template and select the **Miscellaneous** tab.
2. In the **Logon Loop Grace Period** field, select the desired mode of operation from the drop-down list:
   - **Prompt** – if the Agent detects the application's logon form while the grace period is in effect, the Agent will prompt the user whether to complete the logon or ignore the application.
   - **Silent** – if the Agent detects the application's logon form while the grace period is in effect, the Agent will ignore the application and not log the user on.
   - **None** – deactivates the grace period timer. Agent will respond to the application every time it detects the application's logon form.
3. Do one of the following, depending on what you want the Agent to do while the grace period is in effect:
   - If you want the Agent to log the user on each time the launch of the application's executable is detected, select the **Reset for each process** check box.
   - If you would like the Agent to ignore the application until the grace period has expired, leave the **Reset for each process** check box blank.
4. Save your changes and commit them to your repository, if applicable.

If the above steps do not resolve the issue, see Contacting Oracle Support to obtain further assistance.

Troubleshooting Java Application Issues

Use the steps below to diagnose and resolve issues specific to Java applications.


Installed JRE is a supported brand and version?
No
Install a supported JRE.
See release notes for your version of ESSO-LM for a list of supported JREs.
Yes
Using Oracle JInitiator or another JRE not from Sun or IBM?
Yes
You may have to manually install the JHO into target JRE. For instructions, see:
Verifying and Repairing the Installation of the Java Helper Object (JHO)
No
JHO loaded?
No
Contact Oracle Support.
Yes
Done!

Installed JRE is a supported brand and version?

Refer to the release notes for your version of ESSO-LM for a list of supported JREs. If the installed JRE is not supported, you must either upgrade ESSO-LM to a release that supports your current JRE, or replace the current JRE with a version supported by your release of ESSO-LM.

Using Oracle JInitiator or another JRE not made by Sun or IBM?

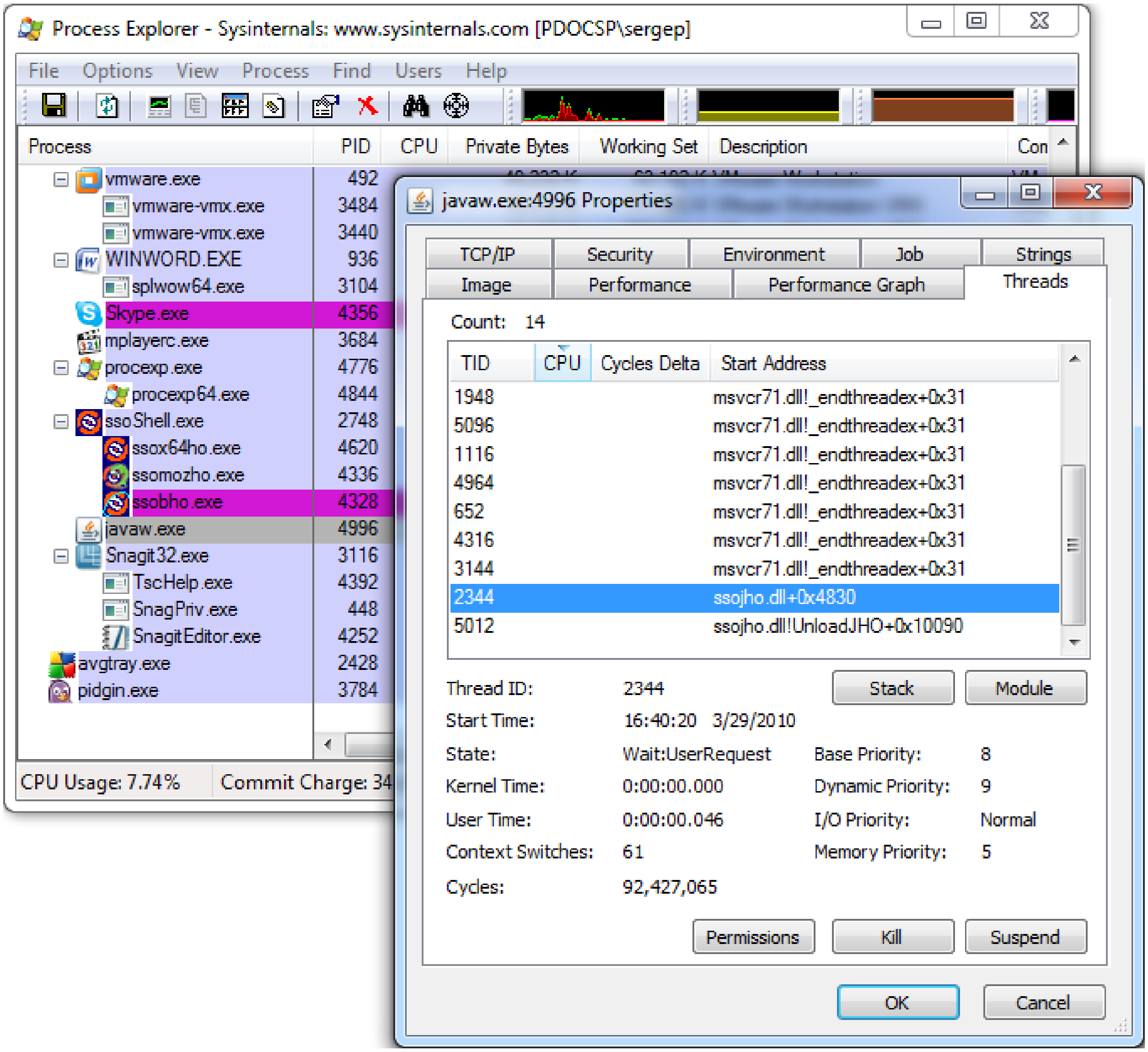
If you are using Oracle JInitiator or another JRE not made by Sun Microsystems or IBM, you might need to manually install the JHO into the JRE. For more information, see Verifying and Repairing the Installation of the Java Helper Object (JHO).

Note: While no issues have been reported when deploying ESSO-LM with non-Sun or non-IBM JREs other than Oracle JInitiator, Oracle does not support nor warrant the proper functioning of ESSO-LM with such JREs.

JHO loaded?

In certain situations, a configuration issue might prevent the JHO from loading when the Java application is launched, even though it has been installed properly. To verify that the JHO is installed and running, use a process viewer tool, such as Microsoft Spy++ (included with Microsoft Visual Studio) or SysInternals Process Explorer (http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx) to verify that the JVM executable has spawned the `ssojho.dll` child thread.

The example below shows the properties box of the Sun JVM executable `javaw.exe` in Process Explorer showing the `ssojho.dll` child thread running:



Process	PID
vmware.exe	492
vmware-vmx.exe	3484
vmware-vmx.exe	3440
WINWORD.EXE	936
splwow64.exe	3104
Skype.exe	4356
mplayerc.exe	3684
procexp.exe	4776
procexp64.exe	4844
ssoShell.exe	2748
ssox64ho.exe	4620
ssomozho.exe	4336
ssobho.exe	4328
javaw.exe	4996
Snagit32.exe	3116
TscHelp.exe	4392
SnagPriv.exe	448
SnagitEditor.exe	4252
avgtray.exe	2428
pidgin.exe	3784

CPU Usage: 7.74%   Commit Charge: 34

Count: 14

TID	CPU	Cycles Delta	Start Address
1948			msvcr71.dll!_endthreadex+0x31
5096			msvcr71.dll!_endthreadex+0x31
1116			msvcr71.dll!_endthreadex+0x31
4964			msvcr71.dll!_endthreadex+0x31
652			msvcr71.dll!_endthreadex+0x31
4316			msvcr71.dll!_endthreadex+0x31
3144			msvcr71.dll!_endthreadex+0x31
2344			ssojho.dll+0x4830
5012			ssojho.dll!UnloadJHO+0x10090

Thread ID: 2344
Start Time: 16:40:20 3/29/2010
State: Wait:UserRequest   Base Priority: 8
Kernel Time: 0:00:00.000   Dynamic Priority: 9
User Time: 0:00:00.046   I/O Priority: Normal
Context Switches: 61   Memory Priority: 5
Cycles: 92,427,065

If you have completed all of the previous troubleshooting steps and the JHO is not loading for the target JRE, contact Oracle Support for assistance.

Verifying and Repairing the Installation of the Java Helper Object (JHO)

> **Note:** The instructions below only apply to the Oracle JInitiator and other JREs made by companies other than Sun Microsystems or IBM. When using Sun and IBM JREs, the ESSO-LM installer performs the required installation automatically.

ESSO-LM recognizes Java applications through the Java Helper Object (JHO). If the JHO is not installed correctly, ESSO-LM will not recognize Java applications. Use the steps below to verify that the JHO has been installed correctly and correct a damaged installation if necessary.

Since multiple Java Runtime Environments (JREs) can coexist on the same machine, the ESSO-LM installer installs the JHO into the JRE currently set as default (the current `JavaHome`).

The default JRE directory is specified by the following registry setting:

```
HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime
   Environment\<JRE_version>\JavaHome
```

The JRE version is specified in the following registry setting:

```
HKEY_LOCAL_MACHINE\Software\JavaSoft\Java Runtime Environment
```

If the user starts a Java application from the default JRE that the JHO has been installed in, ESSO-LM will recognize that Java application. However, if the application is started from a JRE other than the default one, ESSO-LM will not recognize the application. Keep in mind that each JRE installs the JVM executables `java.exe` and `javaw.exe`. Copies of these executables are also installed into the `<%SYSTEMROOT%>\SYSTEM32` directory. ESSO-LM will only recognize Java applications if the JVM is launched from the JRE directory.

Verifying the JHO Installation in a JRE

To check whether the JHO has been installed into a JRE, do the following:

1. Navigate to the lib folder inside the JRE's directory. For example:

   ```
   C:\Program Files\Java\jre1.6.0_01\lib
   ```

2. Open the file `accessibility.properties` in a text editor and check whether the following property is present:

   ```
   assistive_technologies=<other_helpers>,com.passlogix.vgo.ho.jho
   ```

   If the property is present and the JHO string shown above is part of its value, the JHO has been installed into this JRE. If the property or file do not exist, stop here; the JHO has not been installed into this JRE.

3. Check whether the following files are present in the JRE:
   - `<JRE_dir>\bin\ssojho.dll`
   - `<JRE_dir>\lib\ext\jho.jar`
   - `<JRE_dir>\lib\ext\jaccess.jar`

If any of the files are missing, the JHO is not installed correctly.

Repairing a Damaged JHO Installation

To manually install the JHO into a JRE other than the default one, do the following:

1. Make sure the JHO has been correctly installed into the default JRE as described above.
   If the JHO is not installed, run the ESSO-LM installer, select the **Modify** option, and select the Java feature under **Extensions → Helper Objects**, then follow the installer prompts to complete the installation.
2. Add the JHO string to the `assistive_technologies` property in the following file:

   ```
   <JRE_dir>\lib\accessibility.properties
   ```

   Note: If the property and/or file does not exist, create them.

3. Copy the JHO files listed in step 3 in the previous section from the default JRE to their respective locations within the target JRE.

Restricting the Java Helper Object (JHO) to Specific Java Runtime Environments (JREs)

ESSO-LM provides registry settings that allow you to create inclusion and exclusion rules that ESSO-LM will use to decide whether to load the JHO for the target application. You can specify, via regular expressions, the Java Virtual Machine (JVM) executables, JAR files, and command-line parameters that you want the JHO to consider.

When configuring these settings, keep the following in mind:

- If the value for a given setting is omitted, the specified default is used; if a value is set, all non-matching values are ignored.
- The JHO processes the inclusion rules first, followed by the exclusion rules.
- The N suffix is a unique numerical identifier that bundles settings belonging to a specific application. The JHO will process the bundles sequentially in ascending order.
  The N suffix is a positive integer starting at 1.
- You can determine the application's host JVM executable and launch command using the Trace Logging utility. The beginning of the Java trace log will indicate the host JVM and the launch command for the application.

The settings are located at the following path in the Windows Registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Passlogix\Extensions\AccessManager\JHO
```

The settings are divided into the following categories:

- JHO Inclusion Rules
- JHO Exclusion Rules
- Allowed Java Runtime Environment (JRE) Versions

JHO Inclusion Rules

Use these settings to specify the host JVM executables, application JAR files, and command-line parameters that will cause the JHO to load for the target application. The values are case-insensitive regular expressions.

Setting	Description
`JhoIncludeHostNameN`	Specifies the application's host JVM executable(s). For example, if you want the JHO to load when the application is hosted by a JVM executable named `java.exe` or `javaw.exe`, set the value as follows: `JhoIncludeHostName1=.*javaw?\.exe` No-value default: All executables are accepted (i.e., JHO will load for any executable).
`JhoIncludeHostCommandLineN`	Specifies the command used to launch the application. The command string usually includes the full path and name of the JVM executable, the JAR file, as well as any required command-line parameters. For example, if you want the JHO to only load when the application's JAR file is named `Login.jar` (while the rest of the command is allowed to vary), set the value as follows: `JhoIncludeHostCommandLine1=.*Login\.jar.*` No-value default: All command strings accepted (i.e., JHO will load for any command).

JHO Exclusion Rules

Use these settings to specify the host JVM executables, application JAR files, and command-line parameters that will cause the JHO to ignore the target application. The values are case-insensitive regular expressions.

Setting	Description
`JhoExcludeHostNameN`	Specifies the application's host JVM executable(s). For example, if you want the JHO to ignore the application when the host JVM executable is named "java.exe" or "javaw.exe", set the value as follows: `JhoExcludeHostName1=.*javaw?\.exe` No-value default: Blank (i.e., nothing will cause the JHO to ignore the application)
`JhoExcludeHostCommandLineN`	Specifies the command used to launch the application. The command string usually includes the the full path and name of the JVM executable, the JAR file, and any required command-line parameters. For example, if you want the JHO to ignore the application when its JAR file is named "Login.jar" (but the rest of the command is non-consequential), set the value as follows: `JhoExcludeHostCommandLine1=.*Login\.jar.*` No-value default: Blank. (i.e., nothing will cause the JHO to ignore the application)

Allowed Java Runtime Environment (JRE) Versions

These settings allow you to specify the desired JRE/JDK versions for which the JHO will load; all versions falling outside of the specified range will be ignored and the JHO will not load.

The values for both settings must follow the format `x.y.z`, where:

- `x` is the major revision
- `y` is the minor revision
- `z` is denotes the release type (feature, maintenance, or update) and build ID of the installed JRE (for example, `1.6.0_07)`

Setting	Description
`JhoMinimumJavaVersion`	Specifies the lowest allowed JRE/JDK version. Default value: `1.2` (the earliest JRE version supported by the JHO)
`JhoMaximumJavaVersion`	Specifies the highest allowed JRE/JDK version. Default value: Blank (i.e., no upper JRE version limit is imposed)

Customizing the Event Response Behavior of the Java Helper Object (JHO)

In order to troubleshoot and optimize ESSO-LM performance when responding to Java applications, you can modify the settings that govern the Java Helper Object's response to specific events when a Java application has been detected.

The settings are located within the following key in the Windows registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Passlogix\Extensions\AccessManager
```

Event Response Configuration Settings

Setting	Description
`JhoHierarchyEventProcessing`	Determines which Java hierarchy events are recognized. Set the flag as follows: `HIERARCHY_EVENT_CHANGED = 0x1` The above value instructs the JHO to recognize all hierarchy events.
`JhoEventWaitTimeout`	Determines the event processing timeout for JHO controls (in milliseconds). The default value of 0 instructs the JHO to wait indefinitely.
`JhoWindowEventProcessing`	Determines which Java window events are recognized. This flag is a combination of the following values: • `WINDOW_EVENT_OPENED = 0x1` • `WINDOW_EVENT_CLOSED = 0x2` • `WINDOW_EVENT_ACTIVATED = 0x4` • `WINDOW_EVENT_DEACTIVATED = 0x8` • `WINDOW_EVENT_CLOSING = 0x10` • `WINDOW_EVENT_ICONIFIED = 0x20` • `WINDOW_EVENT_DEICONIFIED = 0x40` By default, all window events are recognized.

Setting	Description
`JhoComponentEventProcessing`	Determines which Java component events are recognized. This flag is a combination of the following values: • `COMPONENT_EVENT_SHOWN = 0x1` • `COMPONENT_EVENT_HIDDEN = 0x2` • `COMPONENT_EVENT_ADDED = 0x4` • `COMPONENT_EVENT_REMOVED = 0x8` By default, all component events are recognized.
`JhoInjectType`	Determines the injection type used by the JHO to submit data to the controls. This flag takes one of the following values: • `INJECT_TYPE_DEFAULT = 0` • `INJECT_TYPE_METHOD = 1` • `INJECT_TYPE_ACCESSIBLE = 2` • `INJECT_TYPE_NONACCESSIBLE = 3` • `INJECT_TYPE_ROBOT = 4` By default this flag is set to `INJECT_TYPE_DEFAULT`, in which case the JHO attempts injection using each of following methods, in the order shown, until injection is successful: • `INJECT_TYPE_METHOD` (if an appropriate set method had been found for the control) • `INJECT_TYPE_ACCESSIBLE` (if the control supports accessibility) • `INJECT_TYPE_NONACCESSIBLE` • `INJECT_TYPE_ROBOT` **Note:** For combo and list boxes, the JHO always uses `INJECT_TYPE_METHOD`.

Recommended JHO Event Response Configuration Defaults

We recommend the following default settings on new installations of ESSO-LM:

- `JhoWindowEventProcessing=0x3`
- `JhoComponentEventProcessing=0xB`
- `JhoHierarchyEventProcessing=0x0`

These values instruct the JHO to recognize the following events:

- `WINDOW_EVENT_OPENED (0x1)`
- `WINDOW_EVENT_CLOSED (0x2)`
- `COMPONENT_EVENT_SHOWN (0x1)`
- `COMPONENT_EVENT_HIDDEN (0x2)`
- `COMPONENT_EVENT_REMOVED (0x8)`